Dictyostelium developmental time course

Jakub Orzechowski Westholm, Bart Edelbroek

2024-06-13

Contents

Set up

```
rna_colors <- brewer.pal(6,"OrRd")
names(rna_colors) <- c("rna00", "rna02", "rna04", "rna06", "rna08", "rna10")
prot_colors <- brewer.pal(5,"GnBu")
names(prot_colors) <- c("prot00", "prot02", "prot04", "prot08", "prot10")
combined_colors <- c(rna_colors,prot_colors)

kelly_color <- "grey50"
names(kelly_color) <- "lacal_kelly"

rep_colors <- brewer.pal(n = 4, name = "Set2")
names(rep_colors) <- c("a", "b", "c", "d")

heatmap_color_scale <- colorRamp2(4:-4, colorRampPalette(brewer.pal(11,"RdYlBu"))(9))
```

```
counts_file  <- "data/rna/counts.txt"
annot_dir <- paste(getwd(), "dictybase_20200923/", sep="/")
gene_uniprot_file <- paste(annot_dir, "DDB-GeneID-UniProt.txt", sep="")
gene_info_file <- paste(annot_dir, "gene_information.txt", sep="")
gene_annot_file <- paste(annot_dir, "gene_association.dictyBase", sep="")

go_terms <- AnnotationDbi::select(GO.db, keys= keys(GO.db, keytype="GOID"), columns=c("GOID", "TERM") )
```

```
    dplyr::rename("GO_TERM"="GOID", "GO_DESCRIPTION"="TERM")

read_tsv(gene_uniprot_file) %>% rename_with(function(x){gsub(" ", "_", x)}) %>% dplyr::rename("GENE_ID"=
gene_uniprot <- gene_uniprot[!duplicated(gene_uniprot$GENE_ID),]
#These are genes that are mostly annotated as pseudogenes, but where we find expression in the proteomi
missing_annotations <- cbind.data.frame(GENE_ID = c("DDB_G0290059","DDB_G0293770","DDB_G0282207","DDB_G
                                                                                    UniProt_ID = c("Q54GN0"
gene_uniprot$UniProt_ID[match(missing_annotations$GENE_ID,gene_uniprot$GENE_ID)] <- missing_annotations$

gene_id_to_uniprot <- gene_uniprot$UniProt_ID
names(gene_id_to_uniprot) <- gene_uniprot$GENE_ID


read_tsv(gene_annot_file, comment = "!", col_names = F) -> gene_annot
read_tsv(gene_info_file) %>% rename_with(function(x){gsub(" ", "_", x)}) -> gene_info

gene_annot_tab <- gene_info %>%
    full_join(gene_annot, by=c("GENE_ID"="X2")) %>%
    full_join(gene_uniprot, by=c("GENE_ID"="GENE_ID")) %>%
    left_join(go_terms, by=c("X5"="GO_TERM")) %>%
    dplyr::select(!c("X1", "X3", "X4", "X10", "X11", "X12", "X13", "X14", "X15", "X16", "X17", "Name")) 
    dplyr::rename("GO_TERM"="X5", "GO_REF"="X6", "GO_EVIDENCE"="X7", "GO_TYPE"="X9", "PROT_ANNOT"="X8")

gene_info$UniProt_ID <- gene_id_to_uniprot[gene_info$GENE_ID]

uniprot_to_gene_name_tmp <- gene_annot_tab %>%
    group_by(UniProt_ID) %>%
    summarise(Gene_Name = dplyr::first(Gene_Name))
uniprot_to_gene_name <- uniprot_to_gene_name_tmp$Gene_Name
names(uniprot_to_gene_name) <- uniprot_to_gene_name_tmp$UniProt_ID
rm(uniprot_to_gene_name_tmp)

# Gene Ontology annotations the are not informative, e.g. "Unknown", "Biological Process" etc.
remove_annots <- c("GO:0008150", "GO:0007582", "GO:0044699", "GO:0000004", # Biological process
                                    "GO:0005575", # Cellular component
                                    "GO:0003674" # Molecular function
)

go_tab <- gene_annot_tab %>%
    filter(GO_EVIDENCE != "IBA") %>%
    filter(!(GO_TERM %in% remove_annots)) %>%
    dplyr::select("UniProt_ID", "GO_TERM", "GO_TYPE", "GO_DESCRIPTION")


go_mat_tmp <- pmin(table(go_tab$GO_DESCRIPTION, go_tab$UniProt_ID),1) # binary matrix with GO terms as 
go_mat <- matrix(go_mat_tmp, nrow = nrow(go_mat_tmp), ncol = ncol(go_mat_tmp)) # ugly hack, fix
colnames(go_mat) <- colnames(go_mat_tmp)
rownames(go_mat) <- rownames(go_mat_tmp)
rm(go_mat_tmp)

# Create Gene to GO list,
gene_to_go_list <- split(go_tab$GO_TERM, go_tab$UniProt_ID)
gene_to_go_list <- lapply(gene_to_go_list,function(x){unique(x)})
```

RNA-seq analysis

```
counts <- as.matrix(read_tsv(counts_file, quote = "\"", comment = "#"))
colnames(counts) <- gsub("merged_mapped/SI-2309-|.bam", "", colnames(counts))
rownames(counts) <- counts[,1]
counts <- counts[,-1*1:6]
class(counts) <- "numeric"
counts <- counts[,grep("FS146", colnames(counts))] # only use wt data
counts <- counts[apply(counts,1,sum)>0,] # remove genes with no reads

# Create metadata table, from file names
meta_data_rna <- do.call("rbind",sapply(colnames(counts), function(x){strsplit(x, "-")})) %>%
    data.frame() %>%
    dplyr::rename(Genotype = X1, Time = X2, Rep = X3) %>%
    mutate(Time = str_pad(gsub("h", "", Time), 2, pad="0")) %>%
    mutate(sample_id = rownames(.)) %>%
    mutate(sample_id = gsub("FS146", "wt", sample_id)) %>%
    mutate(sample_id = gsub("-h", "_a", sample_id)) %>%
    mutate(sample_id = gsub("-i", "_b", sample_id)) %>%
    mutate(sample_id = gsub("-j", "_c", sample_id)) %>%
    mutate(sample_id = gsub("-k", "_d", sample_id)) %>%
    mutate(sample_id = gsub("-", "_", sample_id)) %>%
    mutate(Rep = gsub("h", "a", Rep)) %>%
    mutate(Rep = gsub("i", "b", Rep)) %>%
    mutate(Rep = gsub("j", "c", Rep)) %>%
    mutate(Rep = gsub("k", "d", Rep)) %>%
    arrange(Time, Rep)

# Reorder samples
counts <- counts[, rownames(meta_data_rna)]

# Reformat sample names
colnames(counts) <- meta_data_rna$sample_id
rownames(meta_data_rna) <- meta_data_rna$sample_id

rownames(counts) <- gene_id_to_uniprot[rownames(counts)]
counts <- aggregate(counts, list(rownames(counts)),sum)
rownames(counts) <- counts[,1]
counts <- counts[,-1]
```

Differentially expressed genes

```
max_pval <- 1e-2

dds <- DESeqDataSetFromMatrix(countData = counts, colData = meta_data_rna, design = formula(~ Time))
dds <- DESeq(dds, test="LRT", reduced=~1)
res <- results(dds)
LFCrna <- cbind.data.frame(lfcShrink(dds, coef = "Time_02_vs_00", type = "apeglm")$log2FoldChange,
                                                    lfcShrink(dds, coef = "Time_04_vs_00", type = "apeg
                                                    lfcShrink(dds, coef = "Time_06_vs_00", type = "apeg
                                                    lfcShrink(dds, coef = "Time_08_vs_00", type = "apeg
```

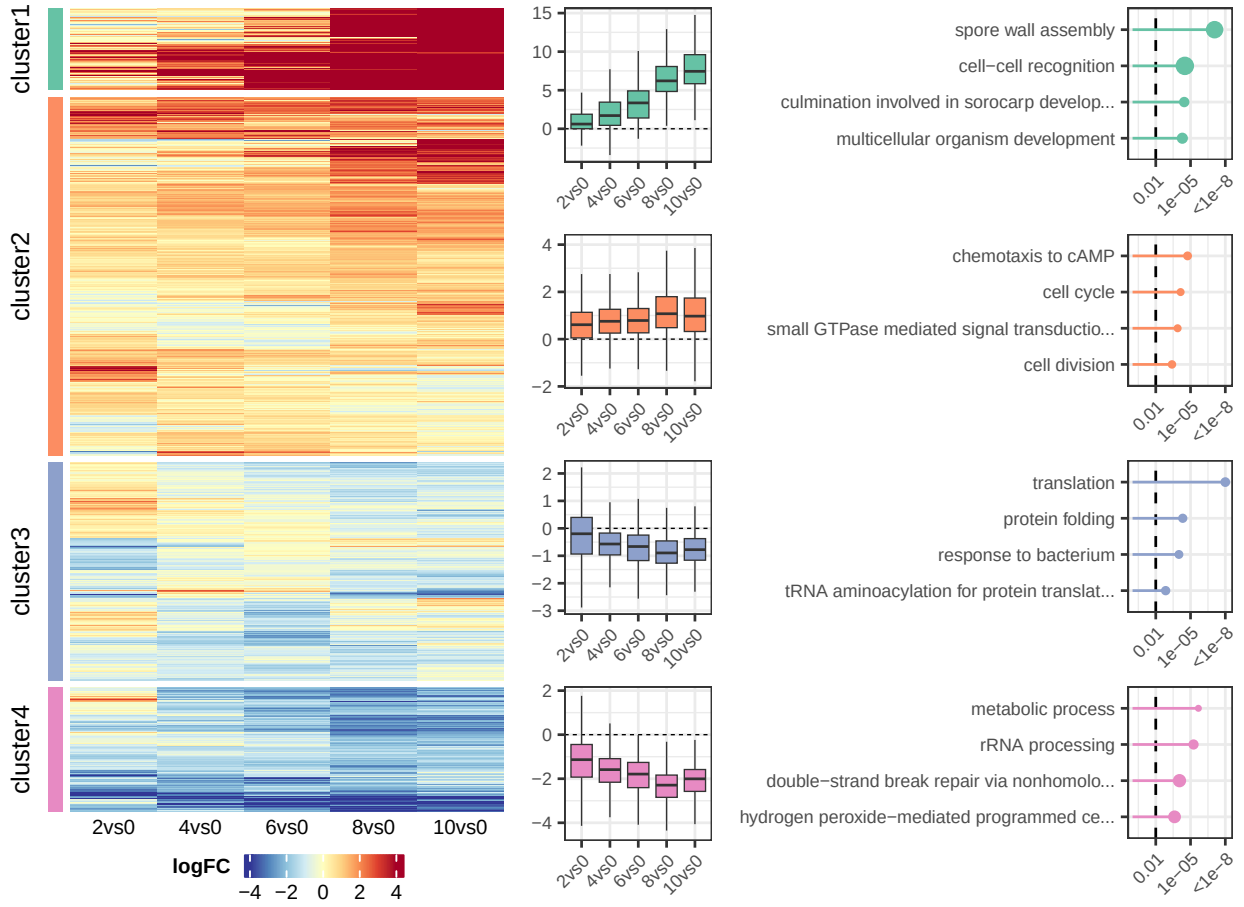
```
                                                    lfcShrink(dds, coef = "Time_10_vs_00", type = "apeg
colnames(LFCrna) <- c("2vs0","4vs0","6vs0","8vs0","10vs0")
dev_genes <- rownames(res)[which(res$padj < max_pval)]



# Do regularized log transformation, then plot all genes affected by developmental time in a heatmap
norm_rna_data  <- vst(dds, blind=F)
linear_rna_normalised <- t(t(2^assay(norm_rna_data))/colSums(2^assay(norm_rna_data)))*1e6

out_rna_table <- cbind.data.frame(
    gene_info[match(rownames(LFCrna),gene_info$UniProt_ID),],
    format(LFCrna,scientific = FALSE),
    res$padj,
    format(linear_rna_normalised,scientific = FALSE)
    )
write.table(out_rna_table, "tables/rna_logFC.tsv", sep = "\t", row.names = F, col.names = T, quote = F)

plot_data <- LFCrna[dev_genes,]
k <- 4
clust_method <- "ward.D"
plotClust <- hclust(dist(plot_data), method = clust_method)



plot_data <- plot_data[plotClust$order,]
plotClust <- hclust(dist(plot_data), method = clust_method)
rna_clusters <- cutree(plotClust, k = k)
rna_clusters <- reorder_clusters(plot_data, rna_clusters)
plot_data <- plot_data[names(rna_clusters),]

clust_annot <- data.frame("cluster"=paste0("cluster",rna_clusters))
rownames(clust_annot) <- rownames(plot_data)
clust_colors <- brewer.pal(n = max(rna_clusters), name = "Set2")
names(clust_colors) <- paste0("cluster",1:max(rna_clusters))

rna_h <- Heatmap(plot_data,
                                cluster_columns = FALSE,
                                cluster_rows = FALSE,
                                show_row_names = FALSE,
                                col = heatmap_color_scale,
                                #column_names_side = c("top"),
                                column_names_rot = 0,   column_names_centered = T,
                                heatmap_legend_param = list(direction = "horizontal",
                                                                                    title = "logFC"
                                                                                    title_gp = gpar
                                                                                    title_position =
                                                                                    labels_gp = gpa
                                                                                    grid_height = u
                                row_split = clust_annot$cluster,
                                left_annotation = rowAnnotation(cluster = clust_annot$cluster,
                                                                                          width =
                                                                                          simple_
```

```
col = li
show_leg
show_ann
                                column_names_gp = gpar(fontsize = 8),
                                row_title_gp = gpar(fontsize = 10)
)
a <- grid.grabExpr(draw(rna_h, heatmap_legend_side = "bottom"))

plotdat <- cbind.data.frame(LFCrna[names(rna_clusters),],rna_clusters)
plotdat <- pivot_longer(plotdat, cols = 1:5)
plotdat$name <- factor(plotdat$name, levels = c("2vs0","4vs0","6vs0","8vs0","10vs0"))
plotdat$rna_clusters <- factor(plotdat$rna_clusters, levels = c(1:k))
plotdat %>%
    group_by(name,rna_clusters)%>%
    mutate(value2 = filter_lims(value))%>%
    ggplot(aes(name,value2, fill = rna_clusters))+
    geom_hline(yintercept = 0, lty = "33", lwd = 0.25)+
    geom_boxplot(lwd = 0.25, outlier.shape = NA)+
    scale_fill_manual(values = unname(clust_colors))+
    facet_wrap(~ rna_clusters, ncol = 1, scales = "free")+
    theme(strip.text.x = element_blank(), axis.title = element_blank(), axis.text.x = element_text(angle


comb_GO <- data.frame()
for(cluster in 1:max(rna_clusters)){
    cat(paste("\n\n### Cluster" ,cluster, "\n\n"))

    cluster_genes <- factor(as.integer(rownames(res) %in% names(rna_clusters[rna_clusters==cluster])))
    names(cluster_genes) <- rownames(res)

    # How many gene have any annotation?
    genes_in_cluster <- names(cluster_genes)[cluster_genes==1]
    genes_with_annot <- names(gene_to_go_list)
    genes_in_cluster_with_annot <- intersect(genes_with_annot, genes_in_cluster)
    cat(paste(length(genes_in_cluster_with_annot), "/", length(genes_in_cluster), "have annotations.\n")

    GOdata <- new("topGOdata",
                               ontology = "BP",
                               allGenes = cluster_genes,
                               description = "",
                               nodeSize = 5,
                               gene2GO = gene_to_go_list,
                               annot = annFUN.gene2GO)
    results_go <- runTest(GOdata, algorithm = "weight01", statistic = "fisher")

    top_go_res <- GenTable(GOdata, Fis = results_go, topNodes = 100)

    comb_GO <- rbind(comb_GO, cbind(top_go_res,"cluster" = cluster))
}
```

```
##
##
## ### Cluster 1
```

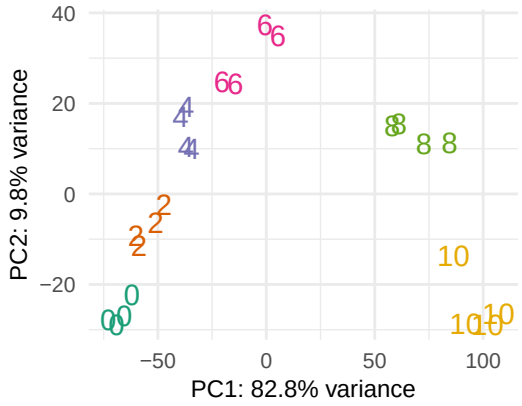
```
##
## 485 / 869 have annotations.
##
##
## ### Cluster 2
##
## 2573 / 3812 have annotations.
##
##
## ### Cluster 3
##
## 1638 / 2313 have annotations.
##
##
## ### Cluster 4
##
## 905 / 1316 have annotations.
```

```
comb_GO$Fis <- as.numeric(comb_GO$Fis)
comb_GO$Enrichment <- comb_GO$Significant/comb_GO$Expected
comb_GO <- comb_GO[comb_GO$Enrichment>1,]

plot_GO <- comb_GO %>%
  slice_min(order_by = Fis, n = 4, by = cluster)
plot_GO <- plot_GO[-16,]
plot_GO$logP <- -log(as.numeric(plot_GO$Fis))
plot_GO$logP[is.na(plot_GO$logP)] <- -log(1e-7)
plot_GO$logP[plot_GO$logP > -log(1e-8)] <- -log(1e-8)
plot_GO$cluster <- factor(paste0("cluster",plot_GO$cluster))
plot_GO$Term <- factor(plot_GO$Term, levels = rev(plot_GO$Term))
c <- ggplot(plot_GO, aes(x = Term, y = logP, size = Enrichment, color = cluster))+
    facet_wrap(~ cluster, scales = "free", ncol = 1)+
    geom_hline(yintercept = -log(0.01), lty="dashed")+
    geom_point(stat = 'identity')+scale_color_manual(values = clust_colors)+
    geom_segment(aes(y = 0, yend = logP, xend = Term), size = 0.5)+
    scale_size(range = c(0.5,3), limits = c(1,20), breaks =c(1,5,20))+
    scale_y_continuous(limits = c(0,max(plot_GO$logP)+0.5), breaks = -log(c(0.01,1e-5,1e-8)), labels = 
    coord_flip()+
    theme(legend.position = "none",
                axis.text = element_text(size = 7),
                axis.text.x = element_text(angle = 45, hjust = 1),
                axis.title = element_blank(),
                strip.text.x = element_blank())
d <- ggarrange(a,b,c, nrow = 1, widths = c(2.5,1,2.5))
d
```




```
ggsave("plots/mRNA_logFC_heatmap.svg", d, "svg", width = 7.1, height = 5, units = "in", dpi = 320)


write.table(comb_GO[comb_GO$Fis<0.01,c(1,2,7,6,3,4,5,8)], "tables/rna_GOterms.tsv", sep = "\t", row.nam
```

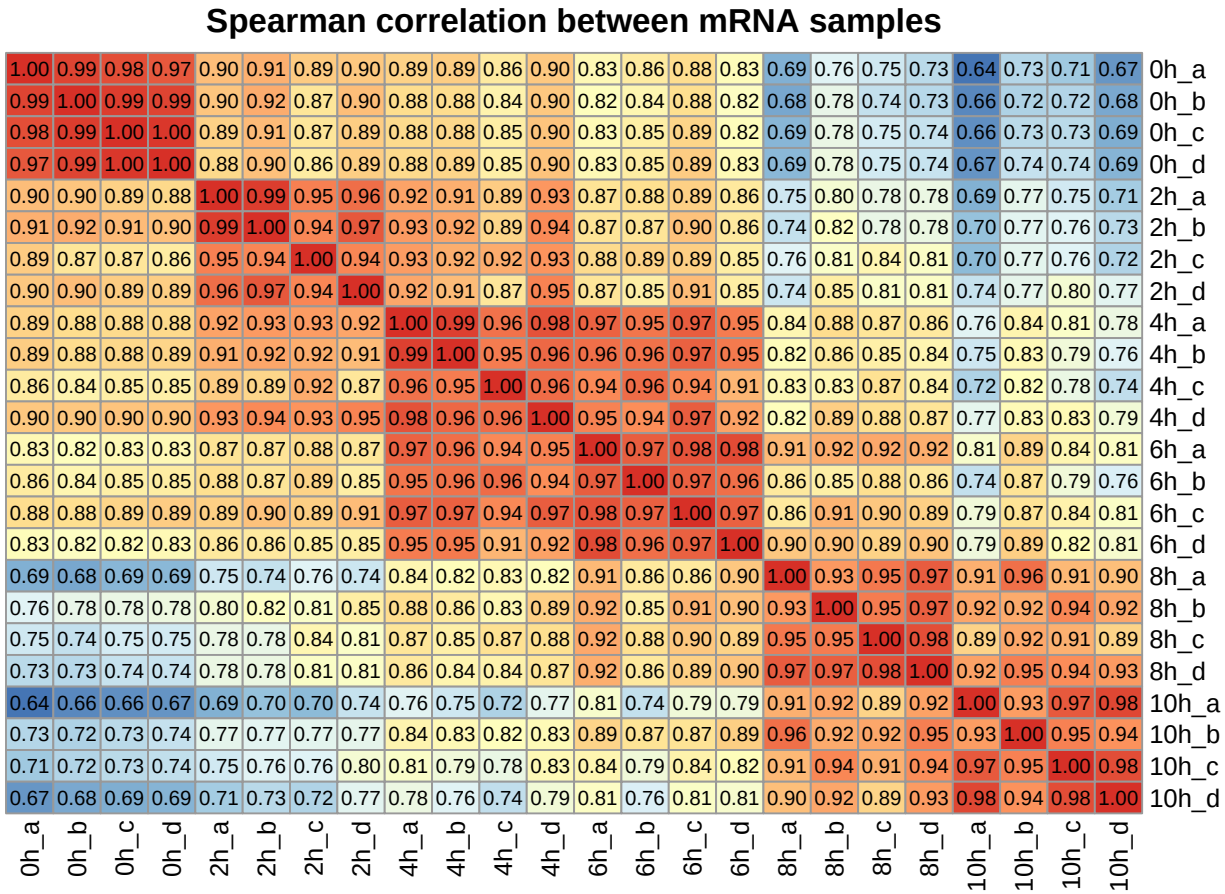
```
ntop = 500
rv = rowVars(as.matrix(assay(norm_rna_data)))
select <- order(rv, decreasing = T)[seq_len(min(ntop, length(rv)))]
mat <- t(assay(norm_rna_data)[select,])
pca <- prcomp(mat)
pcaMat <- data.frame(pca$x)
pcaMat$ID <- factor(rep(seq(0,10,2),each=4))
#pcaMat$ID <- factor(pcaMat$ID, levels = unique(pcaMat$ID[c(seq(1,length(pcaMat$ID),2),seq(2,length(pca
variances <- ((pca$sdev^2) / (sum(pca$sdev^2)))*100
a <- ggplot(pcaMat, aes(x = PC1, y = PC2, color = ID, label = ID)) +
    geom_text()+
    theme_minimal()+
    xlab(paste0("PC1: ", round(variances[1], digits = 1),"% variance"))+
    ylab(paste0("PC2: ", round(variances[2], digits = 1),"% variance"))+
    scale_color_brewer(palette = "Dark2")+
    theme(legend.position = "none",title = element_text(size = 9), axis.text = element_text(size = 8))
a
```




```
ggsave("plots/mRNA_PCA.svg",a,width = 2.8,height = 2.2)
```

