02: Sample Clustering and Batch Effects

Tom Smith, Veronica Dezi, Manasa Ramakrishna

September 05, 2019

Contents

1. Introduction

Here we want to check for batch effects. The expectation is that the replicate number may influence protein abundance. All replicates for a given condition were run within a single TMT-plex experiment hopefully removing variation from multi-plexing and analysis by the spectrometer. BUT.....

```
suppressMessages(library(gridExtra))
source("../../CamProt_R/Utility.R")
```

2. Reading normalised data

We start by reading in the normalised RDS files generated using the first analysis notebook '1_aggregate_to_protein_an_qc.Rmd'. Remember that the data was aggregated into peptides, then into proteins, missing values were imputed using "min" and finally, samples were log centre normalised. Now we want to see if there is anything odd about the data despite us having corrected for any technical artefacts from the experimental process.

```
total_ni_protein_quant <- readRDS("../results/total_ni_res_pro_agg_norm")
oops_ni_protein_quant <- readRDS("../results/rbp_ni_res_pro_agg_norm")

total_us_protein_quant <- readRDS("../results/total_us_res_pro_agg_norm")
oops_us_protein_quant <- readRDS("../results/rbp_us_res_pro_agg_norm")
```

3. Plotting variability

We would like to see the separation of samples into their treatment groups across all 4 datasets - Total N_I, RBP N_I, Total U_S, RBP U_S. RBP = OOPS. However, no experiment is perfect so we expect some degree of variability within each treatment group across the five replicates. What we want to keep an eye out for is outlier samples within each group.

3a. Initial PCAs

We start the process by plotting PCAs - we draw them in pairs PC1&2, PC3&4, PC5&6. We do this for each of the 4 datasets. We hope that there is no pattern or separation by replicate as this would indicate batch effects. PCAs will also flag any samples that are overall quite different to their replicate samples. All the plots can be found in the folder "../plots" and have the suffix "*PCA-on-normalised-data.pdf".

```
names = c("TotalProt-Non-Insulin-vs-Insulin-Stimulated",
    "RBPs-Non-Insulin-vs-Insulin-Stimulated", "TotalProt-Unstarved-vs-Starved",
    "RBPs-Unstarved-vs-Starved")
c = 1

for (i in list(total_ni_protein_quant, oops_ni_protein_quant,
    total_us_protein_quant, oops_us_protein_quant)) {
    p = plotSamplePCA(i)
    pdf(paste("../plots/", names[c], "_PCA-on-normalised-data.pdf",
        sep = ""), paper = "a4r", width = 14, height = 8)
    plot(p$l/sum(p$l), col = "red", ylab = "Variation explained")
    plot(p$pca)
    dev.off()
    c = c + 1
}
```

1. Total Protein : Non-Insulin (4hr-Starved) vs Insulin-Stimulated (30min-Insulin)
In this case, Replicate 1 of the 4hr-Starved experiment seems to be very different to the rest of the samples and that's what PC1 is captuting (variation 70%). PC2 seems to approximately separate the two conditions (variation 8%) and PC5 seems to capture the different replicates (variation <5%). Based on this, we might want to consider removing Replicate 4, 4hr-Starved from downstream analyisis.

2. RBPs : Non-Insulin (4hr-Starved) vs Insulin-Stimulated (30min-Insulin)
Replicates 2 & 4 of the 30min-Insulin treated experiment seems to be very different to the rest of the samples and that's what PC2 is captuting (variation 28%). PC1 seems to separate Replicate 1, 30min-Insulin from all other samples (variation 38%) and there is no real separation based on replicate number even at PC6 which is good. Probably need to draw a heatmap to check what's going on with Rep2, 30-min Insulin. From looking at the raw data

3. Total Protein : Unstarved vs Insulin-Starved (4hr-Starved)
There is a lot of variation in the 4h-Starved samples (replicate 3 being the most extreme) vs the Unstarved samples (except for Replicate 1, Unstarved). PC1 is mostly captuting the differences between Starved and Unstarved samples (variation 52-53%) which is very good. PC2 (15%) and PC3 (12%) also seem to be capturing differences between starved and unstarved samples which is good news when it comes to working out what proteins are actually different between the two conditions. Finally, there is no real separation based on replicate number even at PC6 which is super.

4. RBPs : Unstarved vs Insulin-Starved (4hr-Starved)
In this case, replicate 4 of the 4hr-Starved experiment seems to be very different to the rest of the samples and that's what PC1 is captuting (variation 50%). PC2 (18% variation) and PC3 (8%) seem to be capturing differences between starved and unstarved samples which is good and there is no real separation based on replicate number even at PC6.

Overall, the condition does not appear to have a large impact on relative protein abundance. Note that this does not mean there isn't a change in absolute protein abundance. Also, if the quantification was very noisy, we would also expect this to mask the biological signal. We'll check % CVs below to assess quantification noise.

Notes:
Rayner says that all the Unstarved vs Starved samples were of higher protein concentration than the Insulin

Treated ones. 12ug for each of OOPS samples and 37ug for each of Total protein samples was loaded for TMT-plexing and mass spec. All 4 experiments labelled on the same day and processed in tandem. OOPS went in first, then loads of washes, then Total protein.