Between sample correlation

```
cor_matrix <- cor(x=log2(linear_rna_normalised), method = "spearman")
rownames(cor_matrix) <- substr(rownames(cor_matrix),4,10)
colnames(cor_matrix) <- substr(colnames(cor_matrix),4,10)

#svglite("plots/rna_corr_matrix.svg",width = 6.8,height = 5)
pheatmap(cor_matrix,
                 cluster_cols = F,
                 cluster_rows = F,
                 scale = "none",
                 main = "Spearman correlation between mRNA samples",
                 use_raster = F, display_numbers = T, number_color = "black", legend = F)
```




```
#dev.off()
```

```
#The Rosengarten raw fastq data was pre-processed in the same way as our RNA-seq data
rosengarten_counts <- "data/rna/rosengarten2015_counts.txt"
rosengarten <- as.matrix(read_tsv(file = rosengarten_counts, quote = "\"", comment = "#"))
colnames(rosengarten) <- gsub("star_map.SRR15934|.bam", "", colnames(rosengarten))
rownames(rosengarten) <- gene_id_to_uniprot[rosengarten[,1]]
rosengarten <- rosengarten[,-(1:6)]
class(rosengarten) <- "numeric"
rosengarten <- aggregate(rosengarten, list(rownames(rosengarten)),sum)
rownames(rosengarten) <- rosengarten[,1]
rosengarten <- rosengarten[,-1]
r_metadata <- data.frame("ID"=c(rbind(c(24:34),c(43:53))),
                                                    "Time"=as.factor(rep(c(0:10), each=2)),
                                                    "Rep"=rep(c("a","b"), 11))
rosengarten <- rosengarten[,as.character(r_metadata$ID)]
rds <- DESeqDataSetFromMatrix(countData = rosengarten,  colData = r_metadata, design = formula(~ Time))
rds <- DESeq(rds, test="LRT", reduced=~1)
rres <- results(rds)
norm_rosengarten_data <- vst(rds, blind=F)
```

```
max_pval <- 1e-2
clust_method <- "ward.D"
```

```
venn_colors <- c(our_DE = "#377EB8", rosengarten_DE = "#E41A1C", both_DE = "#984EA3") #Set colors for t

sc_rosengarten <- scale_rows(assay(norm_rosengarten_data)) #Get z-scores of normalized counts for Rosen
colnames(sc_rosengarten) <- paste0(r_metadata$Time,"h.",r_metadata$Rep)
sc_de_genes <- scale_rows(assay(norm_rna_data)) #Get z-scores of normalized counts for our data

rosengarten_dev_genes <- rownames(rres)[rres$padj < max_pval & rownames(rres) %in% rownames(res)] #Sele
dev_genes_filt <- rownames(res)[res$padj < max_pval & rownames(res) %in% rownames(rres)] #Select genes

#Create 3 dataframes: combined zscores of both datasets for genes that are SigDE in both, only rosengar
combined_de <- cbind(
    sc_rosengarten[na.omit(rosengarten_dev_genes[rosengarten_dev_genes %in% dev_genes_filt]),],
    sc_de_genes[na.omit(rosengarten_dev_genes[rosengarten_dev_genes %in% dev_genes_filt]),]
)
rosengarten_de_only <- sc_rosengarten[rosengarten_dev_genes[!(rosengarten_dev_genes %in% dev_genes_filt]
our_de_only <- sc_de_genes[dev_genes_filt[!(dev_genes_filt %in% rosengarten_dev_genes)],]

#Cluster these three dataframes separately, and get the genes in the clustered order
combined_genes_clustered <- rownames(combined_de)[hclust(dist(combined_de),method = clust_method)$order]
rosengarten_genes_clustered <- rownames(rosengarten_de_only)[hclust(dist(rosengarten_de_only),method = c
our_genes_clustered <- rownames(our_de_only)[hclust(dist(our_de_only),method = clust_method)$order]

#Combine the zscores of both datasets but plot only the genes that are SigDE in either of the datasets,
plot_data <- cbind(sc_rosengarten[c(rosengarten_genes_clustered,combined_genes_clustered,our_genes_clust
                                   sc_de_genes[c(rosengarten_genes_clustered,combined_genes_clustered
#Add annotations to show in which datasets these genes were SigDE
row_annot <- data.frame("DE"=c(rep(names(venn_colors)[2],length(rosengarten_genes_clustered)),
                               rep(names(venn_colors)[3],length(combined_g
                               rep(names(venn_colors)[1],length(our_genes_
))
row_annot$DE <- factor(row_annot$DE, levels = c("rosengarten_DE" ,"both_DE", "our_DE"))

svglite("plots/mRNA_rosengarten_heatmap.svg", width = 5, height = 3.6)
Heatmap(plot_data, name = "foo",
                cluster_columns = FALSE,
                cluster_rows = FALSE,
                show_row_names = FALSE,
                use_raster = T,
                col = heatmap_color_scale, show_column_names = F,
                #column_names_side = c("top"),
                column_names_rot = 0,   column_names_centered = T,
                heatmap_legend_param = list(title = "zscore",
                                            labels_gp = gpar(fontsize = 6),
                                            title_gp = gpar(fontsize = 6, fo
                                            grid_width = unit(2,"mm")),
                top_annotation = columnAnnotation(
                    time =  c(rep(0:10,each = 2),rep(seq(0,10,2), each = 4)),
                    col = list(time = colorRamp2(c(0,5,10),c("#EAF6DF","#b2df8a","#33a02c"))),
                    height = unit(1, "mm"),
                    simple_anno_size_adjust = TRUE,
                    annotation_legend_param = list(time = list(title = "time(h)",
```

```
                    show_annotation_name = F
                ),
                column_split = factor(c(rep("rosengarten",22),rep("our_study",24)), levels = c("rosenga
                #cluster_column_slices = FALSE,
                #column_gap = unit(c(rep(0,10),2,rep(0,5)), "mm"),
                row_split = row_annot$DE,
                row_title_rot = 0,
                left_annotation = rowAnnotation(cluster = row_annot$DE,
                                                                    col = list(cluster = ve
                                                                    width = unit(1, "mm"),
                                                                    simple_anno_size_adjust
                                                                    show_legend = F,
                                                                    show_annotation_name = 
                column_title_gp = gpar(fontsize = 8),
                row_title_gp = gpar(fontsize =8)
)

decorate_heatmap_body("foo",{grid.rect(gp = gpar(fill = "transparent", col = "black", lwd = 1))}, slice
decorate_heatmap_body("foo",{grid.rect(gp = gpar(fill = "transparent", col = "black", lwd = 1))}, slice
decorate_heatmap_body("foo",{grid.rect(gp = gpar(fill = "transparent", col = "black", lwd = 1))},row_sl
decorate_heatmap_body("foo",{grid.rect(gp = gpar(fill = "transparent", col = "black", lwd = 1))}, row_s
#dev.off()
#svglite("plots/rosengarten_venn.svg", width = 3, height = 3)
venn_data <- c(length(our_genes_clustered),
                            length(rosengarten_genes_clustered),
                            length(combined_genes_clustered))
names(venn_data) <- c(names(venn_colors[1:2]),paste0(names(venn_colors[1]),"&",names(venn_colors[2])))
plot(euler(venn_data),fills=venn_colors,labels = c())
#dev.off()


milestones <- read.table("data/milestone_genes.txt", sep = "\t", header = FALSE)
milestones$V1 <- gene_id_to_uniprot[milestones$V1]
milestones <- na.omit(milestones[,1:5])
milestone_IDs <- intersect(milestones$V1[
    milestones$V3%in%c("noagg","ripple","lag","tag")
    ], rownames(sc_de_genes))
row_annot <- data.frame(DE = milestones$V4[match(milestone_IDs, milestones$V1)],
                                                dir = milestones$V5[match(milestone_IDs, milestones$V1)]
row_annot$DE <- factor(row_annot$DE, levels = c("noagg","ripple","lag","tag","tip","slug","Mhat","cul",

up_down_colors <- c(down = "#377EB8", up = "#E41A1C")

#svglite("plots/milestones_RNA.svg",width = 3.5, height = 2.1)
Heatmap(sc_de_genes[milestone_IDs,], name = "zscore",
                cluster_columns = FALSE,
                cluster_rows = FALSE,
                show_row_names = FALSE,
                use_raster = T,
                col = heatmap_color_scale, show_column_names = F,
                row_split = row_annot$DE, row_title_rot = 0,
```

```
                heatmap_legend_param = list(title = "zscore",
                                                                    labels_gp = gpar(fontsize = 6),
                                                                    title_gp = gpar(fontsize = 6, fo
                                                                    grid_width = unit(2,"mm")),
                left_annotation = rowAnnotation(cluster = row_annot$dir,
                                                                        col = list(cluster = up
                                                                        width = unit(1, "mm"),
                                                                        simple_anno_size_adjust
                                                                        show_legend = F,
                                                                        show_annotation_name = F
                top_annotation = columnAnnotation(
                    time =  rep(seq(0,10,2), each = 4),
                    col = list(time = colorRamp2(c(0,5,10),c("#EAF6DF","#b2df8a","#33a02c"))),
                    height = unit(1, "mm"),
                    simple_anno_size_adjust = TRUE,
                    annotation_legend_param = list(time = list(title = "time(h)",
                    show_annotation_name = F
                ))
#dev.off()
nrow(sc_de_genes[milestone_IDs,])
```

```
## [1] 864
```

Proteomics analysis

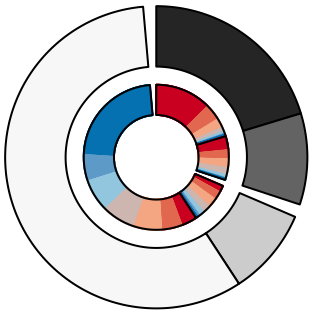
```
mass_spec_file <- "data/protein/fragpipe/combined_protein.tsv"
conv <- "dictybase_20200923/DDB-GeneID-UniProt.txt"
meta_data_protein <- read.table("data/protein/index_ms.tsv", header = T, row.names = 1)
meta_data_protein <- meta_data_protein %>%
    filter(strain == "wt") %>%
    mutate(sample_id = paste(strain, "_",  timepoint, "h_", biorep) %>% gsub(" ", "", .)) %>%
    mutate(timepoint = str_pad(timepoint, 2, pad="0")) %>%
    dplyr::rename(Genotype = strain, Time = timepoint, Rep = biorep)
rownames(meta_data_protein) <- meta_data_protein$sample_id

mass_spec <- as.matrix(read_tsv(mass_spec_file, quote = "", comment = "#"))[,c(2,75:77,81:89,78:80)]
rownames(mass_spec) <- mass_spec[,1]
mass_spec <- mass_spec[,-1]
class(mass_spec) <- "numeric"
colnames(mass_spec) <- meta_data_protein$sample_id
mass_spec <- mass_spec[rownames(mass_spec)%in%gene_id_to_uniprot,]
mass_spec_zero <- mass_spec
mass_spec[mass_spec == 0] <- NA
non_imputed_proteins <- rownames(na.omit(mass_spec))

min_no_na <- 3 # require at least this many non NA values for any branch (i.e. time point)
```

```
branch_list <- list(grep("_0h",colnames(mass_spec)),
                                    grep("_2h",colnames(mass_spec)),
                                    grep("_4h",colnames(mass_spec)),
                                    grep("_8h",colnames(mass_spec)),
                                    grep("_10h",colnames(mass_spec)))

non_na_samples <-   apply(mass_spec,1,
                                              function(x){
                                                  max(sapply(branch_list, function(b){length(which(!is
                                              })

#mass_spec_impute <- as.matrix(impute.MinProb(log2(mass_spec[non_na_samples>=min_no_na,])))
#write.table(mass_spec_impute, "tables/protein_log2_raw.tsv", sep = "\t", row.names = T, col.names = T,
mass_spec_impute <- as.matrix(read.table("tables/protein_log2_raw.tsv", sep = "\t", header = T, row.name

linear_mass_spec_impute <- t(t(2^mass_spec_impute)/colSums(2^mass_spec_impute))*1e6


a <- rownames(na.omit(mass_spec))[rownames(na.omit(mass_spec)) %in% gene_uniprot$UniProt_ID]
b <- rownames(mass_spec_impute)[rownames(mass_spec_impute) %in% gene_uniprot$UniProt_ID]
c <- rownames(mass_spec)[rownames(mass_spec) %in% gene_uniprot$UniProt_ID]

df <- data.frame(uniprotID = na.omit(unique(gene_uniprot$UniProt_ID)), msQuant = "not identified")
rownames(df) <- df$uniprotID
df$msQuant[match(c,df$uniprotID)] <- "quantified in some replicates"
df$msQuant[match(b,df$uniprotID)] <- "quantified across all replicates after imputation"
df$msQuant[match(a,df$uniprotID)] <- "quantified across all replicates"

table(df$msQuant)
```

```
##
##                                  not identified
##                                            7061
##                  quantified across all replicates
##                                            2478
## quantified across all replicates after imputation
##                                            1185
##                     quantified in some replicates
##                                            1142
```

```
mean_rna_expression <- round(rowMeans(assay(norm_rna_data)),0)

df$rnaQuant <- 0
df$rnaQuant[match(names(mean_rna_expression[names(mean_rna_expression)%in%df$uniprotID]), df$uniprotID)]

df <- df[order(match(df$msQuant,c("not identified","quantified in some replicates","quantified across al
df$number <- 1:nrow(df)


col_fun1 <- colorRamp2(c(5, 7, 9, 11), c("#0571b0", "#92c5de", "#f4a582","#ca0020"))
sel_rows <- seq(1, nrow(df), 10)

subs_df <- df[sel_rows,]
```

```
circos.clear()
#svglite("plots/donut.svg", width = 2.6, height = 2.6)
circos.par(start.degree = 90, gap.degree = c(0,5,0,5)[1:length(unique(subs_df$msQuant))])
circos.heatmap.initialize(matrix(subs_df$rnaQuant),
                                                    split = factor(subs_df$msQuant, levels = unique(subs
                                                    cluster = F)
circos.track(ylim = range(subs_df$rnaQuant), bg.col = c("#252525","#636363","#cccccc","#f7f7f7"), track
set_track_gap(mm_h(2))
circos.heatmap(matrix(subs_df$rnaQuant),
                              split = factor(subs_df$msQuant, levels = unique(subs_df$msQuant)),
                              cluster = F,
                              col = col_fun1, bg.border = "black", bg.lwd = 1, track.height = 0.2)
```



```
#dev.off()
```

```
pivot_longer(data.frame(mass_spec_zero), cols = everything(), )%>%
    mutate(uniprotID= rep(rownames(mass_spec_zero),each = 15))%>%
    group_by(uniprotID)%>%
    mutate(missing_vals = sum(value==0),
                log_max_val = log(max(value)+1))%>%
    distinct(uniprotID, .keep_all = TRUE)%>%
    filter(missing_vals < 15)%>%
    #mutate(missing_vals = factor(missing_vals, levels = 0:13))%>%
    ggplot(aes(x = log_max_val, group = missing_vals, fill = missing_vals))+
    geom_area(aes(y = ..count..), stat = "bin", bins = 20, col = "black")+
    labs( x = "log(max. quantfication)")+
    scale_fill_viridis_c("Missing values\nper protein", direction = -1, breaks = c(0,4,8,12))->a
ggsave("plots/prot_expression_missing_vals.svg",a,"svg",width = 3.2,height = 2.2)
```

```
## Warning: The dot-dot notation (‘..count..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(count)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

Differentially expressed proteins

```
max_pval <- 0.01

design <- model.matrix(~ Time, data=meta_data_protein)

# Limma model fitting and F-test.
fit <- lmFit(mass_spec_impute, design)
fit2 <- eBayes(fit)

# Proteins regulated by developmental time
tt_dev <- topTable(fit2, coef=2:5, number=nrow(mass_spec_impute))
LFCprot <- tt_dev[,1:4]
LFCprot <- na.omit(LFCprot)
LFCprot <- LFCprot[,1:4]
colnames(LFCprot) <- c("2vs0","4vs0","8vs0","10vs0")


dev_prot <- rownames(tt_dev)[tt_dev$adj.P.Val < max_pval
                                                    #& apply(abs(LFCprot[,1:4]), 1, max) > 1
]


out_prot_table <- cbind.data.frame(
    gene_info[match(rownames(LFCprot),gene_info$UniProt_ID),],
    format(LFCprot,scientific = FALSE),
    tt_dev$adj.P.Val,
    format(linear_mass_spec_impute,scientific = FALSE)
    )
write.table(out_prot_table, "tables/prot_logFC.tsv", sep = "\t", row.names = F, col.names = T, quote = F

common_expressed <- intersect(rownames(mass_spec_impute),rownames(counts))
gene_numbers <- data.frame(
    name=c("all protein coding genes",
                "quantified transcripts","DE transcripts","% DE of quantified transcripts",
                "quantified proteins","DE proteins", "% DE of quantified proteins",
                "genes with quantified transcript and protein (common genes)", "common genes with DE t

    number=c(length(na.omit(unique(gene_id_to_uniprot))),
                    nrow(counts),length(dev_genes),round(length(dev_genes)/nrow(counts)*100),
                    nrow(mass_spec_impute),length(dev_prot),round(length(dev_prot)/nrow(mass_spec_impu
                    length(common_expressed),length(intersect(common_expressed,intersect(dev_prot,dev_g
                    length(intersect(common_expressed,union(dev_prot,dev_genes))))
)


plot_data <- LFCprot[dev_prot,]

k <- 3
clust_method <- "ward.D"
plotClust <- hclust(dist(plot_data), method = clust_method)
plot_data <- plot_data[plotClust$order,]
plotClust <- hclust(dist(plot_data), method = clust_method)
```

```
prot_clusters <- cutree(plotClust, k = k)
prot_clusters <- reorder_clusters(plot_data, prot_clusters)
plot_data <- plot_data[names(prot_clusters),]

clust_annot <- data.frame("cluster"=paste0("cluster",prot_clusters))
rownames(clust_annot) <- rownames(plot_data)
clust_colors <- brewer.pal(n = max(prot_clusters), name = "Set2")
names(clust_colors) <- paste0("cluster",1:max(prot_clusters))

prot_h <- Heatmap(plot_data,
                                    cluster_columns = FALSE,
                                    cluster_rows = FALSE,
                                    show_row_names = FALSE,
                                    col = heatmap_color_scale,
                                    #column_names_side = c("top"),
                                    column_names_rot = 0,   column_names_centered = T,
                                    heatmap_legend_param = list(direction = "horizontal",
                                                                                      title = "lo
                                                                                      title_gp = 
                                                                                      title_posit
                                                                                      labels_gp =
                                                                                      grid_height
                                    row_split = clust_annot$cluster,
                                    left_annotation = rowAnnotation(cluster = clust_annot$cluster,
                                                                                              wid
                                                                                              sim
                                                                                              col
                                                                                              sho
                                                                                              sho
                                    column_names_gp = gpar(fontsize = 8),
                                    row_title_gp = gpar(fontsize = 10)
)
a <- grid.grabExpr(draw(prot_h, heatmap_legend_side = "bottom"))

plotdat <- cbind.data.frame(LFCprot[names(prot_clusters),],prot_clusters)
plotdat <- pivot_longer(plotdat, cols = 1:4)
plotdat$name <- factor(plotdat$name, levels = c("2vs0","4vs0","6vs0","8vs0","10vs0"))
plotdat$prot_clusters <- factor(plotdat$prot_clusters, levels = c(1:k))
plotdat %>%
    group_by(name,prot_clusters)%>%
    mutate(value2 = filter_lims(value))%>%
    ggplot(aes(name,value2, fill = prot_clusters))+
    geom_hline(yintercept = 0, lty = "33", lwd = 0.25)+
    geom_boxplot(lwd = 0.25, outlier.shape = NA)+
    scale_fill_manual(values = unname(clust_colors))+
    facet_wrap(~ prot_clusters, ncol = 1, scales = "free")+
    theme(strip.text.x = element_blank(), axis.title = element_blank(), axis.text.x = element_text(angl


comb_GO <- data.frame()
for(cluster in 1:max(prot_clusters)){
    cat(paste("\n\n### Cluster" ,cluster, "\n\n"))
```

```
    cluster_genes <- factor(as.integer(rownames(tt_dev) %in% names(prot_clusters[prot_clusters==cluster]
    names(cluster_genes) <- rownames(tt_dev)

    # How many gene have any annotation?
    genes_in_cluster <- names(cluster_genes)[cluster_genes==1]
    genes_with_annot <- names(gene_to_go_list)
    genes_in_cluster_with_annot <- intersect(genes_with_annot, genes_in_cluster)
    cat(paste(length(genes_in_cluster_with_annot), "/", length(genes_in_cluster), "have annotations.\n"

    GOdata <- new("topGOdata",
                                ontology = "BP",
                                allGenes = cluster_genes,
                                description = "",
                                nodeSize = 5,
                                gene2GO = gene_to_go_list,
                                annot = annFUN.gene2GO)
    results_go <- runTest(GOdata, algorithm = "weight01", statistic = "fisher")

    top_go_res <- GenTable(GOdata, Fis = results_go, topNodes = 100)

    comb_GO <- rbind(comb_GO, cbind(top_go_res,"cluster" = cluster))
}
```

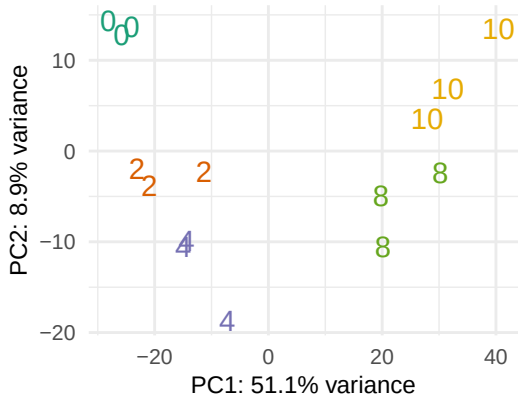
```
##
##
## ### Cluster 1
##
## 137 / 179 have annotations.
##
##
## ### Cluster 2
##
## 190 / 230 have annotations.
##
##
## ### Cluster 3
##
## 215 / 263 have annotations.
```

```
comb_GO$Fis <- as.numeric(comb_GO$Fis)
comb_GO$Enrichment <- comb_GO$Significant/comb_GO$Expected
comb_GO <- comb_GO[comb_GO$Enrichment>1,]

plot_GO <- comb_GO %>%
  slice_min(order_by = Fis, n = 4, by = cluster)

plot_GO$logP <- -log(as.numeric(plot_GO$Fis))
plot_GO$logP[is.na(plot_GO$logP)] <- -log(1e-7)
plot_GO$cluster <- factor(paste0("cluster",plot_GO$cluster))
plot_GO$Term <- factor(plot_GO$Term, levels = rev(plot_GO$Term))
c <- ggplot(plot_GO, aes(x = Term, y = logP, size = Enrichment, color = cluster))+
    facet_wrap(~ cluster, scales = "free", ncol = 1)+
    geom_hline(yintercept = -log(0.01), lty="dashed")+
```

```
    geom_point(stat = 'identity')+scale_color_manual(values = clust_colors)+
    geom_segment(aes(y = 0, yend = logP, xend = Term), size = 0.5)+
    scale_size(range = c(0.5,3),limits = c(1,20), breaks =c(1,5,20))+
    scale_y_continuous(limits = c(0,max(plot_GO$logP)+0.5), breaks = -log(c(0.01,1e-04,1e-06,1e-08)), la
    coord_flip()+
    theme(legend.position = "none",
                axis.text = element_text(size = 7),
                axis.text.x = element_text(angle = 45, hjust = 1),
                axis.title = element_blank(),
                strip.text.x = element_blank())

d <- ggarrange(a,b,c, nrow = 1, widths = c(2.5,1,4))
d

ggsave("plots/prot_logFC_heatmap.svg", d, "svg", width = 6.8, height = 3.4, units = "in")

write.table(comb_GO[comb_GO$Fis<0.01,c(1,2,7,6,3,4,5,8)], "tables/prot_GOterms.tsv", sep = "\t", row.nar
```

PCA

```
ntop = 300
rv = rowVars(as.matrix(mass_spec_impute))
select <- order(rv, decreasing = T)[seq_len(min(ntop, length(rv)))]
mat <- t(mass_spec_impute[select,])
pca <- prcomp(mat)
pcaMat <- data.frame(pca$x)
pcaMat$ID <- factor(rep(c(0,2,4,8,10), each = 3))
#pcaMat$ID <- factor(pcaMat$ID, levels = unique(pcaMat$ID[c(seq(1,length(pcaMat$ID),2),seq(2,length(pca
variances <- ((pca$sdev^2) / (sum(pca$sdev^2)))*100
a <- ggplot(pcaMat, aes(x = PC1, y = PC2, color = ID, label = ID)) +
    geom_text()+
    theme_minimal()+
    xlab(paste0("PC1: ", round(variances[1], digits = 1),"% variance"))+
    ylab(paste0("PC2: ", round(variances[2], digits = 1),"% variance"))+
    scale_color_manual(values = brewer.pal(6, "Dark2")[-4])+
    theme(legend.position = "none",title = element_text(size = 9), axis.text = element_text(size = 8))
a
```



```
ggsave("plots/prot_PCA.svg",a,width = 2.8,height = 2.2)
```

Between sample correlation

```
cor_matrix <- cor(x=mass_spec_impute, method = "spearman")
rownames(cor_matrix) <- substr(rownames(cor_matrix),4,10)
colnames(cor_matrix) <- substr(colnames(cor_matrix),4,10)

#svglite("plots/prot_corr_matrix.svg",width = 6.8,height = 3.5)
pheatmap(cor_matrix,
                 cluster_cols = F,
                 cluster_rows = F,
                 scale = "none",
                 main = "Spearman correlation between protein samples",
                 use_raster = F, display_numbers = T, number_color = "black", legend = F)
#dev.off()
```

```
plotdat <- scale_rows(linear_mass_spec_impute)

milestone_IDs <- intersect(milestones$V1[
    milestones$V3%in%c("noagg","ripple","lag","tag")
    ], rownames(plotdat))
row_annot <- data.frame(DE = milestones$V4[match(milestone_IDs, milestones$V1)],
                                                 dir = milestones$V5[match(milestone_IDs, milestones$V1)]
row_annot$DE <- factor(row_annot$DE, levels = c("noagg","ripple","lag","tag","tip","slug","Mhat","cul",

up_down_colors <- c(down = "#377EB8", up = "#E41A1C")

#svglite("plots/milestones_prot.svg", width = 3.8, height = 3)
Heatmap(plotdat[milestone_IDs,], name = "zscore",
                cluster_columns = FALSE,
                cluster_rows = FALSE,
                show_row_names = FALSE,
                use_raster = T,
                col = heatmap_color_scale, show_column_names = F,
                row_split = row_annot$DE, row_title_rot = 0,
                heatmap_legend_param = list(title = "zscore",
                                                                    labels_gp = gpar(fontsize = 6),
                                                                    title_gp = gpar(fontsize = 6, f
                                                                    grid_width = unit(2,"mm")),
                left_annotation = rowAnnotation(cluster = row_annot$dir,
                                                                        col = list(cluster = up
                                                                        width = unit(1, "mm"),
                                                                        simple_anno_size_adjust
                                                                        show_legend = F,
                                                                        show_annotation_name = F
                top_annotation = columnAnnotation(
                    time =  rep(c(0,2,4,8,10), each = 3),
                    col = list(time = colorRamp2(c(0,5,10),c("#EAF6DF","#b2df8a","#33a02c"))),
                    height = unit(1, "mm"),
                    simple_anno_size_adjust = TRUE,
```

```
                    annotation_legend_param = list(time = list(title = "time(h)",




                    show_annotation_name = F
                ))
#dev.off()
```

Comparison to other data

```
#Load kelly data and impute missing values as previously
kelly <- tibble(read.csv(file = "data/protein/Kelly_proteomics.csv", sep = ";")[,c(5,31:42)])

#Generate design matrix for Limma
cond <- as.factor(rep(c("dev","ctrl"), each = 6))
time <- as.factor(rep(c(8,0.5), 2, each = 3))
lev <- as.factor(paste0(cond,time))
design <- model.matrix(~0+lev)
colnames(design) <- gsub("lev","",colnames(design))

colnames(kelly) <- c("uniprotID",paste0(cond,time,paste0("_",1:3)))
kelly%>%
    separate_wider_delim(uniprotID,delim = ";",names = c("A",NA), too_few = "align_start",too_many = "d
    mutate(across(everything(), ~ replace(., . == "Filtered", 0)))%>%
    mutate(across(-c(A), as.numeric))%>%
    group_by(A)%>%
    summarise(across(everything(), sum))%>%
    column_to_rownames(var = "A") -> kelly
kelly <- kelly[rownames(kelly)%in%gene_id_to_uniprot,]
kelly[kelly == 0] <- NA

branchListR <- list(1:3,4:6,7:9,10:12)

non_na_samples <-   apply(kelly,1,
                                                    function(x){
                                                        max(sapply(branchListR, function(b){length(whic
                                                    })
kelly_impute <- as.matrix(impute.MinProb(log2(kelly[non_na_samples>=min_no_na,])))
```

```
## [1] 0.3933404
```

```
#Limma protein analysis as previously
fit <- lmFit(kelly_impute, design)
fit2 <- contrasts.fit(fit, makeContrasts("dev0.5-ctrl0.5","dev8-ctrl8", levels = design))
fit2 <- eBayes(fit2)
ke_dev <- data.frame(topTable(fit2, number=nrow(kelly_impute)))
LFCke <- ke_dev[,1:2]


ke_dev_prot <- rownames(ke_dev)[ke_dev$adj.P.Val < max_pval
```

```
                                                                    #& abs(ke_dev[,1]) > 1
]

ke_zscores <- data.frame(scale_rows(kelly_impute))[,c(10:12,4:6,7:9,1:3)]
prot_zscores <- data.frame(scale_rows(mass_spec_impute))

combined_de <- cbind(
    ke_zscores[ke_dev_prot[ke_dev_prot %in% dev_prot],],
    prot_zscores[ke_dev_prot[ke_dev_prot %in% dev_prot],]
)
ke_de_only <- ke_zscores[ke_dev_prot[!(ke_dev_prot %in% dev_prot)],]
our_de_only <- prot_zscores[dev_prot[!(dev_prot %in% ke_dev_prot)],]

#Cluster these three dataframes separately, and get the genes in the clustered order
combined_prot_clustered <- rownames(combined_de)[hclust(dist(combined_de),method = clust_method)$order]
kelly_prot_clustered <- rownames(ke_de_only)[hclust(dist(ke_de_only),method = clust_method)$order]
our_prot_clustered <- rownames(our_de_only)[hclust(dist(our_de_only),method = clust_method)$order]



plot_data <- na.omit(as.matrix(cbind(LFCke[c(kelly_prot_clustered,combined_prot_clustered,our_prot_clust
                                                    LFCprot[c(kelly_prot_clustered,combined_prot_cl
#svglite("plots/protein_kelly_heatmap_LFC.svg", width = 3.8, height = 2.2)
Heatmap(plot_data, name = "foo",
                cluster_columns = FALSE,
                cluster_rows = TRUE,
                show_row_names = FALSE,
                use_raster = T,
                col = heatmap_color_scale, show_column_names = F,
                column_names_side = c("top"),
                column_names_rot = 0,   column_names_centered = T,
                heatmap_legend_param = list(title = "logFC",
                                                                labels_gp = gpar(fontsize = 6),
                                                                    title_gp = gpar(fontsize = 6, f
                                                                    grid_width = unit(3,"mm")),
                top_annotation = columnAnnotation(
                    time =  c(c(0.5,8),seq(2,10,2)[-3]),
                    col = list(time = colorRamp2(c(0,5,10),c("#EAF6DF","#b2df8a","#33a02c"))),
                    height = unit(1, "mm"),
                simple_anno_size_adjust = TRUE,
                    annotation_legend_param = list(time = list(title = "time(h)",




                    show_annotation_name = F
                    ),
                column_split = factor(c(rep("kelly",2),rep("our_study",4)), levels = c("kelly","our_stu
                #cluster_column_slices = FALSE,
                #column_gap = unit(c(rep(0,10),2,rep(0,5)), "mm"),
                #row_split = row_annot$DE,
                row_title_rot = 0,
#               left_annotation = rowAnnotation(cluster = row_annot$DE,
```

```
#                                                                                col = list(cluster = ve
#                                                                                width = unit(1, "mm"),
#                                                                            simple_anno_size_adjust = T
#                                                                                show_legend = F,
#                                                                                show_annotation_name = 
                column_title_gp = gpar(fontsize = 8),
                row_title_gp = gpar(fontsize =8)
)

#dev.off()
```

Combined analysis

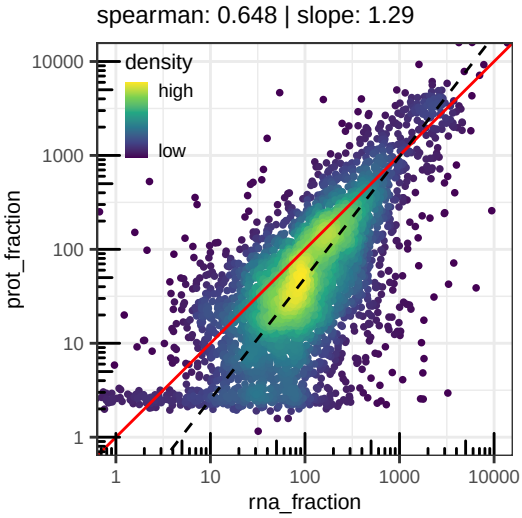
Across genes correlation

```
# convert protein ids to gene ids
use_genes <- intersect(rownames(linear_rna_normalised), rownames(linear_mass_spec_impute))

protein_data <- t(t(linear_mass_spec_impute[use_genes,])/(colSums(linear_mass_spec_impute[use_genes,])))
protein_data_log <- log2(protein_data)
rna_data <- t(t(linear_rna_normalised[use_genes,])/(colSums(linear_rna_normalised[use_genes,])))*1e6
rna_data_log <- log2(rna_data)

rna_data_means <- time_point_means(rna_data)
prot_data_means <- time_point_means(protein_data)


for (i in c("0h","2h","4h","8h","10h")) {
    plotdat <- data.frame("prot_fraction"=prot_data_means[,i], "rna_fraction"=rna_data_means[,i])
    model <- lmodel2(log10(prot_fraction) ~ log10(rna_fraction), plotdat, range.y = "interval", range.x
    print(model$regression.results[4,2])
    x <- densCols(log10(plotdat$rna_fraction), log10(plotdat$prot_fraction), colramp = colorRampPalette
    plotdat$dens <- col2rgb(x)[1,] / 256
    ggplot(plotdat[order(plotdat$dens),], aes(rna_fraction,prot_fraction, col=dens))+
        geom_point(shape = 16, alpha = 1, size = 1)+
        geom_abline(slope = 1, col="red")+
        geom_abline(slope = model$regression.results[4,3], intercept = model$regression.results[4,2], l
        scale_x_log10()+#limits = c(1e-6,1e-2))+
        scale_y_log10()+#limits = c(1e-6,1e-2))+
        scale_color_viridis_c("density", limits = c(0,1), breaks = c(0.1,0.9), labels = c("low","high")
        guides(color=guide_colorbar(ticks.colour = "NA"))+
        coord_cartesian(xlim = c(1,1e4),ylim = c(1,1e4))+
        ggtitle(paste0(
        "spearman: ", signif(cor.test(x = plotdat$prot_fraction, y = plotdat$rna_fraction, method = "sp
        " | slope: ", signif(model$regression.results[4,3], digits = 3)
        ))+
        annotation_logticks()+
        theme(legend.key.height = unit(2, "mm"), legend.position = c(0.15,0.85), legend.background = el
}
```

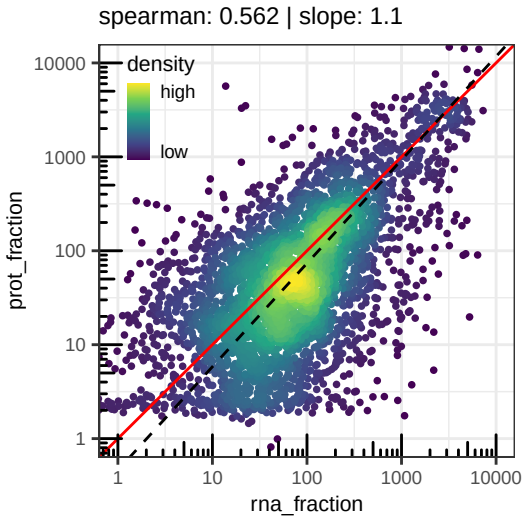
```
## [1] -0.8891229
```

```
## Warning in cor.test.default(x = plotdat$prot_fraction, y =
## plotdat$rna_fraction, : Cannot compute exact p-value with ties

## [1] -0.3357256

## Warning in cor.test.default(x = plotdat$prot_fraction, y =
## plotdat$rna_fraction, : Cannot compute exact p-value with ties

## [1] -0.5520787

## Warning in cor.test.default(x = plotdat$prot_fraction, y =
## plotdat$rna_fraction, : Cannot compute exact p-value with ties

## [1] -0.613554
## [1] -0.9243054

## Warning in cor.test.default(x = plotdat$prot_fraction, y =
## plotdat$rna_fraction, : Cannot compute exact p-value with ties
```

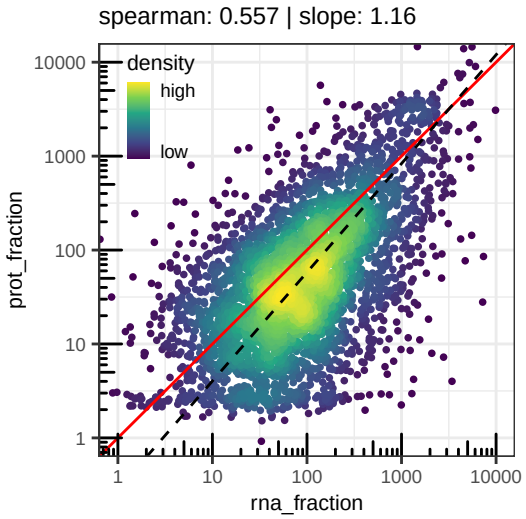
```
a[["0h"]]
```



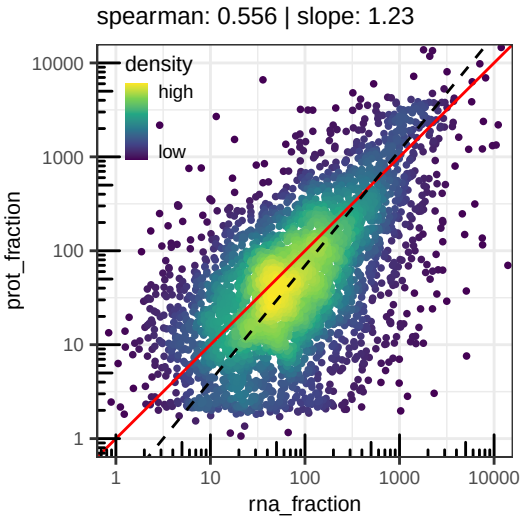

```
a[["2h"]]
```


spearman: 0.562 | slope: 1.1
density
high
low
prot_fraction
rna_fraction
10000
1000
100
10
1
1
10
100
1000
10000

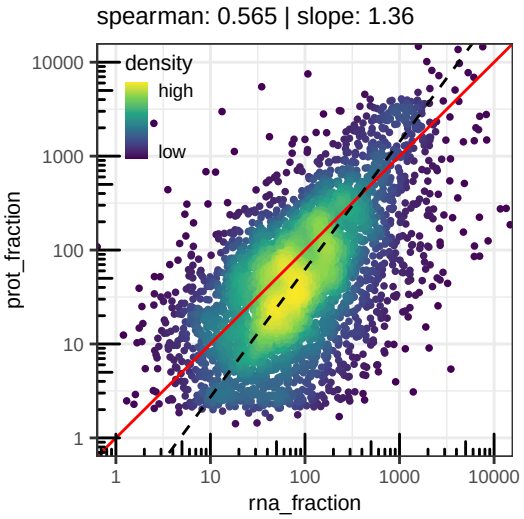
```
a[["4h"]]
```


spearman: 0.557 | slope: 1.16
density
high
low
prot_fraction
rna_fraction
10000
1000
100
10
1
1
10
100
1000
10000

```
a[["8h"]]
```




```
a[["10h"]]
```




```
ggsave("plots/correlation_across_genes.svg",a[["0h"]],"svg",width = 2.8, height = 2.8)
ggsave("plots/correlation_across_genes_2h.svg",a[["2h"]],"svg",width = 2.8, height = 2.8)
ggsave("plots/correlation_across_genes_4h.svg",a[["4h"]],"svg",width = 2.8, height = 2.8)
ggsave("plots/correlation_across_genes_8h.svg",a[["8h"]],"svg",width = 2.8, height = 2.8)
ggsave("plots/correlation_across_genes_10h.svg",a[["10h"]],"svg",width = 2.8, height = 2.8)



which.min(abs(cumsum(sort(prot_data_means[,1]))-0.1))
```

```
## Q55G70
##      1
```

```
which.min(abs(cumsum(sort(prot_data_means[,1], decreasing = T))-0.1))
```

```
## P07830
##      1
```

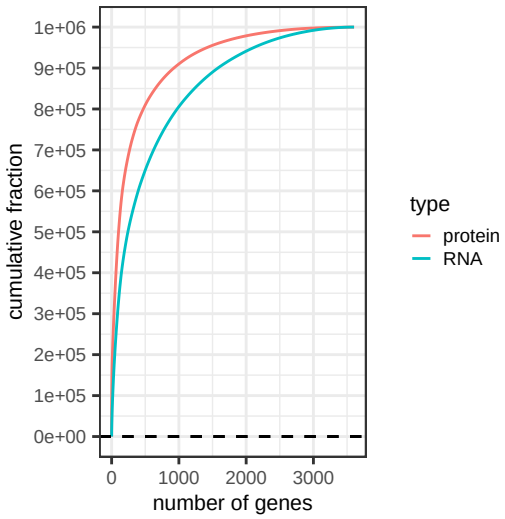
```
which.min(abs(cumsum(sort(rna_data_means[,1]))-0.1))
```

```
## Q54BV1
##      1
```

```
which.min(abs(cumsum(sort(rna_data_means[,1], decreasing = T))-0.1))
```

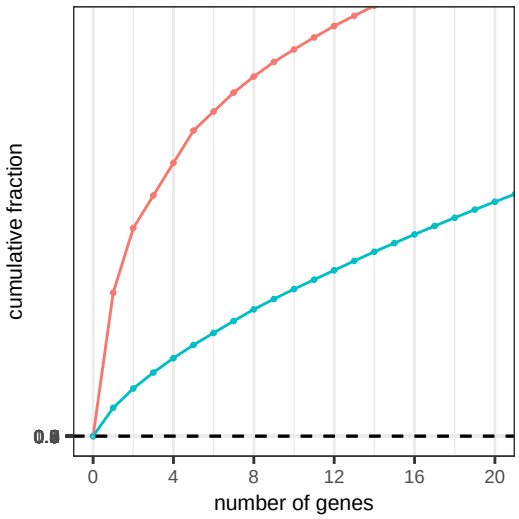
```
## P54657
##      1
```

```
plotdat <- rbind(
    data.frame("y"=c(0,cumsum(sort(prot_data_means[,1], decreasing = T))), "x"=0:nrow(prot_data_means),
    data.frame("y"=c(0,cumsum(sort(rna_data_means[,1], decreasing = T))), "x"=0:nrow(rna_data_means), "
)
ggplot(plotdat, aes(x, y, col = type))+
    #geom_point(shape = 1, size = 0.5)+
    geom_line()+
    geom_hline(yintercept = 0.1, lty="dashed")+
    geom_hline(yintercept = 0.9, lty="dashed")+
    scale_y_continuous(breaks = seq(0,1e6,1e5))+
    labs(x="number of genes", y ="cumulative fraction")->a
a
```




```
ggsave("plots/cumulative_fraction.svg",a, width = 3.4, height = 2.8)

ggplot(plotdat, aes(x, y, col = type))+
    geom_point(shape = 1, size = 0.5)+
```

```
    geom_line()+
    geom_hline(yintercept = 0.1, lty="dashed")+
    geom_hline(yintercept = 0.9, lty="dashed")+
    scale_y_continuous(breaks = seq(0,1,0.1))+
    scale_x_continuous(breaks = seq(0,20,4))+
    coord_cartesian(xlim = c(0,20), ylim = c(0,0.2e6))+
    labs(x="number of genes", y ="cumulative fraction")+
    theme(legend.position = "none")->b
b
```



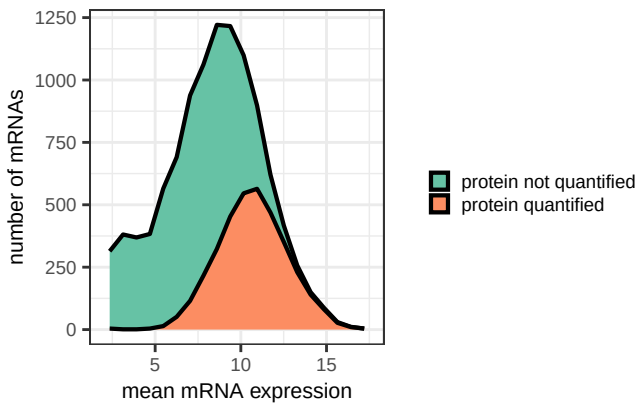

```
ggsave("plots/cumulative_fraction_zoom.svg",b, width = 1.4, height = 1.4)
```

Overall statistics

Expression ranges

```
mean_rna <- apply(assay(norm_rna_data),1,mean)

hist_data <- data.frame(mean_rna_exp=mean_rna,
                                                prot_exp=names(mean_rna) %in% rownames(mass_spec_impute
                                                prot_de=names(mean_rna) %in%  dev_prot)


hist_data$protein <- apply(hist_data,1,
                                                     function(x){
                                                        if(!x[2]){return("Not_expressed")}
                                                        if(x[2]){ return("Expressed")}
                                                     })

hist_data$protein <- factor(hist_data$protein, levels = c("Not_expressed","Expressed"))

ggplot(hist_data, aes(x=mean_rna_exp, fill=protein)) +
    geom_area(aes(y = ..count..), stat = "bin", bins = 20, col = "black", lwd = 0.75)+
    scale_fill_manual(values = brewer.pal(4, "Set2"), name = "", labels = c("protein not quantified","p
```

```
    labs(x="mean mRNA expression", y="number of mRNAs")->a
a
```

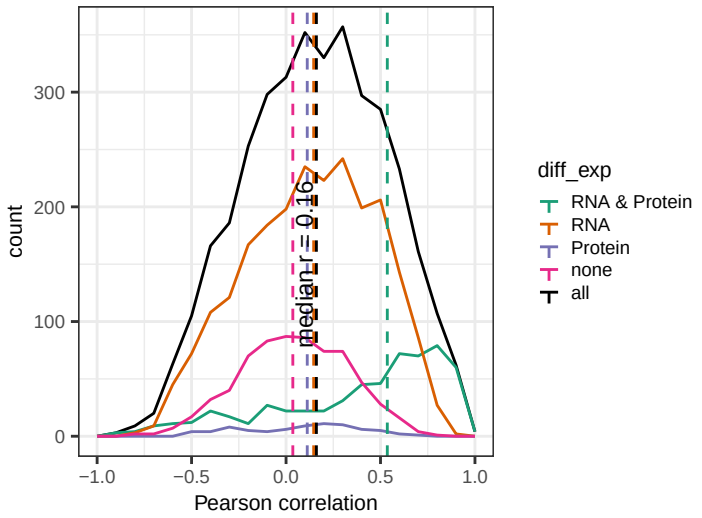



```
ggsave("plots/protein_quantified_rna_expression.svg",a,"svg",width = 3.5, height = 2.2)
```

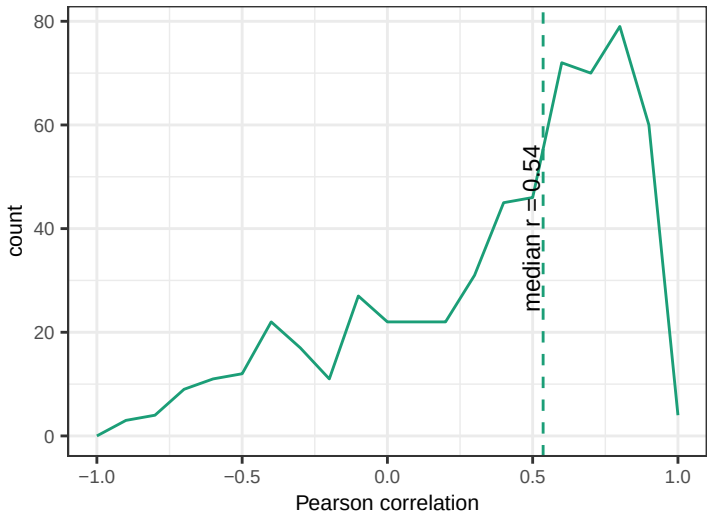
Per gene correlation

```
common_samples <- intersect(colnames(rna_data),colnames(protein_data))

cors <- sapply(use_genes, function(x){cor(protein_data_log[x,common_samples], rna_data_log[x,common_samp
pvals <- sapply(use_genes, function(x){cor.test(protein_data_log[x,common_samples], rna_data_log[x,comm

hist_data <- data.frame(correlation=cors,
                                                de_prot=names(cors) %in% dev_prot,
                                                de_rna=names(cors) %in% dev_genes,
                                                p_value=pvals)

hist_data$diff_exp <- apply(hist_data,1,
                                                        function(x){
                                                            if(all(x[2] & x[3])){return("RNA & Protein"
                                                            if(all(x[2] & !x[3])){return("Protein")}
                                                            if(all(!x[2]& x[3])){return("RNA")}
                                                            if(all(!x[2]& !x[3])){return("none")}
                                                        })
hist_data$diff_exp <- factor(hist_data$diff_exp, levels = c("RNA & Protein","RNA","Protein","none"))

median_data <- hist_data %>%
    group_by(diff_exp) %>%
    summarise( median_val = median(correlation, na.rm=T))

median_data <- rbind(median_data, hist_data %>%
                                            summarise( diff_exp = "all", median_val = median(correlatio


ggplot(hist_data, aes(x=correlation)) +
    geom_freqpoly(binwidth = 0.1) +
```

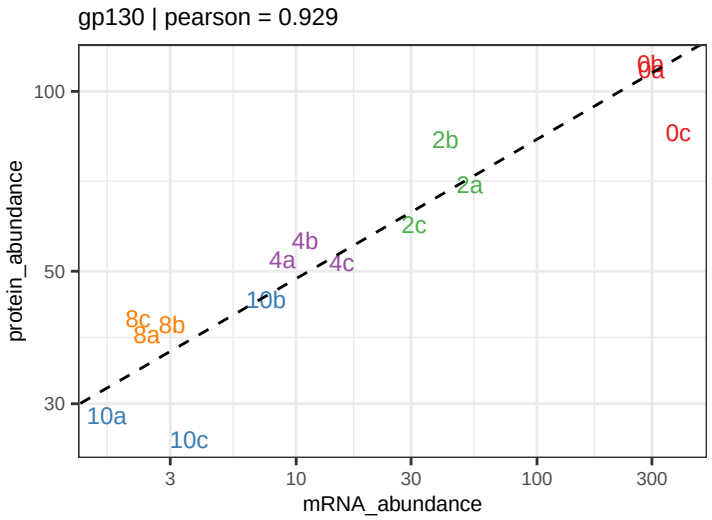
```
    geom_freqpoly(aes(col= diff_exp), binwidth = 0.1)+
    xlim(-1, 1) +
    geom_vline(data=median_data, aes(xintercept = median_val, col = diff_exp), linetype = "dashed" ) +
    geom_text(data=median_data[5,], aes(label = paste0("median r = ",round(median_val,digits = 2)), x =
    scale_color_manual(values = c(brewer.pal(4, "Dark2"),"black"))+
    labs(y = "count",x = "Pearson correlation")->a
a
```



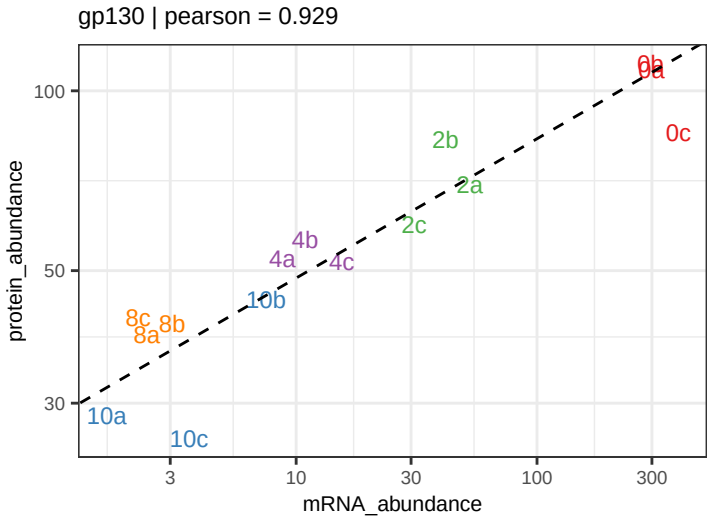

```
ggsave("plots/correlation_per_gene.svg",a,"svg",width = 3.8,height = 2.8)


ggplot(hist_data[hist_data$diff_exp=="RNA & Protein",], aes(x=correlation)) +
    geom_freqpoly(binwidth = 0.1, col=brewer.pal(4, "Dark2")[1]) +
    xlim(-1, 1) +
    geom_vline(data=median_data[1,], aes(xintercept = median_val), linetype = "dashed", col=brewer.pal(
    geom_text(data=median_data[1,], aes(label = paste0("median r = ",round(median_val,digits = 2)), x =
    labs(y = "count",x = "Pearson correlation")->a
a
```



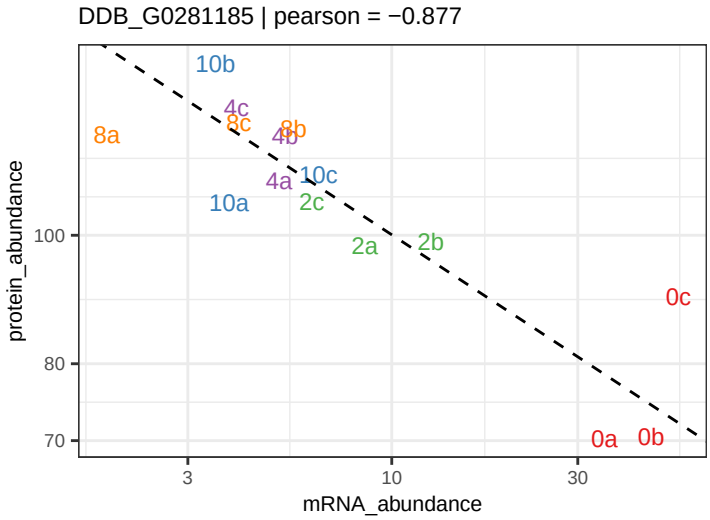
```
ggsave("plots/correlation_per_gene_DEonly.svg",a,"svg",width = 2.3,height = 2.3)


plot_ids <- c(names(which.max(na.omit(cors[names(cors)%in%intersect(intersect(dev_prot, dev_genes), non
                              names(whichmedian(na.omit(cors[names(cors)%in%intersect(intersect(dev_prot,
                              names(which.min(na.omit(cors[names(cors)%in%intersect(intersect(dev_prot, d
)
plot_id <- plot_ids[1]
scatter_corr <- cbind.data.frame("mRNA_abundance"=rna_data[plot_id,common_samples],
                                                                    "protein_abundance"=protein_data[plot_
                                                                    "time"=factor(paste0(rep(c(0,2,4,8,10)
                                                                    "sample"=factor(paste0(rep(c(0,2,4,8,1
model <- lmodel2(log10(protein_abundance) ~ log10(mRNA_abundance), scatter_corr, range.y = "interval", 
print(ggplot(scatter_corr, aes(mRNA_abundance,protein_abundance, label = sample, color = time))+
                geom_text(size = 3)+
                scale_color_brewer(palette = "Set1")+
                scale_x_log10()+
                scale_y_log10()+
                #geom_smooth(color = "black",linetype = "dashed", alpha = 0.5, method = lm, se = FALSE)
                geom_abline(slope = model$regression.results[4,3], intercept = model$regression.results
                labs(title = paste0(uniprot_to_gene_name[plot_id]," | pearson = ", round(cors[plot_id],
                theme(legend.position = "none"))->a
```



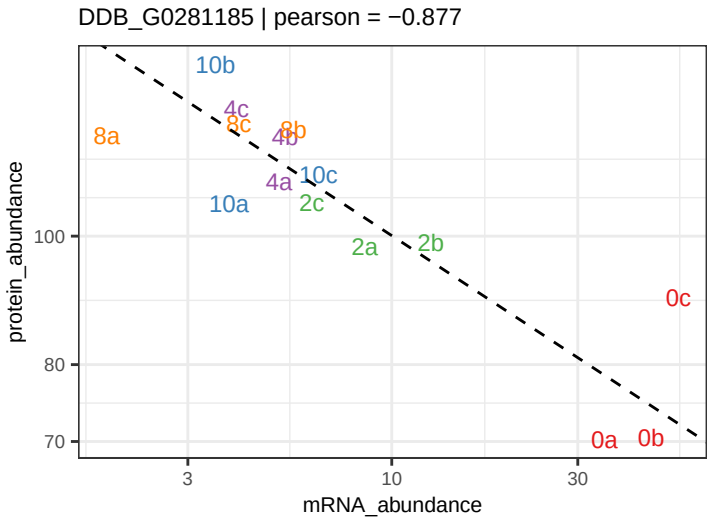

```
a
```



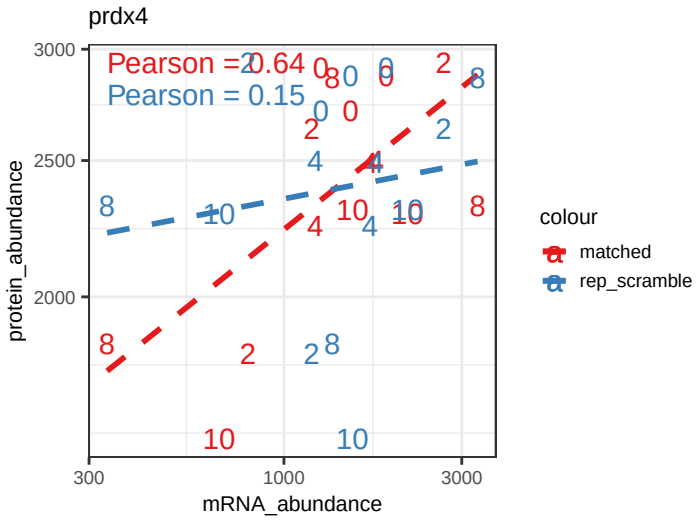

```
ggsave("plots/high_gene_correlation.svg",a,"svg",width = 2.3, height = 2.3)

plot_id <- plot_ids[3]
scatter_corr <- cbind.data.frame("mRNA_abundance"=rna_data[plot_id,common_samples],
                                                                    "protein_abundance"=protein_data[plot_
                                                                    "time"=factor(paste0(rep(c(0,2,4,8,10)
                                                                    "sample"=factor(paste0(rep(c(0,2,4,8,1
model <- lmodel2(log10(protein_abundance) ~ log10(mRNA_abundance), scatter_corr, range.y = "interval", 
print(ggplot(scatter_corr, aes(mRNA_abundance,protein_abundance, label = sample, color = time))+
                geom_text(size = 3)+
                scale_color_brewer(palette = "Set1")+
                scale_x_log10()+
                scale_y_log10()+
                #geom_smooth(color = "black",linetype = "dashed", alpha = 0.5, method = lm, se = FALSE)
                geom_abline(slope = model$regression.results[4,3], intercept = model$regression.results
                labs(title = paste0(uniprot_to_gene_name[plot_id]," | pearson = ", round(cors[plot_id],
                theme(legend.position = "none"))->a
```



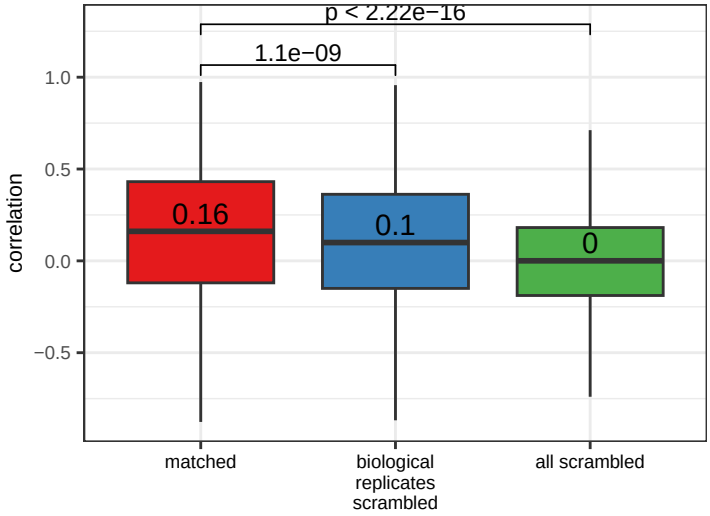
```
a
```




```
ggsave("plots/low_gene_correlation.svg",a,"svg",width = 2.3, height = 2.3)

cors <- sapply(use_genes, function(x){cor(protein_data_log[x,common_samples], rna_data_log[x,common_samp
cors_rep_scramble <- sapply(use_genes, function(x){cor(protein_data_log[x,common_samples], rna_data_log
cors_rep_scramble2 <- sapply(use_genes, function(x){cor(protein_data_log[x,common_samples], rna_data_log
cors_all_scramble <- sapply(use_genes, function(x){cor(protein_data_log[x,common_samples], rna_data_log

plot_id <- "Q555L5"
scatter_corr <- cbind.data.frame("mRNA_abundance"=rna_data[plot_id,common_samples],
                                                                    "protein_abundance"=protein_data[plot_
                                                                    "sample"=paste0(rep(c(0,2,4,8,10),each
ggplot(scatter_corr, aes(mRNA_abundance,protein_abundance, label = sample, col = "matched"))+
    geom_text()+
    scale_x_log10()+
    scale_y_log10()+
    geom_smooth(linetype = "dashed", alpha = 0.5, method = lm, se = FALSE)+
    geom_text(data = scatter_corr[1,],aes(x = min(scatter_corr$mRNA_abundance) , y = max(scatter_corr$p
    geom_text(aes(mRNA_abundance[c(2,3,1,5,6,4,8,9,7,11,12,10,14,15,13)],protein_abundance, col = "rep_
    geom_smooth(aes(mRNA_abundance[c(2,3,1,5,6,4,8,9,7,11,12,10,14,15,13)],protein_abundance, col = "re
    geom_text(data = scatter_corr[1,],aes(x = min(scatter_corr$mRNA_abundance) , y = max(scatter_corr$p
    labs(title = paste0(uniprot_to_gene_name[plot_id]))+
    scale_color_brewer(palette = "Set1")->a
a
```



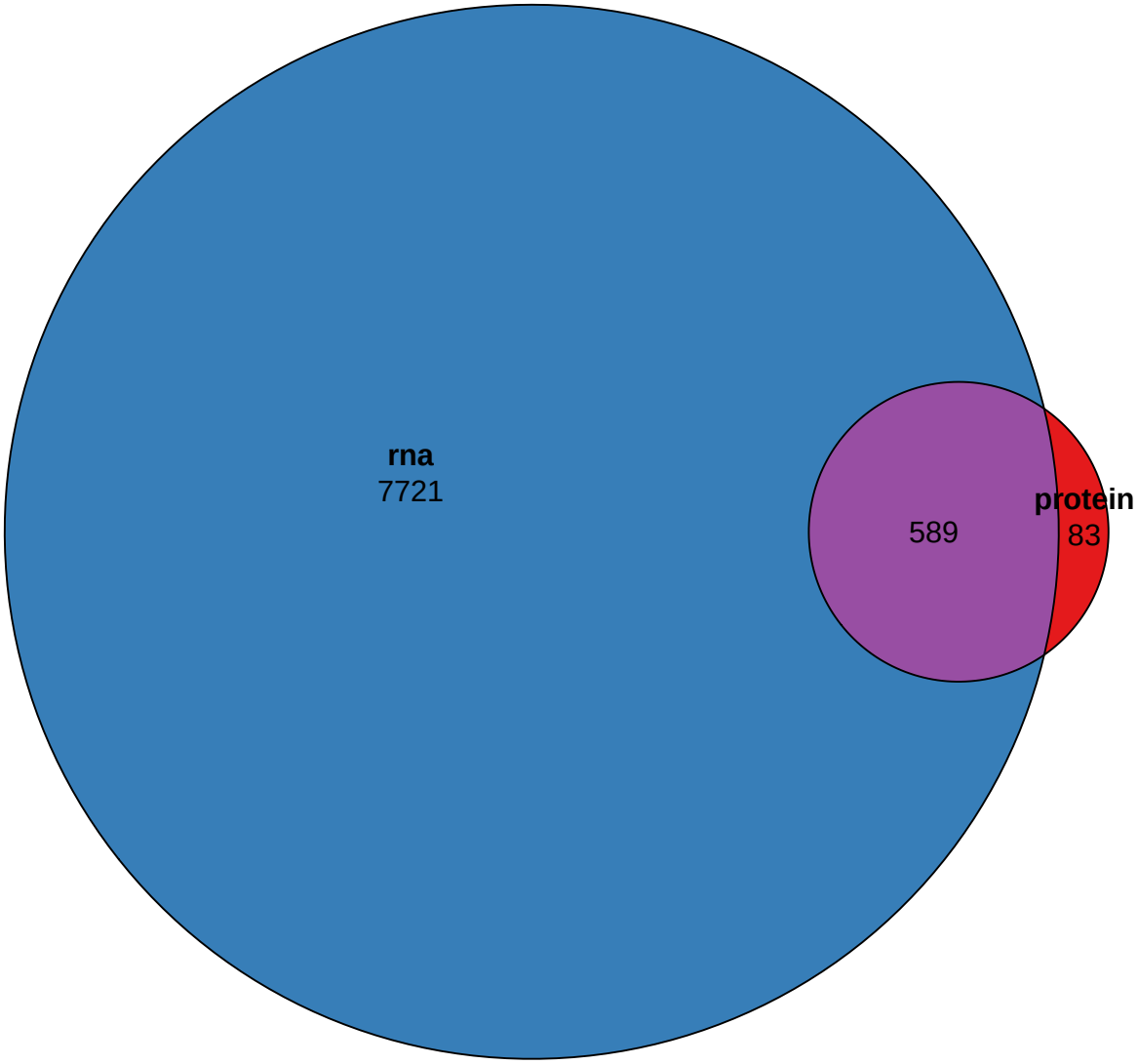

```
ggsave("plots/gene_correlation_scrambled.svg",a,"svg",width = 3.8,height = 2.8)

box_scramble <- na.omit(rbind(cbind.data.frame(cors = cors,scramble = "matched"),cbind.data.frame(cors =
box_scramble$scramble <- factor(box_scramble$scramble, levels = c("matched","rep_scramble","all_scrambl
my_comparisons <- list(c("matched","rep_scramble"),c("matched","all_scramble"))
medians <- box_scramble %>% group_by(scramble) %>% summarise(median = median(cors))
ggplot(box_scramble,aes(x =  scramble, y = cors, fill = scramble))+
    geom_boxplot(outlier.shape = NA)+
    stat_compare_means(comparisons = my_comparisons, size = 3, method = "t.test")+
    scale_x_discrete(labels = c("matched","biological\nreplicates\nscrambled","all scrambled"))+
    labs(y = "correlation")+
    geom_text(data = medians, mapping = aes(x = scramble, y = median+.1, label = round(median,2)))+
    scale_fill_brewer(palette = "Set1")+
    theme(legend.position = "none", axis.title.x = element_blank(), axis.text.x = element_text(colour =
a
```



```
ggsave("plots/gene_correlation_boxplot_scramble.svg",a,"svg",width = 2.8,height = 2.8)

out_comb_table <- cbind.data.frame(
    gene_info[match(use_genes,gene_info$UniProt_ID),],
    cors[use_genes], pvals[use_genes],
    rna_data[use_genes,common_samples], protein_data[use_genes,]
)

write.table(out_comb_table, "tables/comb_corr.tsv", sep = "\t", row.names = F, col.names = T, quote = F
```

Heatmaps

```
both_de <- cbind(
    LFCrna[dev_genes[dev_genes %in% dev_prot],],
    LFCprot[dev_genes[dev_genes %in% dev_prot],]
)
rna_de_only <- LFCrna[dev_genes[!(dev_genes %in% dev_prot)],]
prot_de_only <- LFCprot[dev_prot[!(dev_prot %in% dev_genes)],]

both_clustered <- rownames(both_de)[hclust(dist(both_de),method = clust_method)$order]
rna_only_clustered <- rownames(rna_de_only)[hclust(dist(rna_de_only),method = clust_method)$order]
prot_only_clustered <- rownames(prot_de_only)[hclust(dist(prot_de_only),method = clust_method)$order]

venn_rna_prot_colors <- venn_colors
names(venn_rna_prot_colors) <- c("rna","protein","both")
venn_data <- c(length(rna_only_clustered),
                            length(prot_only_clustered),
                            length(both_clustered))
names(venn_data) <- c(names(venn_rna_prot_colors[1:2]),paste0(names(venn_rna_prot_colors[1]),"&",names(v
plot(euler(venn_data),quantities = TRUE,fills=venn_rna_prot_colors)
```




```
plot_data <- as.matrix(cbind(LFCrna[c(both_clustered),],
                                                    LFCprot[c(both_clustered),]))

row_annot <- data.frame("DE"=c(rep(names(venn_rna_prot_colors)[3],length(both_clustered))
))




nrow(plot_data)
```

```
## [1] 589
```

```
#svglite("plots/mRNA_protein_heatmap_DE.svg", width = 3.8, height = 3)
Heatmap(plot_data,
                cluster_columns = FALSE,
                cluster_rows = T,use_raster = T,
```

```
                show_row_names = FALSE,
                col = heatmap_color_scale,
                column_names_side = c("top"),
                column_names_rot = 0,   column_names_centered = T, column_names_gp = gpar(fontsize = 10)
                heatmap_legend_param = list(title = "logFC",
                                                                    labels_gp = gpar(fontsize = 6),
                                                                    title_gp = gpar(fontsize = 6, fo
                                                                    grid_width = unit(3,"mm")),
                column_split = factor(c(rep("Transcriptomics",5),rep("Proteomics",4)), levels = c("Trans
                #cluster_column_slices = FALSE,
                #column_gap = unit(c(rep(0,10),2,rep(0,5)), "mm"),
                column_title_gp = gpar(fontsize = 10, fontface="bold"),
                row_title_gp = gpar(fontsize =8)
)
#dev.off()


plot_data <- na.omit(as.matrix(cbind(LFCrna[union(dev_genes,dev_prot),],
                                                                    LFCprot[union(dev_genes,dev_pro

row_annot <- data.frame(DE = rep("both", nrow(plot_data)))
rownames(row_annot) <- rownames(plot_data)
row_annot[intersect(rownames(plot_data),rna_only_clustered),] <- "rna"
row_annot[intersect(rownames(plot_data),prot_only_clustered),] <- "protein"

nrow(plot_data)
```

```
## [1] 2934
```

```
#svglite("plots/mRNA_protein_heatmap_all.svg", width = 6.8, height = 3)
Heatmap(plot_data,
                cluster_columns = FALSE,
                cluster_rows = T,use_raster = T,
                show_row_names = FALSE,
                col = heatmap_color_scale,
                column_names_side = c("top"),
                column_names_rot = 0,   column_names_centered = T, column_names_gp = gpar(fontsize = 10)
                heatmap_legend_param = list(title = "logFC",
                                                                    labels_gp = gpar(fontsize = 8),
                                                                    title_gp = gpar(fontsize = 8, fo
                                                                    grid_width = unit(3,"mm")),
                column_split = factor(c(rep("Transcriptomics",5),rep("Proteomics",4)), levels = c("Trans
                #cluster_column_slices = FALSE,
                #column_gap = unit(c(rep(0,10),2,rep(0,5)), "mm"),
                column_title_gp = gpar(fontsize = 10, fontface="bold"),
                row_title_gp = gpar(fontsize =8)
)
#dev.off()


rna_prot_table <- cbind.data.frame(
    gene_info[match(rownames(both_de),gene_info$UniProt_ID),],
    both_de,
```

```
    res$padj[match(rownames(both_de),rownames(res))],
    tt_dev$adj.P.Val[match(rownames(both_de),rownames(tt_dev))]
)

rna_prot_table <- rna_prot_table[both_clustered,]

head(rna_prot_table, n = 10)
```

```
##             GENE_ID    Gene_Name                                  Synonyms
## Q54H99 DDB_G0289605         fmoC                                      <NA>
## Q75JG1 DDB_G0276125 DDB_G0276125                                      <NA>
## Q54T55 DDB_G0281993 DDB_G0281993                                      <NA>
## Q552D2 DDB_G0276215 DDB_G0276215                                      <NA>
## Q558S7 DDB_G0273017 DDB_G0273017                                      <NA>
## Q54J14 DDB_G0288377         gxcH                                    RacGEF
## Q54PS9 DDB_G0284353         osbI                                      <NA>
## Q54SZ9 DDB_G0282107 DDB_G0282107                                      <NA>
## Q55C57 DDB_G0270218         glkA glycogen synthase kinase-like kinase A
## Q54GT4 DDB_G0289925         fmoG                                       FMO
##                                                                                                Gene_produ
## Q54H99                  flavin-containing monooxygenase, dimethylaniline monooxygenase [N-oxide-formi
## Q75JG1                                                                                                 -
## Q54T55                                                                                                 -
## Q552D2                                                                  glycoside hydrolase family 15 prot
## Q558S7                                                                                 isocitrate ly
## Q54J14              pleckstrin homology (PH) domain-containing protein, RhoGEF domain-containing prot
## Q54PS9                                                        oxysterol binding family protein, membe
## Q54SZ9                                                           putative phospholipid transfer prot
## Q55C57                                                    putative serine/threonine-protein kinase (
## Q54GT4 flavin-containing monooxygenase, N,N-dimethylaniline,NADPH:oxygen oxidoreductase, N-oxide-form
##        UniProt_ID        2vs0       4vs0     6vs0     8vs0    10vs0        2vs0
## Q54H99     Q54H99 -0.06089907  1.9664678 4.056785 6.781484 7.043163 -0.02643865
## Q75JG1     Q75JG1 -0.62803958  2.0207838 2.134841 4.822079 7.221099  0.16754061
## Q54T55     Q54T55  0.16762826  0.0667313 2.561283 6.774538 6.999453  0.07122094
## Q552D2     Q552D2  0.97134185  1.5647933 3.249114 6.153541 8.030254 -0.22103626
## Q558S7     Q558S7 -0.51918055 -0.3756603 1.216442 4.880374 6.567020  0.39840132
## Q54J14     Q54J14 -0.15966877  0.5114775 1.966289 4.783156 4.932081  0.21619460
## Q54PS9     Q54PS9 -0.64911695  1.1768052 2.080190 3.637187 4.150633 -0.24715606
## Q54SZ9     Q54SZ9  0.60304310  1.4355737 4.039211 6.969568 6.163505 -0.21727051
## Q55C57     Q55C57 -0.52887626  0.6314144 4.833352 6.340174 6.160718  0.16437778
## Q54GT4     Q54GT4  0.96490421  2.1510873 3.825829 4.832011 4.618946 -0.20666185
##                4vs0       8vs0     10vs0
## Q54H99 -0.026035834 -0.2458041 -0.730418
## Q75JG1  0.176596485 -0.2095826  2.307729
## Q54T55  0.358136205  0.8909120  2.127893
## Q552D2 -0.003044373 -0.4997853  1.421011
## Q558S7 -0.062155141  0.8947809  3.545791
## Q54J14 -0.081835953  0.0583158  1.484529
## Q54PS9 -0.233017775  0.9730920  2.217895
## Q54SZ9 -0.129300346  1.9189969  4.177482
## Q55C57  0.136397916  2.4070880  3.033617
## Q54GT4 -0.605947010  0.8209957  1.603787
##        res$padj[match(rownames(both_de), rownames(res))]
```

```
## Q54H99                                          6.949573e-55
## Q75JG1                                          4.665309e-61
## Q54T55                                         1.875905e-172
## Q552D2                                          2.141643e-93
## Q558S7                                         1.961059e-268
## Q54J14                                          1.564945e-47
## Q54PS9                                          5.718323e-41
## Q54SZ9                                          1.270432e-95
## Q55C57                                          3.512620e-58
## Q54GT4                                          6.965500e-52
##        tt_dev$adj.P.Val[match(rownames(both_de), rownames(tt_dev))]
## Q54H99                                                4.509230e-03
## Q75JG1                                                8.995916e-03
## Q54T55                                                9.813782e-03
## Q552D2                                                7.700114e-03
## Q558S7                                                2.698449e-03
## Q54J14                                                8.279620e-03
## Q54PS9                                                4.641163e-03
## Q54SZ9                                                6.872628e-03
## Q55C57                                                9.388251e-05
## Q54GT4                                                9.949970e-03
```

```
write.table(rna_prot_table, file = "tables/rna_prot_table.tsv", sep = "\t", quote = F, row.names = T, c


loomis_devgenes <- read.table("data/loomis_2015_devgenes.txt", sep = "\t", header = FALSE)


plot_data <- cbind(sc_de_genes[common_expressed,],
                               scale_rows(linear_mass_spec_impute[common_expressed,]))
plot_data <- plot_data[loomis_devgenes$V1[loomis_devgenes$V1%in%rownames(plot_data)
                                                                                #&loomis_devgenes$V
                                                                                ],]



row_annot <- data.frame(stage = loomis_devgenes$V2[loomis_devgenes$V1%in%rownames(plot_data)])
row_annot$stage <- factor(row_annot$stage, levels = c("agg","slug","cul"))

rownames(plot_data) <- uniprot_to_gene_name[rownames(plot_data)]

#svglite("plots/loomis_devgenes.svg", width = 6, height = 5)
Heatmap(plot_data,
                name = "zscore",
                cluster_columns = FALSE,
                cluster_rows = T,
                show_row_names = TRUE,
                use_raster = T,
                col = heatmap_color_scale, show_column_names = F,
                row_split = factor(row_annot$stage, levels = c("cul","slug","agg")),
                column_split = factor(c(rep("Transcriptomics",24),rep("Proteomics",15)), levels = c("Tra
                top_annotation = columnAnnotation(
                    time =  c(rep(seq(0,10,2),each = 4), rep(c(0,2,4,8,10),each = 3)),
                    col = list(time = colorRamp2(c(0,5,10),c("#EAF6DF","#b2df8a","#33a02c"))),
                    height = unit(1, "mm"),
```

```
                simple_anno_size_adjust = TRUE,
                    annotation_legend_param = list(time = list(title = "time(h)",
                                                               show_annotation_name = F),
                heatmap_legend_param = list(title = "zscore",
                                                                    labels_gp = gpar(fontsize = 8),
                                                                    title_gp = gpar(fontsize = 8, fo
                                                                    grid_width = unit(3,"mm")),
                column_title_gp = gpar(fontsize = 10, fontface="bold"),
                row_names_gp = gpar(fontsize = 8)
                )
#dev.off()
```

Missing Proteins

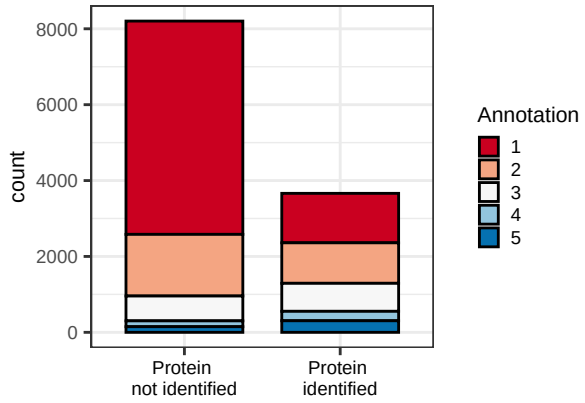
```
uniprot_data <- read.table("data/uniprot_length_annotation.tsv", sep = "\t", header = T, row.names = 1)
all_prots <- na.omit(unique(gene_id_to_uniprot))

combined_means <- cbind.data.frame(
    "mean_rna" = apply(counts, 1, mean)[all_prots],
    "mean_prot" = apply(linear_mass_spec_impute, 1, mean)[all_prots]
)
rownames(combined_means) <- all_prots
combined_means <- combined_means %>% mutate(expressed = case_when(
    !is.na(mean_rna)&!is.na(mean_prot)&mean_rna>1 ~ "both",
    !is.na(mean_rna)&mean_rna>1 ~ "rna",
    !is.na(mean_prot) ~ "protein",
    .default = "not_expressed")
)
table(combined_means$expressed)
```

```
##
##          both not_expressed       protein           rna
##          3600          1371            63          6832
```

```
combined_means <- cbind.data.frame(combined_means, uniprot_data[all_prots,])
combined_means$mean_rna[is.na(combined_means$mean_rna)] <- min(na.omit(combined_means$mean_rna))/2
combined_means$mean_prot[is.na(combined_means$mean_prot)] <- min(na.omit(combined_means$mean_prot))/2

combined_means$prot_identified <- combined_means$expressed=="both"|combined_means$expressed=="protein"
combined_means$Annotation <- factor(combined_means$Annotation, levels = 1:5)
combined_means$expressed <- factor(combined_means$expressed, levels = c("both","protein","rna","not_expr

ggplot(combined_means, aes(x= prot_identified, fill=Annotation))+geom_histogram(stat="count", color = "
```

```
## Warning in geom_histogram(stat = "count", color = "black", width = 0.75):
## Ignoring unknown parameters: `binwidth`, `bins`, and `pad`
```

```
a
```

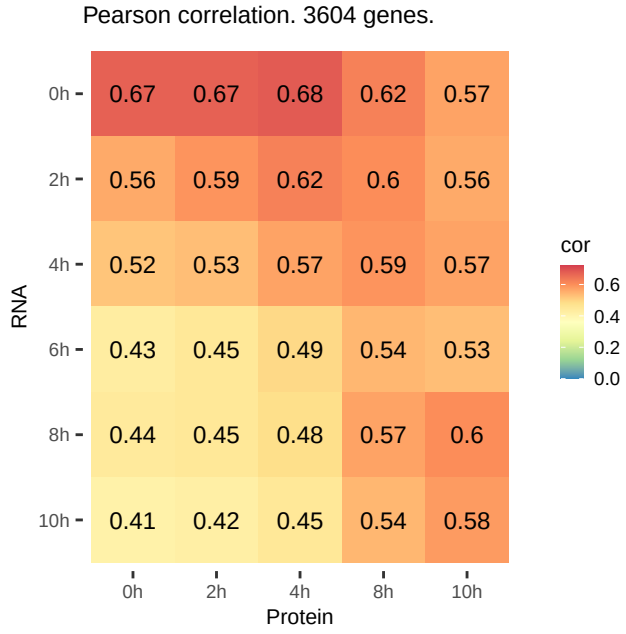



```
ggsave("plots/prot_identification_annotation.svg",a,"svg",width = 3.2,height = 2.2)
```

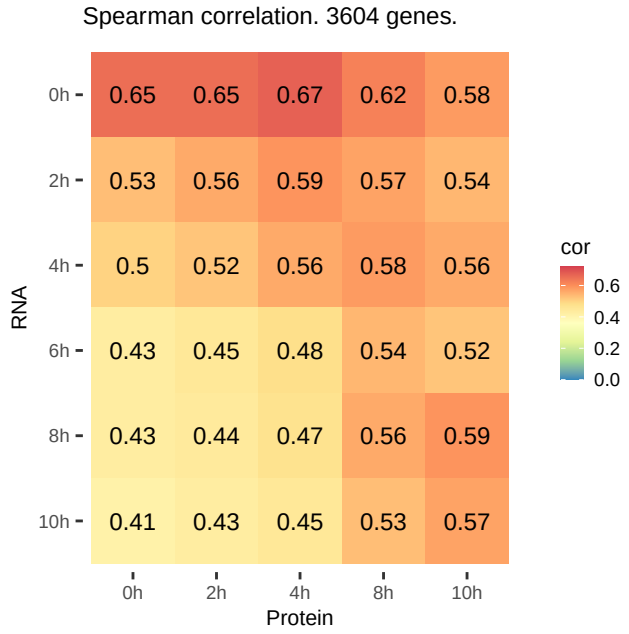
Time-lag

All genes

```
scale_color <- c(0,0.72)
plot_time_lag_cor(log2(rna_data_means), log2(prot_data_means), cor_method = "pearson", scale_color = sca
```



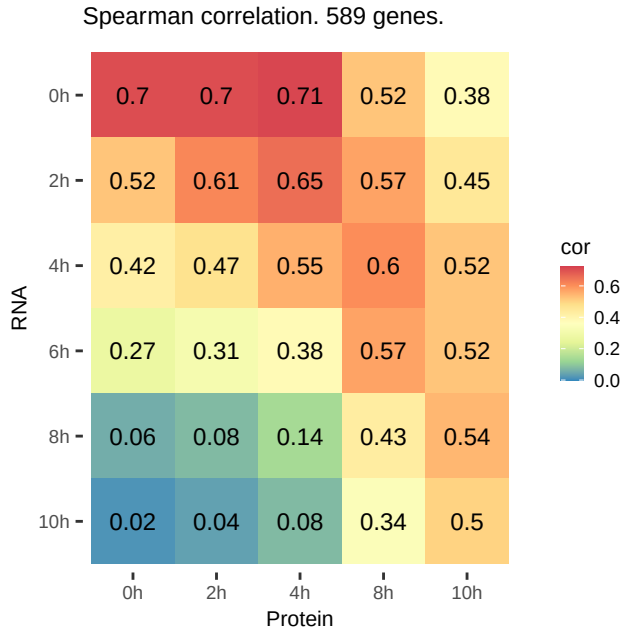

```
plot_time_lag_cor(log2(rna_data_means), log2(prot_data_means), cor_method = "spearman", scale_color = sc
a
```



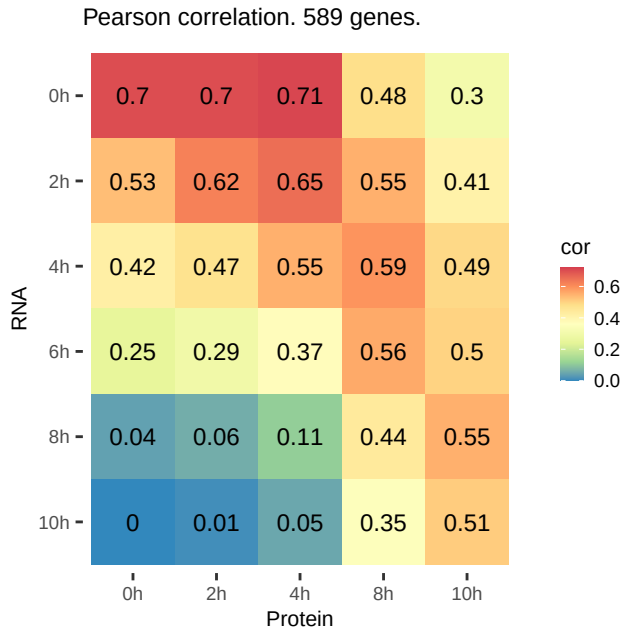

```
ggsave("plots/spearman_corrmatrix_all.svg",a,"svg",width = 3.4,height = 3.4)
```

Differentially expressed proteins+genes

```
de_genes_all <- intersect(dev_prot, dev_genes)
plot_time_lag_cor(log10(rna_data_means), log10(prot_data_means), use_genes=de_genes_all, scale_color = s
a
```

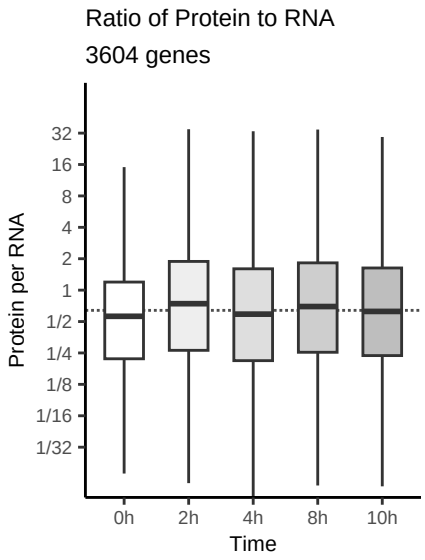


```
ggsave("plots/spearman_corrmatrix_DE.svg",a,"svg",width = 3.4,height = 3.4)
plot_time_lag_cor(log10(rna_data_means), log10(prot_data_means), use_genes=de_genes_all, cor_method = "
```



Ratios of RNA to protein

```
intersect(
    rownames(na.omit(remove_outliers(log2(linear_mass_spec_impute), 8))),
    rownames(na.omit(remove_outliers(log2(linear_rna_normalised), 8)))
) -> use_genes

fractions_df <- data.frame(
    prot_data_means,rna_data_means[,-4]
)
fractions <- data.frame(fractions_df[,1:5]/fractions_df[,6:10])
fractions$geneID <- rownames(fractions)
plotdat <- fractions %>% pivot_longer(!geneID, names_to = "timepoint", values_to = "prot_per_rna")
plotdat$timepoint <- factor(substr(plotdat$timepoint,2,4), levels = c("0h","2h","4h","8h","10h"))
ggplot(plotdat, aes(x = timepoint, y= prot_per_rna, fill=timepoint))+
    geom_hline(yintercept = median(plotdat$prot_per_rna), color = "grey30", lty = "11")+
    geom_boxplot(outlier.shape = NA, width = 0.6)+
    ggtitle("Ratio of Protein to RNA" ,paste0(length(unique(plotdat$geneID)), " genes"))+
    scale_y_continuous(trans="log2",
                                        breaks = c(1/32,1/16,1/8,1/4,1/2,1,2,4,8,16,32),
                                        labels = c("1/32","1/16","1/8","1/4","1/2","1","2","4","8","16
    scale_fill_manual(values = colorRampPalette(c("white","grey"))(5))+
    coord_cartesian(ylim = c(1/64,64))+
    labs(y = "Protein per RNA", x = "Time")+
    theme(legend.position = "none", panel.grid = element_blank(), panel.border = element_blank(), axis.l
a
```




```
ggsave("plots/prot_per_rna_all.svg",a,"svg",width = 2.3, height = 3)
oneway.test(prot_per_rna ~ timepoint, data = plotdat, var.equal = T)
```

```
##
##  One-way analysis of means
##
## data:  prot_per_rna and timepoint
## F = 0.45287, num df = 4, denom df = 18015, p-value = 0.7704
```

```
pairwise.t.test(plotdat$prot_per_rna, plotdat$timepoint, p.adjust.method = "none")
```

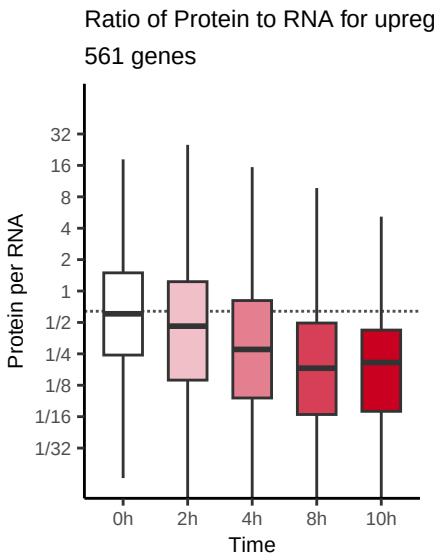
```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  plotdat$prot_per_rna and plotdat$timepoint
##
##     0h   2h   4h   8h
## 2h  0.23 -    -    -
## 4h  0.58 0.52 -    -
## 8h  0.28 0.89 0.61 -
## 10h 0.48 0.62 0.88 0.71
##
## P value adjustment method: none
```

```
summary(glht(aov(prot_per_rna ~ timepoint, data = plotdat), linfct = mcp(timepoint = "Dunnett")))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
```

```
## Fit: aov(formula = prot_per_rna ~ timepoint, data = plotdat)
##
## Linear Hypotheses:
##               Estimate Std. Error t value Pr(>|t|)
## 2h - 0h == 0    1.6224     1.3453   1.206    0.563
## 4h - 0h == 0    0.7500     1.3453   0.557    0.949
## 8h - 0h == 0    1.4447     1.3453   1.074    0.658
## 10h - 0h == 0   0.9491     1.3453   0.705    0.890
## (Adjusted p values reported -- single-step method)
```

```
fc_cutoff <- 2
up_genes0_10 <- rownames(rna_data_means)[apply(rna_data_means[,c("10h", "0h")],1,function(x){abs(x[1]/x
down_genes0_10 <- rownames(rna_data_means)[apply(rna_data_means[,c("10h", "0h")],1,function(x){abs(x[2]/

up_genes0_10 <- intersect(up_genes0_10, dev_genes)
down_genes0_10 <- intersect(down_genes0_10, dev_genes)
upplotdat <- plotdat[plotdat$geneID%in%up_genes0_10,]
downplotdat <- plotdat[plotdat$geneID%in%down_genes0_10,]
ggplot(upplotdat, aes(x = timepoint, y= prot_per_rna, fill=timepoint))+
    geom_hline(yintercept = median(plotdat$prot_per_rna), color = "grey30", lty = "11")+
    geom_boxplot(outlier.shape = NA, width = 0.6, notch = F)+
    ggtitle("Ratio of Protein to RNA for upregulated genes" ,paste0(length(unique(plotdat[plotdat$geneI
    scale_y_continuous(trans="log2",
                                        breaks = c(1/32,1/16,1/8,1/4,1/2,1,2,4,8,16,32),
                                        labels = c("1/32","1/16","1/8","1/4","1/2","1","2","4","8","16
    scale_fill_manual(values = colorRampPalette(c("white","#ca0020"))(5))+
    coord_cartesian(ylim = c(1/64,64))+
    labs(y = "Protein per RNA", x = "Time")+
    theme(legend.position = "none", panel.grid = element_blank(), panel.border = element_blank(), axis.l
a
```




```
ggsave("plots/prot_per_rna_up.svg",a,"svg",width = 2.3, height = 3)
oneway.test(prot_per_rna ~ timepoint, data = upplotdat, var.equal = T)
```

```
##
```

```
##  One-way analysis of means
##
## data:  prot_per_rna and timepoint
## F = 3.2595, num df = 4, denom df = 2800, p-value = 0.01122
```

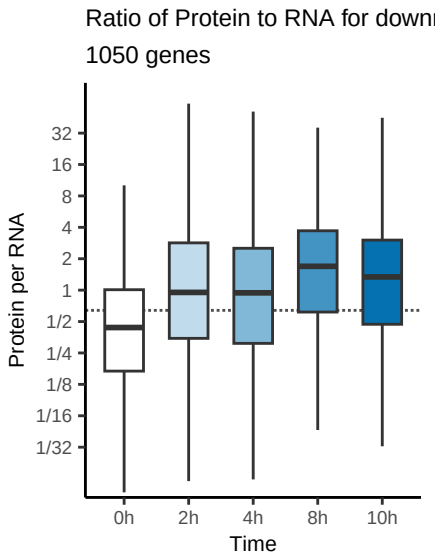
```
pairwise.t.test(upplotdat$prot_per_rna, upplotdat$timepoint, p.adjust.method = "none")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  upplotdat$prot_per_rna and upplotdat$timepoint
##
##     0h     2h     4h     8h
## 2h  0.7973 -      -      -
## 4h  0.1247 0.0732 -      -
## 8h  0.0195 0.0095 0.4226 -
## 10h 0.0145 0.0069 0.3631 0.9143
##
## P value adjustment method: none
```

```
summary(glht(aov(prot_per_rna ~ timepoint, data = upplotdat), linfct = mcp(timepoint = "Dunnett")))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
## Fit: aov(formula = prot_per_rna ~ timepoint, data = upplotdat)
##
## Linear Hypotheses:
##                Estimate Std. Error t value Pr(>|t|)
## 2h - 0h == 0    0.2164     0.8425   0.257   0.9971
## 4h - 0h == 0   -1.2938     0.8425  -1.536   0.3472
## 8h - 0h == 0   -1.9695     0.8425  -2.338   0.0654 .
## 10h - 0h == 0  -2.0601     0.8425  -2.445   0.0497 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
ggplot(downplotdat, aes(x = timepoint, y= prot_per_rna, fill=timepoint))+
    geom_hline(yintercept = median(plotdat$prot_per_rna), color = "grey30", lty = "11")+
    geom_boxplot(outlier.shape = NA, width = 0.6, notch = F)+
    ggtitle("Ratio of Protein to RNA for downregulated genes" ,paste0(length(unique(plotdat[plotdat$gene
    scale_y_continuous(trans="log2",
                                        breaks = c(1/32,1/16,1/8,1/4,1/2,1,2,4,8,16,32),
                                        labels = c("1/32","1/16","1/8","1/4","1/2","1","2","4","8","16
    scale_fill_manual(values = colorRampPalette(c("white","#0571b0"))(5))+
    coord_cartesian(ylim = c(1/64,64))+
    labs(y = "Protein per RNA", x = "Time")+
    theme(legend.position = "none", panel.grid = element_blank(), panel.border = element_blank(), axis.l
a
```




```
ggsave("plots/prot_per_rna_down.svg",a,"svg",width = 2.3, height = 3)
oneway.test(prot_per_rna ~ timepoint, data = downplotdat, var.equal = T)
```

```
##
##  One-way analysis of means
##
## data:  prot_per_rna and timepoint
## F = 5.5461, num df = 4, denom df = 5245, p-value = 0.0001878
```

```
pairwise.t.test(downplotdat$prot_per_rna, downplotdat$timepoint, p.adjust.method = "none")
```

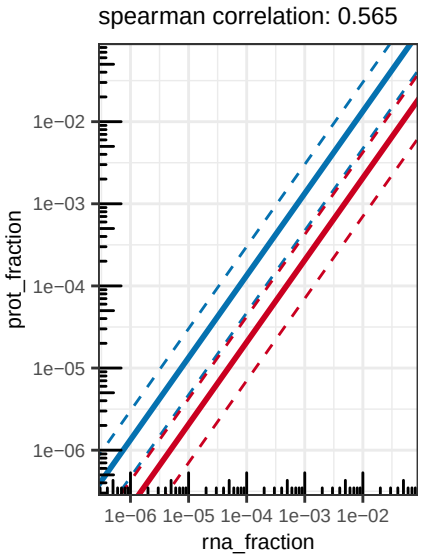
```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  downplotdat$prot_per_rna and downplotdat$timepoint
##
##     0h      2h    4h    8h
## 2h  0.010   -     -     -
## 4h  0.052   0.526 -     -
## 8h  4.3e-05 0.129 0.031 -
## 10h 9.4e-05 0.183 0.049 0.852
##
## P value adjustment method: none
```

```
summary(glht(aov(prot_per_rna ~ timepoint, data = downplotdat), linfct = mcp(timepoint = "Dunnett")))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Dunnett Contrasts
##
##
```

```
## Fit: aov(formula = prot_per_rna ~ timepoint, data = downplotdat)
##
## Linear Hypotheses:
##               Estimate Std. Error t value Pr(>|t|)
## 2h - 0h == 0    2.5097     0.9742   2.576    0.035 *
## 4h - 0h == 0    1.8921     0.9742   1.942    0.162
## 8h - 0h == 0    3.9890     0.9742   4.095   <0.001 ***
## 10h - 0h == 0   3.8072     0.9742   3.908   <0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
median_10 <- quantile(plotdat$prot_per_rna[plotdat$timepoint=="10h"])
median_10_down <- quantile(plotdat$prot_per_rna[plotdat$geneID%in%down_genes0_10&plotdat$timepoint=="10
median_10_up <- quantile(plotdat$prot_per_rna[plotdat$geneID%in%up_genes0_10&plotdat$timepoint=="10h"])


plotdat <- cbind.data.frame("prot_fraction" = fractions_df[,5], "rna_fraction" = fractions_df[,10])
rownames(plotdat) <- rownames(fractions_df)


corr <- round(cor(plotdat[,1],plotdat[,2], method = "spearman"),3)
ggplot(plotdat, aes(rna_fraction,prot_fraction))+
                geom_point(data = plotdat[rownames(plotdat)%in%down_genes0_10,], shape = 16, size = 1, a
                geom_point(data = plotdat[rownames(plotdat)%in%up_genes0_10,], shape = 16, size = 1, col
                #geom_point(alpha = 0.4, shape = 1)+
                geom_abline(slope = 1, size = 0.5, linetype = 2, intercept = log10(median_10_down[c(2,4)
                geom_abline(slope = 1, size = 1, linetype = 1, intercept = log10(median_10_down[3]), col
                geom_abline(slope = 1, size = 0.5, linetype = 2, intercept = log10(median_10_up[c(2,4)]]
                geom_abline(slope = 1, size = 1, linetype = 1, intercept = log10(median_10_up[3]), col="
                scale_x_log10()+#limits = c(1e-6,1e-2))+
                scale_y_log10()+#limits = c(1e-6,1e-2))+
                coord_cartesian(xlim = c(5e-7,5e-2),ylim = c(5e-7,5e-2))+
                labs(title = paste0("spearman correlation: ",corr))+
                annotation_logticks()->a
a
```




```
ggsave("plots/correlation_across_genes_10h_updown.svg",a,"svg",width = 3.4, height = 3.4)
```

Mefisto analysis

```
# Prepare data, and create MOFA object
all_samples <- sort(union(colnames(protein_data), colnames(rna_data)))

# MOFA requires protein data to have the same samples as RNA data, but samples with only NAs are allowe
protein_data_mefisto <- matrix(NA, nrow=nrow(protein_data), ncol=length(all_samples))
rownames(protein_data_mefisto) <- rownames(protein_data)
colnames(protein_data_mefisto) <- all_samples
protein_data_mefisto[rownames(protein_data_log),colnames(protein_data_log)] <- protein_data_log
protein_data_mefisto <- protein_data_mefisto[, colnames(rna_data)]

rna_data_mefisto <- rna_data_log

# Use all genes for which we have both protein and RNA data
use_genes <- intersect(rownames(protein_data_mefisto), rownames(rna_data_mefisto))
rna_data_mefisto <- rna_data_mefisto[intersect(rownames(rna_data_mefisto),use_genes), ]
protein_data_mefisto <- protein_data_mefisto[intersect(rownames(protein_data_mefisto),use_genes), ]


scaled_rna_data <- scale_rows(rna_data_mefisto)
scaled_protein_data <- scale_rows(protein_data_mefisto)


n_factors  <- 6

data <- list(protein=scaled_protein_data,
                        rna=scaled_rna_data)
meta_data_mofa <- meta_data_rna %>%
    mutate(Time = as.numeric(Time)) %>%
```

```
    dplyr::rename(sample = sample_id)

mofa_object <- create_mofa(data)
samples_metadata(mofa_object) <- meta_data_mofa
mofa_object <- set_covariates(mofa_object, covariates = "Time")

data_opts <- get_default_data_options(mofa_object)
model_opts <- get_default_model_options(mofa_object)
model_opts$num_factors <- n_factors
train_opts <- get_default_training_options(mofa_object)
train_opts$maxiter <- 1000
mefisto_opts <- get_default_mefisto_options(mofa_object)

mofa_object <- prepare_mofa(mofa_object, model_options = model_opts,
                                                        mefisto_options = mefisto_opts,
                                                        training_options = train_opts,
                                                        data_options = data_opts)
mofa_object  <- run_mofa(mofa_object, use_basilisk = T)
```

```
##
##         ############################################################
##         ###           __  __  ____  ______                       ###
##         ###          |  \/  |/ __ \|  ____/\    _                ###
##         ###          | \  / | |  | | |__ /  \ _| |_              ###
##         ###          | |\/| | |  | |  __/ /\ \_   _|             ###
##         ###          | |  | | |__| | | / ____ \|_|               ###
##         ###          |_|  |_|\____/|_|/_/    \_\                ###
##         ###                                                      ###
##         ############################################################
##
##
##
## use_float32 set to True: replacing float64 arrays by float32 arrays to speed up computations...
##
## Successfully loaded view='protein' group='group1' with N=24 samples and D=3604 features...
## Successfully loaded view='rna' group='group1' with N=24 samples and D=3604 features...
##
##
## Loaded 1 covariate(s) for each sample...
##
##
## Model options:
## - Automatic Relevance Determination prior on the factors: False
## - Automatic Relevance Determination prior on the weights: True
## - Spike-and-slab prior on the factors: False
## - Spike-and-slab prior on the weights: False
## Likelihoods:
## - View 0 (protein): gaussian
## - View 1 (rna): gaussian
##
##
##
##
```

```
## ######################################
## ## Training the model with seed 42 ##
## ######################################
##
##
## ELBO before training: -829702.18
##
## Iteration 1: time=0.01, ELBO=-226853.70, deltaELBO=602848.476 (72.65841819%), Factors=6
## Iteration 2: time=0.02, Factors=6
## Iteration 3: time=0.01, Factors=6
## Iteration 4: time=0.02, Factors=6
## Iteration 5: time=0.01, Factors=6
## Iteration 6: time=0.01, ELBO=-181921.18, deltaELBO=44932.519 (5.41549976%), Factors=6
## Iteration 7: time=0.01, Factors=6
## Iteration 8: time=0.01, Factors=6
## Iteration 9: time=0.01, Factors=6
## Iteration 10: time=0.01, Factors=6
## Iteration 11: time=0.01, ELBO=-181093.69, deltaELBO=827.485 (0.09973279%), Factors=6
## Iteration 12: time=0.01, Factors=6
## Iteration 13: time=0.01, Factors=6
## Iteration 14: time=0.01, Factors=6
## Iteration 15: time=0.01, Factors=6
## Iteration 16: time=0.01, ELBO=-180942.39, deltaELBO=151.308 (0.01823646%), Factors=6
## Iteration 17: time=0.01, Factors=6
## Iteration 18: time=0.01, Factors=6
## Iteration 19: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 20: time=0.38, Factors=6
## Iteration 21: time=0.01, ELBO=-180834.98, deltaELBO=107.404 (0.01294487%), Factors=6
## Iteration 22: time=0.01, Factors=6
## Iteration 23: time=0.01, Factors=6
## Iteration 24: time=0.01, Factors=6
## Iteration 25: time=0.01, Factors=6
## Iteration 26: time=0.01, ELBO=-180789.56, deltaELBO=45.425 (0.00547485%), Factors=6
## Iteration 27: time=0.01, Factors=6
## Iteration 28: time=0.01, Factors=6
## Iteration 29: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 30: time=0.41, Factors=6
## Iteration 31: time=0.02, ELBO=-180756.13, deltaELBO=33.430 (0.00402916%), Factors=6
## Iteration 32: time=0.01, Factors=6
## Iteration 33: time=0.01, Factors=6
## Iteration 34: time=0.01, Factors=6
## Iteration 35: time=0.01, Factors=6
## Iteration 36: time=0.01, ELBO=-180729.82, deltaELBO=26.311 (0.00317115%), Factors=6
## Iteration 37: time=0.01, Factors=6
## Iteration 38: time=0.01, Factors=6
## Iteration 39: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 40: time=0.33, Factors=6
## Iteration 41: time=0.01, ELBO=-180708.25, deltaELBO=21.563 (0.00259891%), Factors=6
## Iteration 42: time=0.01, Factors=6
## Iteration 43: time=0.01, Factors=6
## Iteration 44: time=0.01, Factors=6
```

```
## Iteration 45: time=0.01, Factors=6
## Iteration 46: time=0.01, ELBO=-180690.10, deltaELBO=18.155 (0.00218811%), Factors=6
## Iteration 47: time=0.01, Factors=6
## Iteration 48: time=0.01, Factors=6
## Iteration 49: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 50: time=0.34, Factors=6
## Iteration 51: time=0.01, ELBO=-180674.56, deltaELBO=15.538 (0.00187268%), Factors=6
## Iteration 52: time=0.01, Factors=6
## Iteration 53: time=0.01, Factors=6
## Iteration 54: time=0.01, Factors=6
## Iteration 55: time=0.01, Factors=6
## Iteration 56: time=0.01, ELBO=-180661.10, deltaELBO=13.463 (0.00162260%), Factors=6
## Iteration 57: time=0.01, Factors=6
## Iteration 58: time=0.01, Factors=6
## Iteration 59: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 60: time=0.33, Factors=6
## Iteration 61: time=0.01, ELBO=-180649.34, deltaELBO=11.756 (0.00141693%), Factors=6
## Iteration 62: time=0.01, Factors=6
## Iteration 63: time=0.01, Factors=6
## Iteration 64: time=0.01, Factors=6
## Iteration 65: time=0.01, Factors=6
## Iteration 66: time=0.02, ELBO=-180638.99, deltaELBO=10.347 (0.00124712%), Factors=6
## Iteration 67: time=0.01, Factors=6
## Iteration 68: time=0.01, Factors=6
## Iteration 69: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 70: time=0.37, Factors=6
## Iteration 71: time=0.01, ELBO=-180629.87, deltaELBO=9.129 (0.00110024%), Factors=6
## Iteration 72: time=0.01, Factors=6
## Iteration 73: time=0.01, Factors=6
## Iteration 74: time=0.01, Factors=6
## Iteration 75: time=0.01, Factors=6
## Iteration 76: time=0.01, ELBO=-180621.76, deltaELBO=8.109 (0.00097735%), Factors=6
## Iteration 77: time=0.01, Factors=6
## Iteration 78: time=0.01, Factors=6
## Iteration 79: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 80: time=0.32, Factors=6
## Iteration 81: time=0.01, ELBO=-180614.53, deltaELBO=7.230 (0.00087143%), Factors=6
## Iteration 82: time=0.01, Factors=6
## Iteration 83: time=0.01, Factors=6
## Iteration 84: time=0.01, Factors=6
## Iteration 85: time=0.01, Factors=6
## Iteration 86: time=0.01, ELBO=-180608.10, deltaELBO=6.428 (0.00077472%), Factors=6
## Iteration 87: time=0.01, Factors=6
## Iteration 88: time=0.01, Factors=6
## Iteration 89: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 90: time=0.31, Factors=6
## Iteration 91: time=0.01, ELBO=-180602.32, deltaELBO=5.777 (0.00069628%), Factors=6
## Iteration 92: time=0.01, Factors=6
## Iteration 93: time=0.01, Factors=6
```

```
## Iteration 94: time=0.01, Factors=6
## Iteration 95: time=0.01, Factors=6
## Iteration 96: time=0.01, ELBO=-180597.12, deltaELBO=5.203 (0.00062705%), Factors=6
## Iteration 97: time=0.01, Factors=6
## Iteration 98: time=0.01, Factors=6
## Iteration 99: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 100: time=0.31, Factors=6
## Iteration 101: time=0.01, ELBO=-180592.47, deltaELBO=4.652 (0.00056068%), Factors=6
## Iteration 102: time=0.01, Factors=6
## Iteration 103: time=0.01, Factors=6
## Iteration 104: time=0.01, Factors=6
## Iteration 105: time=0.01, Factors=6
## Iteration 106: time=0.01, ELBO=-180588.25, deltaELBO=4.217 (0.00050828%), Factors=6
## Iteration 107: time=0.01, Factors=6
## Iteration 108: time=0.01, Factors=6
## Iteration 109: time=0.01, Factors=6
## Optimising sigma node...
## Iteration 110: time=0.32, Factors=6
## Iteration 111: time=0.01, ELBO=-180584.46, deltaELBO=3.792 (0.00045708%), Factors=6
## Iteration 112: time=0.01, Factors=6
## Iteration 113: time=0.01, Factors=6
## Iteration 114: time=0.01, Factors=6
## Iteration 115: time=0.01, Factors=6
## Iteration 116: time=0.01, ELBO=-180580.96, deltaELBO=3.496 (0.00042138%), Factors=6
##
## Converged!
##
##
##
## ########################
## ## Training finished ##
## ########################
##
##
## Saving model in /var/folders/fn/d9s_md6973q56nxwtrbqy8z5tnn75_/T//RtmpLmzzBC/mofa_20240613-203609.hd
```

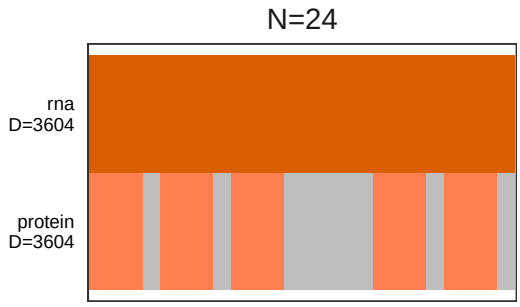
```
mofa_object
```

```
## Trained MEFISTO with the following characteristics:
##  Number of views: 2
##  Views names: protein rna
##  Number of features (per view): 3604 3604
##  Number of groups: 1
##  Groups names: group1
##  Number of samples (per group): 24
##  Number of covariates per sample: 1
##  Number of factors: 6
```

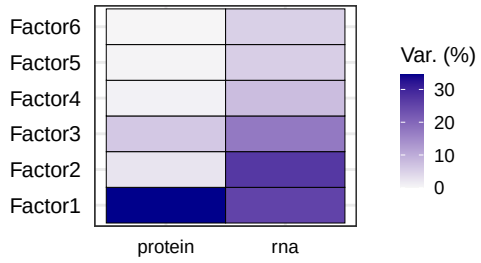
```
## Flip factor 1, so positive values mean increasing expression
# mofa_object@expectations[[3]][[1]][,"Factor1"] <- -1 * mofa_object@expectations[[3]][[1]][,"Factor1"]
# mofa_object@expectations[[3]][[2]][,"Factor1"] <- -1 * mofa_object@expectations[[3]][[2]][,"Factor1"]
# mofa_object@expectations[[2]][[1]][,"Factor1"] <- -1 * mofa_object@expectations[[2]][[1]][,"Factor1"]
```

	protein	rna
Factor1	34.47	25.26
Factor2	2.56	27.01
Factor3	6.24	17.28
Factor4	0.79	7.74
Factor5	0.57	5.45
Factor6	0.20	5.20

```
plot_data_overview(mofa_object)
```



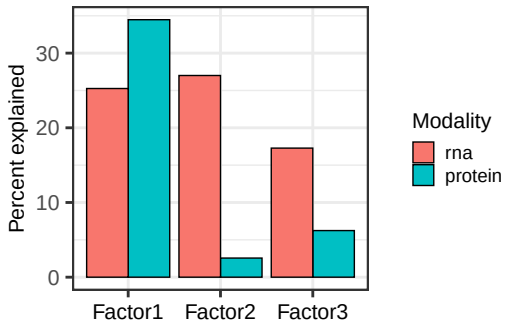

```
plot_variance_explained(mofa_object)
```



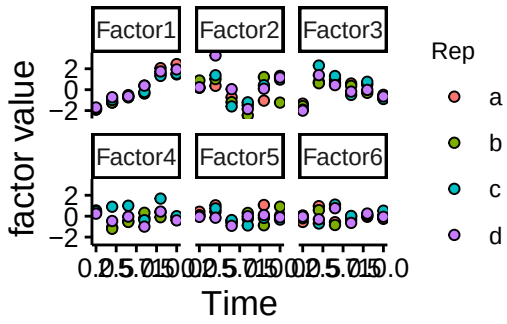

```
calculate_variance_explained(mofa_object)$r2_per_factor$group1 %>%
    data.frame() %>%
    mutate(protein = round(protein,2), rna = round(rna,2)) %>%
    kable %>%
    kable_styling(font_size = 10, fixed_thead = T) %>%
    row_spec(0, color="white", background="black")
```

```
plot_data <- calculate_variance_explained(mofa_object)$r2_per_factor$group1 %>%
    as.data.frame() %>%
    rownames_to_column(var="Factor") %>%
    pivot_longer(cols = -Factor, names_to = "Modality", values_to = "Percent_explained") %>%
    mutate(Modality = factor(Modality, levels = c("rna","protein")))

ggplot(plot_data[1:6,], aes(x=Factor, y=Percent_explained, fill=Modality)) +
```

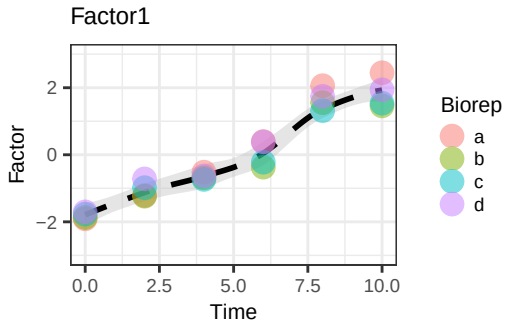
```
    geom_bar(position="dodge", stat="identity", col = "black", lwd = 0.25)+
    labs(y="Percent explained")+
    theme(axis.text = element_text(size = 8), axis.title.x = element_blank(), axis.text.x = element_tex
a
```

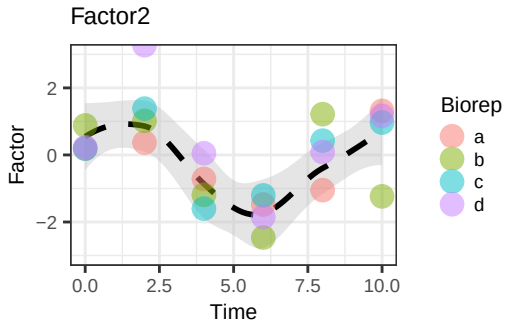


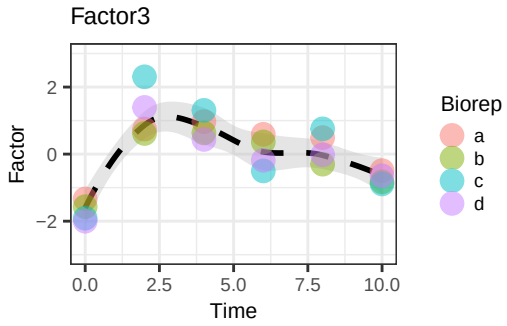

```
ggsave("plots/perc_varience_explained.svg",a,"svg",width = 2.8, height = 1.5)

plot_factors_vs_cov(mofa_object, color_by="Rep", factors=1:n_factors )
```

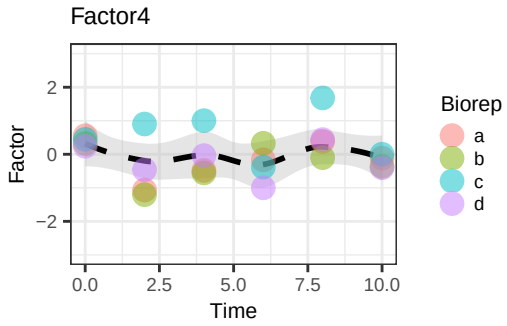


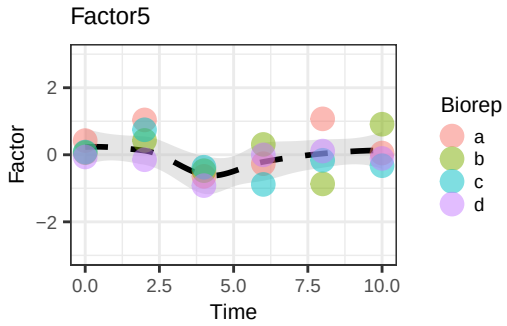

```
for(tmp_factor in colnames(get_factors(mofa_object)[[1]])){
    plot_data <- data.frame(Factor=get_factors(mofa_object)[[1]][,tmp_factor],
                                                    Time=mofa_object@covariates[[1]][1,],
                                                    Biorep=rep(c("a","b","c","d"), 6))

    p <- ggplot(plot_data, aes(y=Factor, x=Time, col=Biorep)) +
        geom_smooth(fill="gray75", col="black", lty = "dashed") +
        geom_point(size=3, stroke = 1, alpha = 0.5) +
        coord_cartesian(ylim = c(-3,3))+
        labs(title=tmp_factor)

    print(p)
}
```

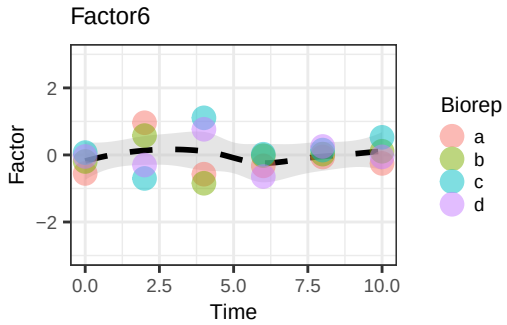

Factor1
Factor
2
0
−2
0.0
2.5
5.0
7.5
10.0
Time
Biorep
a
b
c
d


Factor2
Factor
2
0
−2
0.0
2.5
5.0
7.5
10.0
Time
Biorep
a
b
c
d


Factor3
Factor
2
0
−2
0.0
2.5
5.0
7.5
10.0
Time
Biorep
a
b
c
d

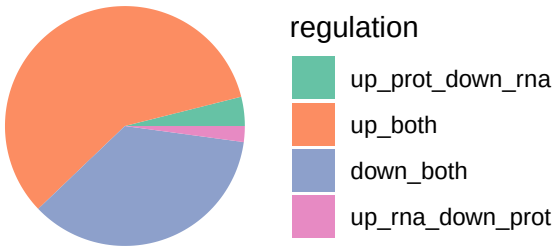

Factor4
Factor
2
0
−2
0.0
2.5
5.0
7.5
10.0
Time
Biorep
a
b
c
d





RNA vs proteins in factors

```
# get gene ids present in both data sets
use_genes <- intersect(rownames(data$protein), rownames(data$rna))

# extract loadings for these genes
loadings <- get_expectations(mofa_object, "W")

prot_vals <- loadings[["protein"]][paste(use_genes,"_protein", sep=""), 1]
rna_vals <- loadings[["rna"]][paste(use_genes,"_rna", sep=""), 1]
loadings_1 <- data.frame(gene=names(prot_vals), protein=prot_vals, rna=rna_vals)
loadings_1$regulation <- "none"
loadings_1$regulation[loadings_1$protein>0.45&loadings_1$rna>0.45] <- "up_both"
loadings_1$regulation[loadings_1$protein< -0.45&loadings_1$rna< -0.45] <- "down_both"
loadings_1$regulation[loadings_1$protein>0.45&loadings_1$rna< -0.45] <- "up_prot_down_rna"
loadings_1$regulation[loadings_1$protein< -0.45&loadings_1$rna>0.45] <- "up_rna_down_prot"
loadings_1$regulation <- factor(loadings_1$regulation, c("up_prot_down_rna","up_both","down_both","up_r
loadings_1$group <- ""

ggplot(loadings_1[loadings_1$regulation!="none",], aes(x= group, fill = regulation))+
    geom_bar(position = "stack")+
    scale_fill_brewer(palette = "Set2")+
    coord_polar("y", start = 0.5*pi)+
    theme_void()->a
a
```




```
ggsave("plots/pie_factor1_045cutoff.svg",a,"svg",height = 4, width = 4)
summary(loadings_1$regulation)
```

```
## up_prot_down_rna          up_both        down_both up_rna_down_prot
##               11              163              100                6
##             none
##             3324
```

```
use_genes <- substr(loadings_1$gene[loadings_1$regulation!="none"],1,6)

table_fact1 <- cbind.data.frame(
    gene_info[match(use_genes,gene_info$UniProt_ID),],
    loadings_1[loadings_1$regulation!="none",c(2:4)]
)

head(table_fact1, n = 10)
```

```
##                     GENE_ID    Gene_Name                        Synonyms
## B0G0Z7_protein DDB_G0294597 DDB_G0294597                            <NA>
## B0G117_protein DDB_G0278053         ilvB                            <NA>
## B0G191_protein DDB_G0291800 DDB_G0291800                            <NA>
## B0G198_protein DDB_G0295683 DDB_G0295683                            <NA>
## C7G034_protein DDB_G0285063 DDB_G0285063                            <NA>
## O76765_protein DDB_G0285845         phdA                         DG1093
## O96042_protein DDB_G0291253         dia2                            <NA>
## O97113_protein DDB_G0275439         cad2                         DdCAD2
## P02888_protein DDB_G0273067       dscD-1     dscD, discoidin I, D chain
## P03967_protein DDB_G0292996         rasD ras, Dd-ras, Ddras, rasA, 12E2
##
## B0G0Z7_protein                                                                beta-lacta
## B0G117_protein                        thiamine pyrophosphate-binding enzyme family protein, 2-hydroxy
## B0G191_protein                                      glutathione-dependent formaldehyde-activating
## B0G198_protein LIM-type zinc finger-containing protein, cofilin/tropomyosin type actin binding domai
```

```
## C7G034_protein                                                                  putative E3 ubiqu
## O76765_protein                                                                         PH domai
## O96042_protein
## O97113_protein                                                                           putati
## P02888_protein                                                                             disc
## P03967_protein
##                UniProt_ID    protein        rna       regulation
## B0G0Z7_protein     B0G0Z7 -0.4907367 -0.4519353        down_both
## B0G117_protein     B0G117 -0.5146492 -0.5658317        down_both
## B0G191_protein     B0G191 -0.5210975 -0.5408589        down_both
## B0G198_protein     B0G198 -0.4605801 -0.6111194        down_both
## C7G034_protein     C7G034  0.4848203  0.5760756          up_both
## O76765_protein     O76765  0.6048333  0.6162034          up_both
## O96042_protein     O96042  0.6117399  0.5026487          up_both
## O97113_protein     O97113  0.6076571  0.6466885          up_both
## P02888_protein     P02888  0.4897117 -0.5559713 up_prot_down_rna
## P03967_protein     P03967  0.5799839  0.5730160          up_both
```

```
write.table(table_fact1, file = "tables/table_fact1.tsv", sep = "\t", quote = F, row.names = T, col.name
```

Gene Onotology vs MEFISTO factors (GSEA)

Gene Set Enrichment Analysis is used to see if any Gene Ontology terms have higher (or lower) levels of the MEFISTO factors than expected by chance. We do this separately for the RNA and protein data.

```
gsa_res_lst_rna <- list()
i <- 1
max_gsea_pval <- 1e-3
max_gsea_terms <- 50

for(f in 1:3){
    cat("#### Factor ", f, " \n\n")

    factor_data_tmp <- get_weights(mofa_object, views="rna", factors = f)
    factor_data <- factor_data_tmp$rna[,1]
    names(factor_data) <- rownames(factor_data_tmp$rna) %>%
        str_replace(., "_rna", "")

    for(ont in c("P")){
        #cat("##### ", ont," \n\n")

        piano_go_data <- gene_annot_tab %>%
            #dplyr::filter(GO_TYPE == ont) %>%
            dplyr::select(UniProt_ID, GO_TERM, GO_DESCRIPTION, GO_TYPE) %>%
            unite(col = GO, GO_TYPE, GO_TERM, GO_DESCRIPTION, sep = " ")

        dicty_gsc <- loadGSC(piano_go_data)
        gsa_res <- runGSA(factor_data,
                                          gsc = dicty_gsc,
                                          gsSizeLim = c(10, Inf),
```

Name	Genes (tot)	Stat (dist.dir)	p adj (dist.dir.up)

Name	Genes (tot)	Stat (dist.dir)	p adj (dist.dir.dn)
P GO:0006412 translation	181	-0.44821	0
F GO:0003735 structural constituent of ribosome	112	-0.51483	0
C GO:0005840 ribosome	110	-0.53431	0

```
                                        geneSetStat = "gsea",
                                        ncpus = 4)

        gsa_res_lst_rna[[i]] <- gsa_res
        i <- i +1

        gsea_up_tab <- GSAsummaryTable(gsaRes = gsa_res) %>%
            dplyr::filter(`p adj (dist.dir.up)`<max_gsea_pval) %>%
            arrange(`p adj (dist.dir.up)`) %>%
            dplyr::select("Name", "Genes (tot)", "Stat (dist.dir)", "p adj (dist.dir.up)") %>%
            head(max_gsea_terms)

        gsea_down_tab <- GSAsummaryTable(gsaRes = gsa_res) %>%
            dplyr::filter(`p adj (dist.dir.dn)`<max_gsea_pval) %>%
            arrange(`p adj (dist.dir.dn)`) %>%
            dplyr::select("Name", "Genes (tot)", "Stat (dist.dir)", "p adj (dist.dir.dn)") %>%
            head(max_gsea_terms)


        gsea_up_tab %>%
            kable(booktabs = TRUE) %>%
            kable_styling(font_size = 10, fixed_thead = T) %>%
            row_spec(0, color="white", background="black") %>%
            scroll_box(width = "100%", height = "300px") %>%
            print

        gsea_down_tab %>%
            kable(booktabs = TRUE) %>%
            kable_styling(font_size = 10) %>%
            row_spec(0, color="white", background="black") %>%
            scroll_box(width = "100%", height = "300px") %>%
            print


        write.table(gsea_up_tab, file=paste("tables/GSEA_RNA_F", f, "_up.tsv" ,sep=""), row.names = F, s
        write.table(gsea_down_tab, file=paste("tables/GSEA_RNA_F", f, "_down.tsv" ,sep=""), row.names =
    }
}
```

RNA

Factor 1 Checking arguments... done! Calculating gene set statistics... done! Calculating gene set significance... done! Adjusting for multiple testing... done!

Factor 2 Checking arguments... done! Calculating gene set statistics... done! Calculating gene set significance... done! Adjusting for multiple testing... done!

Factor 3 Checking arguments... done! Calculating gene set statistics... done! Calculating gene set significance... done! Adjusting for multiple testing... done!

Protein

```
gsa_res_lst_prot <- list()
i <- 1

for(f in 1:3){
    cat("#### Factor ", f, " \n\n")

    factor_data_tmp <- get_weights(mofa_object, views="protein", factors = f)
    factor_data <- factor_data_tmp$protein[,1]
    names(factor_data) <- rownames(factor_data_tmp$protein) %>%
        str_replace(., "_protein", "")

    for(ont in c("P")){
        #cat("##### ", ont," \n\n")

        piano_go_data <- gene_annot_tab %>%
            #dplyr::filter(GO_TYPE == ont) %>%
            dplyr::select(UniProt_ID, GO_TERM, GO_DESCRIPTION, GO_TYPE) %>%
            unite(col = GO, GO_TYPE, GO_TERM, GO_DESCRIPTION, sep = " ")

        dicty_gsc <- loadGSC(piano_go_data)
        gsa_res <- runGSA(factor_data,
                                            gsc = dicty_gsc,
                                            gsSizeLim = c(10, Inf),
                                            geneSetStat = "gsea",
                                            ncpus = 4)

        gsa_res_lst_prot[[i]] <- gsa_res
        i <- i +1

    gsea_up_tab <- GSAsummaryTable(gsaRes = gsa_res) %>%
            dplyr::filter(`p adj (dist.dir.up)`< max_gsea_pval ) %>%
            arrange(`p adj (dist.dir.up)`) %>%
            dplyr::select("Name", "Genes (tot)", "Stat (dist.dir)", "p adj (dist.dir.up)") %>%
            head(max_gsea_terms)

        gsea_down_tab <-GSAsummaryTable(gsaRes = gsa_res) %>%
            dplyr::filter(`p adj (dist.dir.dn)`< max_gsea_pval ) %>%
            arrange(`p adj (dist.dir.dn)`) %>%
            dplyr::select("Name", "Genes (tot)", "Stat (dist.dir)", "p adj (dist.dir.dn)") %>%
            head(max_gsea_terms)


        gsea_up_tab %>%
```

Name	Genes (tot)	Stat (dist.dir)	p adj (dist.dir.up)
C GO:0005615 extracellular space	209	0.35760	0.0000000
C GO:0045335 phagocytic vesicle	266	0.37838	0.0000000
P GO:0006412 translation	181	0.40368	0.0000000
P GO:0000281 mitotic cytokinesis	64	0.44270	0.0000000
P GO:0006909 phagocytosis	51	0.46516	0.0000000
C GO:0005938 cell cortex	83	0.54912	0.0000000
C GO:0030864 cortical actin cytoskeleton	23	0.71009	0.0000000
C GO:0001891 phagocytic cup	23	0.71135	0.0000000
F GO:0051015 actin filament binding	48	0.70770	0.0000000
C GO:0015629 actin cytoskeleton	38	0.59214	0.0000000
F GO:0003779 actin binding	87	0.61612	0.0000000
C GO:0031252 cell leading edge	47	0.59456	0.0000000
C GO:0031143 pseudopodium	33	0.68753	0.0000000
P GO:0009617 response to bacterium	46	0.55659	0.0000000
F GO:0044183 protein folding chaperone	10	0.77309	0.0000000
F GO:0003735 structural constituent of ribosome	112	0.47161	0.0000000
C GO:0005840 ribosome	110	0.51342	0.0000000
C GO:0022625 cytosolic large ribosomal subunit	52	0.62406	0.0000000
C GO:0030529 NA	122	0.48422	0.0000000
F GO:0005089 NA	17	0.71702	0.0000000
P GO:0035023 regulation of Rho protein signal transduction	19	0.67300	0.0000000
C GO:0005856 cytoskeleton	84	0.55462	0.0000000
C GO:0005694 chromosome	18	0.62852	0.0000000
C GO:0060187 cell pole	10	0.78934	0.0000000
C GO:0044291 cell-cell contact zone	11	0.81676	0.0000000
P GO:0045010 actin nucleation	15	0.68560	0.0000000
C GO:0042995 cell projection	22	0.75766	0.0000000
C GO:0022627 cytosolic small ribosomal subunit	32	0.70528	0.0000000
C GO:0005885 Arp2/3 protein complex	10	0.76145	0.0000528
C GO:0030175 filopodium	18	0.60453	0.0002380
F GO:0005200 structural constituent of cytoskeleton	12	0.68302	0.0004701
P GO:0051017 actin filament bundle assembly	13	0.66867	0.0004853
P GO:0007010 cytoskeleton organization	11	0.70915	0.0006900
P GO:0006418 tRNA aminoacylation for protein translation	29	0.50883	0.0007109
P GO:0030041 actin filament polymerization	26	0.52360	0.0007140
P GO:0051591 response to cAMP	11	0.70242	0.0007579

Name	Genes (tot)	Stat (dist.dir)	p adj (dist.dir.dn)
C GO:0000502 proteasome complex	38	-0.68038	0.0000000
C GO:0005839 proteasome core complex	16	-0.73805	0.0000000
F GO:0004298 threonine-type endopeptidase activity	15	-0.77680	0.0000000
C GO:0031201 SNARE complex	22	-0.66523	0.0001108

Name	Genes (tot)	Stat (dist.dir)	p adj (dist.dir.up)
C GO:0016021 NA	501	0.12721	0
C GO:0016020 membrane	674	0.14424	0
P GO:0051603 proteolysis involved in protein catabolic process	27	0.69240	0
C GO:0000502 proteasome complex	38	0.68827	0
C GO:0005839 proteasome core complex	16	0.72859	0
F GO:0004298 threonine-type endopeptidase activity	15	0.78209	0

Name	Genes
F GO:0003676 nucleic acid binding	
C GO:0005730 nucleolus	
C GO:0032040 small-subunit processome	
P GO:0042254 ribosome biogenesis	
P GO:0000462 maturation of SSU-rRNA from tricistronic rRNA transcript (SSU-rRNA, 5.8S rRNA, LSU-rRNA)	
P GO:0006364 rRNA processing	
F GO:0004004 NA	
P GO:0008033 tRNA processing	
F GO:0003729 mRNA binding	

```
            kable(booktabs = TRUE) %>%
            kable_styling(font_size = 10, fixed_thead = T) %>%
            row_spec(0, color="white", background="black") %>%
            scroll_box(width = "100%", height = "300px") %>%
            print

        gsea_down_tab %>%
            kable(booktabs = TRUE) %>%
            kable_styling(font_size = 10) %>%
            row_spec(0, color="white", background="black") %>%
            scroll_box(width = "100%", height = "300px") %>%
            print


        write.table(gsea_up_tab, file=paste("tables/GSEA_PROT_F", f, "_up.tsv" ,sep=""), row.names = F,
        write.table(gsea_down_tab, file=paste("tables/GSEA_PROT_F", f, "_down.tsv" ,sep=""), row.names =

    }
}
```

Factor 1 Checking arguments... done! Calculating gene set statistics... done! Calculating gene set significance... done! Adjusting for multiple testing... done!

Factor 2 Checking arguments... done! Calculating gene set statistics... done! Calculating gene set significance... done! Adjusting for multiple testing... done!

Factor 3 Checking arguments... done! Calculating gene set statistics... done! Calculating gene set significance... done! Adjusting for multiple testing... done!

Name	Genes (tot)	Stat (dist.dir)	p adj
F GO:0051015 actin filament binding	48	0.59055	
F GO:0003779 actin binding	87	0.50147	
P GO:0043161 proteasome-mediated ubiquitin-dependent protein catabolic process	20	0.80382	
C GO:0000502 proteasome complex	38	0.82335	
P GO:0006511 ubiquitin-dependent protein catabolic process	44	0.67932	
F GO:0004175 endopeptidase activity	19	0.74904	
C GO:0005839 proteasome core complex	16	0.82123	
F GO:0004298 threonine-type endopeptidase activity	15	0.86798	
C GO:0008540 proteasome regulatory particle, base subcomplex	10	0.86737	
C GO:0005938 cell cortex	83	0.45893	
P GO:0031156 regulation of sorocarp development	11	0.77894	
P GO:0030433 ubiquitin-dependent ERAD pathway	15	0.74259	
C GO:0005783 endoplasmic reticulum	98	0.43073	
P GO:0030435 sporulation resulting in formation of a cellular spore	35	0.53955	
P GO:0051603 proteolysis involved in protein catabolic process	27	0.57530	
F GO:0036459 NA	13	0.70913	
P GO:0016579 protein deubiquitination	13	0.70913	
P GO:0031152 aggregation involved in sorocarp development	69	0.45788	

Name	Genes
C GO:0005739 mitochondrion	
C GO:0005730 nucleolus	
C GO:0030687 preribosome, large subunit precursor	
P GO:0042254 ribosome biogenesis	
P GO:0006364 rRNA processing	
P GO:0016126 sterol biosynthetic process	
P GO:0008202 steroid metabolic process	
F GO:0003735 structural constituent of ribosome	
F GO:0004812 aminoacyl-tRNA ligase activity	
F GO:0016491 oxidoreductase activity	
C GO:0005759 mitochondrial matrix	
F GO:0019843 rRNA binding	
P GO:0000462 maturation of SSU-rRNA from tricistronic rRNA transcript (SSU-rRNA, 5.8S rRNA, LSU-rRNA)	
P GO:0006629 lipid metabolic process	
P GO:0000027 ribosomal large subunit assembly	

Name	Genes (tot)	Stat (dist.dir)	p adj (dist.dir.up)

Name	Genes (tot)	Stat (dist.dir)	p adj (dist.dir.dn)
P GO:0006635 fatty acid beta-oxidation	19	-0.73981	0.0000000
C GO:0005777 peroxisome	34	-0.62998	0.0003383

Name	Genes (tot)	Stat (dist.dir)	p adj (dist.dir.up)

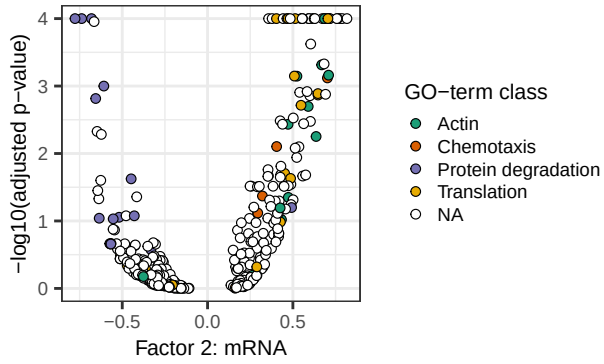
Name	Genes (tot)	Stat (dist.dir)	p adj (dist.dir.dn)

Combining results for RNA and protein

```
factor1_bp_rna <- GSAsummaryTable(gsaRes = gsa_res_lst_rna[[1]])

factor2_bp_rna <- GSAsummaryTable(gsaRes = gsa_res_lst_rna[[2]])

factor3_bp_rna <- GSAsummaryTable(gsaRes = gsa_res_lst_rna[[3]])

factor1_bp_prot <- GSAsummaryTable(gsaRes = gsa_res_lst_prot[[1]])

factor2_bp_prot <- GSAsummaryTable(gsaRes = gsa_res_lst_prot[[2]])

factor3_bp_prot <- GSAsummaryTable(gsaRes = gsa_res_lst_prot[[3]])


combine_gsea_results <- function(rna_res, protein_res, xlabel="rna", ylabel="protein", title=""){
    plot_data <- as_tibble(rna_res) %>%
        inner_join(as_tibble(protein_res), by = "Name", suffix = c(".rna", ".prot")) %>%
        dplyr::rename_with( ~ gsub("\\ \\(dist.dir\\)", "", .x), fixed = TRUE) %>%
        dplyr::rename_with( ~ gsub("\\ \\(tot\\)", "", .x), fixed = TRUE) %>%
        rowwise() %>%
        mutate(min_padj = min(`p adj (dist.dir.up).rna`, `p adj (dist.dir.dn).rna`, `p adj (dist.dir.up
        # mutate(min_padj = min(`p (dist.dir.up).rna`, `p (dist.dir.dn).rna`, `p (dist.dir.up).prot`, `
        mutate(minus_log10_padj = -1*log10(min_padj+1e-6)) %>%
        dplyr::select(Name, Genes.rna, Stat.rna, Stat.prot, minus_log10_padj) %>%
        filter(minus_log10_padj >= 2) %>%
        dplyr::rename(Genes = Genes.rna) %>%
        mutate(type = case_when(
            grepl("ribo|translation|small-subunit|rRNA", Name) ~ "Ribosome",
            grepl("phagocytosis", Name) ~ "Phagocytosis",
            grepl("pinocytosis", Name) ~ "Pinocytosis",
            grepl("ubiquitin|proteolysis|proteas|peptidase", Name) ~ "Protein degradation",
            grepl("peroxisome|oxidation", Name) ~ "Oxidation",
            grepl("actin|cortex ", Name) ~ "Actin",
            grepl("membrane", Name) ~ "Membrane",
            grepl("metabo|tricarboxylic", Name) ~ "Metabolism",
            grepl("protein binding", Name) ~ "Protein binding",
            grepl("sorocarp|development", Name) ~ "Sorocarp",
            grepl("chemotaxis|motility|migration", Name) ~ "Chemotaxis",
            .default = NULL
        ))%>%
        mutate(order = case_when(is.na(type) ~ 2, .default = 1))

    ggplot(plot_data, aes(x = Stat.rna, y = Stat.prot, size=minus_log10_padj, colour=type, order=order)
        geom_point() +
        scale_size(range = c(1,4), breaks = 2:4, limits = c(2,4), labels = c(0.01,0.001,0.0001))+
        xlim(-1,1) +
        ylim(-1,1) +
        labs(x=xlabel,y=ylabel,size="p-value",color="GO-term")
}
```

```
factor2_bp_rna %>%
    `colnames<-`(c("Name","genes","stat","pup","padjup","pdown","padjdown","genesup","genesdown"))%>%
mutate(type = case_when(
            grepl("ribo|translation|small-subunit|rRNA", Name) ~ "Translation",
            grepl("ubiquitin|proteolysis|proteas|peptidase", Name) ~ "Protein degradation",
            grepl("actin|cortex|cytoskeleton ", Name) ~ "Actin",
            grepl("chemotaxis|motility|migration|cAMP", Name) ~ "Chemotaxis",
            .default = NULL
        )) %>% mutate(padj = coalesce(padjup, padjdown))%>%
    rowwise() %>%
    mutate(logpadj = min(-log10(padj),4)) -> factor2_bp_rna_classified


ggplot(factor2_bp_rna_classified, aes(y = logpadj,x= stat, fill = type))+
    geom_point(shape = 21, size = 1.5, stroke = 0.3)+
    scale_fill_manual(values = brewer.pal(9, "Dark2")[-c(4,5)], na.value = "white")+
    labs(y = "-log10(adjusted p-value)", x = "Factor 2: mRNA",fill = "GO-term class")->a
a
```

Biological process




```
ggsave("plots/factor2_RNA_goterm.svg",a,"svg",width = 3.3,height = 2)
```

Genes selected from functional annotations

```
use_genes <- substr(loadings_1$gene[loadings_1$regulation=="up_prot_down_rna"|loadings_1$regulation=="u

lines_data <- rbind(
    pivot_longer(data.frame(scale_rows(log10(rna_data_means[use_genes,]))), gene = use_genes, data = "mR
    pivot_longer(data.frame(scale_rows(log10(prot_data_means[use_genes,]))), gene = use_genes, data = "p
lines_data$name <- as.numeric(gsub('[Xh]','',lines_data$name))

heatmap_data <- lines_data
heatmap_data$colname <- paste0(heatmap_data$data,"_",heatmap_data$name)
heatmap_data <- heatmap_data[,c(1,5,4)]
```

```r
heatmap_data <- pivot_wider(heatmap_data, names_from = colname)
heatmap_data <- heatmap_data[,-1]
rownames(heatmap_data) <- uniprot_to_gene_name[use_genes]

colnames(heatmap_data) <- c("0","2","4","6","8","10","0","2","4","8","10")

#svglite("plots/factor1_updown_heatmap.svg",width = 3.7,height = 3)
Heatmap(heatmap_data,
                cluster_columns = FALSE,
                cluster_rows = T,use_raster = F,
                show_row_names = T, row_split = 2, show_row_dend = F,
                col = colorRamp2(2:-2, colorRampPalette(brewer.pal(11,"RdYlBu"))(5)),
                column_names_side = c("top"),
                column_names_rot = 0,   column_names_centered = T, column_names_gp = gpar(fontsize = 7)
                heatmap_legend_param = list(title = "z-score",
                                                                labels_gp = gpar(fontsize = 6),
                                                                title_gp = gpar(fontsize = 6, f
                                                                grid_width = unit(3,"mm")),
                column_split = factor(c(rep("mRNA",6),rep("protein",5)), levels = c("mRNA","protein")),
                #cluster_column_slices = FALSE,
                #column_gap = unit(c(rep(0,10),2,rep(0,5)), "mm"),
                column_title_gp = gpar(fontsize = 8, fontface="bold"),
                row_title_gp = gpar(fontsize = 0)
)->a
a
#dev.off()



use_genes <- names(uniprot_to_gene_name)[uniprot_to_gene_name%in%c("arcA","arcB","arcC","arcD","arcE","a


lines_data <- rbind(
    pivot_longer(data.frame(scale_rows(log10(rna_data_means[use_genes,]))), gene = c("arcA","arcB","arcC
    pivot_longer(data.frame(scale_rows(log10(prot_data_means[use_genes,]))), gene = c("arcA","arcB","arc
lines_data$name <- as.numeric(gsub('[Xh]','',lines_data$name))

ggplot(lines_data, aes(name, value))+
    geom_smooth(color = "grey20", level = 0.95)+
    geom_point(aes(fill = gene), shape = 21, stroke = 0.3)+
    scale_fill_brewer(palette = "Dark2")+
    scale_x_continuous("time(h)", breaks = seq(0,10,2))+
    scale_y_continuous("z-score")+
    facet_grid(~data)->a
a
ggsave("plots/arp2_3_line.svg",a,"svg",width = 3.7, height = 2)

use_genes <- gene_annot_tab %>%
    filter(GO_TERM == "GO:0000502") %>% #Proteasome complex
    dplyr::select(UniProt_ID) %>%
    as.data.frame() %>%
    .[,1] %>%
    unique()
```

```
use_genes <- intersect(use_genes, rownames(rna_data))


heatmap_data <- na.omit(as.matrix(cbind(LFCrna[use_genes,],
                                                                    LFCprot[use_genes,])))

pheatmap(heatmap_data,
                 cluster_cols = F,
                 cluster_rows = T,
                 fontsize_row = 0.1,
                 gaps_col = 5,
                 color =  heatmap_color_scale,
                 heatmap_legend_param = list(title = "logFC"),
                 main = "Proteasome complex",
                 use_raster = F)

scatterplot_rna_prot_highlight(loadings, use_genes = common_expressed, highlight_genes = use_genes, rna
```

Arp2/3 and Proteasome complex genes

Print data for shiny app

```
shiny_data_dir <- "shinyapp/shiny_data/"
dir.create(shiny_data_dir)
```

```
## Warning in dir.create(shiny_data_dir): 'shinyapp/shiny_data' already exists
```

```
# Normalized RNA-seq data
write.table(assay(norm_rna_data), paste(shiny_data_dir,"rna_exp.tsv", sep=""), sep="\t")

# Normalized protein data
write.table(mass_spec_impute, paste(shiny_data_dir,"protein_exp.tsv", sep=""), sep="\t")

# Info on which values have been imputed (matrix TRUE/FALSE)
protein_exp_is_imputed <- is.na(mass_spec[rownames(mass_spec_impute),colnames(mass_spec_impute)])
write.table(protein_exp_is_imputed, paste(shiny_data_dir,"protein_exp_is_imputed.tsv", sep=""), sep="\t

# Differential expression p-values
shiny_rna_de_res <- as.data.frame(res) %>% rownames_to_column("UniProt_ID")
write_tsv(shiny_rna_de_res, paste(shiny_data_dir,"rna_de_res.tsv", sep=""))

shiny_protein_de_res <- as.data.frame(tt_dev) %>% rownames_to_column("UniProt_ID")
write_tsv(shiny_protein_de_res, paste(shiny_data_dir,"protein_de_res.tsv", sep=""))

# Gene annotations, RNA id, protein id, gene name, description
use_genes <- union(rownames(tt_dev), rownames(res))
shiny_gene_info_tab <- gene_annot_tab %>%
    group_by(UniProt_ID) %>%
    summarise(GENE_ID = dplyr::first(GENE_ID),
                        Gene_Name = dplyr::first(Gene_Name),
```

```
                        Gene_products = dplyr::first(Gene_products)) %>%
    mutate(Gene_products = replace_na(Gene_products, "")) %>%
    filter(UniProt_ID %in% use_genes)
write.table(shiny_gene_info_tab, paste(shiny_data_dir,"gene_info.tsv", sep=""),  row.names = F, sep="\t
```

Session info

```
sessionInfo()
```

```
## R version 4.3.1 (2023-06-16)
## Platform: x86_64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.6.7
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-x86_64/Resources/lib/libRlapack.dylib;  LAPACK v
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/Stockholm
## tzcode source: internal
##
## attached base packages:
## [1] stats4    grid      stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] multcomp_1.4-25             TH.data_1.1-2
##  [3] MASS_7.3-60                 survival_3.5-7
##  [5] svglite_2.1.3               imputeLCMD_2.1
##  [7] impute_1.76.0               pcaMethods_1.94.0
##  [9] norm_1.0-11.1               tmvtnorm_1.6
## [11] gmm_1.8                     sandwich_3.1-0
## [13] Matrix_1.6-4                mvtnorm_1.2-4
## [15] lmodel2_1.7-3               ggpubr_0.6.0
## [17] ggrepel_0.9.5               scales_1.3.0
## [19] eulerr_7.0.0                patchwork_1.2.0
## [21] piano_2.18.0                snowfall_1.84-6.3
## [23] snow_0.4-4                  topGO_2.54.0
## [25] SparseM_1.81                GO.db_3.18.0
## [27] AnnotationDbi_1.64.1        graph_1.80.0
## [29] MOFA2_1.12.1                circlize_0.4.15
## [31] RColorBrewer_1.1-3          limma_3.58.1
## [33] ggfortify_0.4.16            lubridate_1.9.3
## [35] forcats_1.0.0               stringr_1.5.1
## [37] purrr_1.0.2                 readr_2.1.5
## [39] tidyr_1.3.0                 tibble_3.2.1
## [41] tidyverse_2.0.0             DT_0.31
## [43] kableExtra_1.3.4.9000       biomaRt_2.58.0
```

```
## [45] DESeq2_1.42.0               SummarizedExperiment_1.32.0
## [47] Biobase_2.62.0              MatrixGenerics_1.14.0
## [49] matrixStats_1.2.0           GenomicRanges_1.54.1
## [51] GenomeInfoDb_1.38.5         IRanges_2.36.0
## [53] S4Vectors_0.40.2            BiocGenerics_0.48.1
## [55] cowplot_1.1.2               ggplot2_3.4.4
## [57] ComplexHeatmap_2.18.0       knitr_1.45
## [59] dplyr_1.1.4
##
## loaded via a namespace (and not attached):
##   [1] bitops_1.0-7            httr_1.4.7              webshot_0.5.5
##   [4] doParallel_1.0.17       numDeriv_2016.8-1.1     tools_4.3.1
##   [7] backports_1.4.1         utf8_1.2.4              R6_2.5.1
##  [10] HDF5Array_1.30.0        mgcv_1.9-1              uwot_0.1.16
##  [13] apeglm_1.24.0           rhdf5filters_1.14.1     GetoptLong_1.0.5
##  [16] withr_2.5.2             prettyunits_1.2.0       gridExtra_2.3
##  [19] textshaping_0.3.7       cli_3.6.2               shinyjs_2.1.0
##  [22] labeling_0.4.3          slam_0.1-50             systemfonts_1.0.5
##  [25] bbmle_1.0.25.1          rstudioapi_0.15.0       RSQLite_2.3.4
##  [28] visNetwork_2.1.2        generics_0.1.3          shape_1.4.6
##  [31] gtools_3.9.5            vroom_1.6.5             car_3.1-2
##  [34] fansi_1.0.6             abind_1.4-5             lifecycle_1.0.4
##  [37] yaml_2.3.8              carData_3.0-5           gplots_3.1.3
##  [40] rhdf5_2.46.1            SparseArray_1.2.3       BiocFileCache_2.10.1
##  [43] Rtsne_0.17              blob_1.2.4              promises_1.2.1
##  [46] bdsmatrix_1.3-6         crayon_1.5.2            shinydashboard_0.7.2
##  [49] dir.expiry_1.10.0       lattice_0.22-5          KEGGREST_1.42.0
##  [52] magick_2.8.2            pillar_1.9.0            fgsea_1.28.0
##  [55] rjson_0.2.21            marray_1.80.0           codetools_0.2-19
##  [58] fastmatch_1.1-4         glue_1.7.0              data.table_1.14.10
##  [61] vctrs_0.6.5             png_0.1-8               gtable_0.3.4
##  [64] emdbook_1.3.13          cachem_1.0.8            xfun_0.41
##  [67] S4Arrays_1.2.0          mime_0.12               coda_0.19-4
##  [70] pheatmap_1.0.12         iterators_1.0.14        sets_1.0-25
##  [73] statmod_1.5.0           ellipsis_0.3.2          nlme_3.1-164
##  [76] bit64_4.0.5             progress_1.2.3          filelock_1.0.3
##  [79] KernSmooth_2.23-22      colorspace_2.1-0        DBI_1.2.0
##  [82] tidyselect_1.2.0        bit_4.0.5               compiler_4.3.1
##  [85] curl_5.2.0              rvest_1.0.3             basilisk.utils_1.14.1
##  [88] xml2_1.3.6              DelayedArray_0.28.0     caTools_1.18.2
##  [91] relations_0.6-13        rappdirs_0.3.3          digest_0.6.33
##  [94] rmarkdown_2.25          basilisk_1.14.1         XVector_0.42.0
##  [97] htmltools_0.5.7         pkgconfig_2.0.3         dbplyr_2.4.0
## [100] fastmap_1.1.1           rlang_1.1.3             GlobalOptions_0.1.2
## [103] htmlwidgets_1.6.4       shiny_1.8.0             farver_2.1.1
## [106] zoo_1.8-12              jsonlite_1.8.8          BiocParallel_1.36.0
## [109] RCurl_1.98-1.14         magrittr_2.0.3          GenomeInfoDbData_1.2.11
## [112] Rhdf5lib_1.24.1         munsell_0.5.0           Rcpp_1.0.12
## [115] reticulate_1.34.0       stringi_1.8.3           zlibbioc_1.48.0
## [118] plyr_1.8.9              parallel_4.3.1          Biostrings_2.70.1
## [121] splines_4.3.1           hms_1.1.3               polylabelr_0.2.0
## [124] locfit_1.5-9.8          igraph_1.6.0            ggsignif_0.6.4
## [127] reshape2_1.4.4          XML_3.99-0.16           evaluate_0.23
## [130] tzdb_0.4.0              foreach_1.5.2           httpuv_1.6.13
```

```
## [133] polyclip_1.10-6        clue_0.3-65             broom_1.0.5
## [136] xtable_1.8-4           rstatix_0.7.2           later_1.3.2
## [139] ragg_1.2.7             viridisLite_0.4.2       memoise_2.0.1
## [142] cluster_2.1.6          corrplot_0.92           timechange_0.2.0
```