3b. Dodgey sample ? Go MAD !

Dodgey sample list is as follows
1. Replicate 1, 4hr-Starved, Total NI : Tag 126
2. Replicate 2, 30-min Insulin, RBP NI : Tag 130C
3. Replicate 4, 4hr-Starved, Total US: Tag 130C
4. Replicate 4, 4hr-Starved, RBP US: Tag 130C

We are checking median absolute deviation for samples that are problematic.To calculate MAD,
1. take the median across all proteins/peptides for a sample
2. subtract each value from the median to get deviations
3. take the absolute value of the deviations (remove +/- signs so all of them are positive)
4. take the median of the deviations

A low MAD says that the values of expression in the sample are very similar or don't deviate very much from the median i.e the dynamic range of protein expression values is low. Since a median is less affected by outliers, this gives us a more robust estimation of variation within a sample.

```
total_ni_res_pro_agg = readRDS("../results/total_ni_res_pro_agg.rds")
rbp_ni_res_pro_agg = readRDS("../results/rbp_ni_res_pro_agg.rds")
total_us_res_pro_agg = readRDS("../results/total_us_res_pro_agg.rds")
rbp_us_res_pro_agg = readRDS("../results/rbp_us_res_pro_agg.rds")

# N_I
apply(exprs(total_ni_res_pro_agg), 2, function(x) mad(x,
    constant = 1))
```

```
##    126   127N   127C   128N   128C   129N   129C   130N   130C    131
##  8.500 11.175 12.000 14.750 12.800 13.700 13.900 12.100 11.200 13.100
```

```
apply(exprs(rbp_ni_res_pro_agg), 2, function(x) mad(x,
    constant = 1))
```

```
##   126  127N  127C  128N  128C  129N  129C  130N  130C   131
## 15.95 18.30 12.20 13.90 14.10 17.10  7.40 15.80  6.90  9.55
```

```
# U_S
apply(exprs(total_us_res_pro_agg), 2, function(x) mad(x,
    constant = 1))
```

```
##   126  127N  127C  128N  128C  129N  129C  130N  130C   131
## 12.10 14.20 18.90 13.60 14.90 13.15  9.75 10.70  5.40 13.30
```

```
apply(exprs(rbp_us_res_pro_agg), 2, function(x) mad(x,
    constant = 1))
```

```
##    126   127N   127C   128N   128C   129N   129C   130N   130C    131
##  7.600 15.400 33.150 14.600 21.775 18.425  5.200  5.675  3.100 10.700
```

3c. PCAs after dodgey samples being removed

Looking at the data values, the samples we think are problematic have very low Median Absolute Deviation values relative to their replicate friends. Interestingly, tags 126 and 130C consistently have low MADs

irrespective of the experiment. For the first experiment, Tag 126 has the least MAD, for all other experiments, tag 130C seems to be the culprit. For now, we will remove those samples with the smallest mad in each of the experiments. Again, these PCAs can be found in the folder "../plots/" and have the suffix "PCA-on-normalised-data-dodgey-samples-removed.pdf".

```
# Removing dodgey samples
total_ni_protein_quant_nd = total_ni_protein_quant[,
    grep("126", sampleNames(total_ni_protein_quant),
        invert = T)]
oops_ni_protein_quant_nd = oops_ni_protein_quant[,
    grep("130C", sampleNames(oops_ni_protein_quant),
        invert = T)]
total_us_protein_quant_nd = total_us_protein_quant[,
    grep("130C", sampleNames(total_us_protein_quant),
        invert = T)]
oops_us_protein_quant_nd = oops_us_protein_quant[,
    grep("130C", sampleNames(oops_us_protein_quant),
        invert = T)]

names = c("TotalProt-Non-Insulin-vs-Insulin-Stimulated",
    "RBPs-Non-Insulin-vs-Insulin-Stimulated", "TotalProt-Unstarved-vs-Starved",
    "RBPs-Unstarved-vs-Starved")
c = 1

for (i in list(total_ni_protein_quant_nd, oops_ni_protein_quant_nd,
    total_us_protein_quant_nd, oops_us_protein_quant_nd)) {
    p = plotSamplePCA(i)
    pdf(paste("../plots/", names[c], "_PCA-on-normalised-data-dodgey-samples-removed.pdf",
        sep = ""), paper = "a4r", width = 14, height = 8)
    plot(p$l/sum(p$l), col = "red", ylab = "Variation explained")
    plot(p$pca)
    dev.off()
    c = c + 1
}

saveRDS(total_ni_protein_quant_nd, file = "../results/total_ni_res_pro_agg_norm_nododgey.rds")
saveRDS(oops_ni_protein_quant_nd, file = "../results/rbp_ni_res_pro_agg_norm_nododgey.rds")
saveRDS(total_us_protein_quant_nd, file = "../results/total_us_res_pro_agg_norm_nododgey.rds")
saveRDS(oops_us_protein_quant_nd, file = "../results/rbp_us_res_pro_agg_norm_nododgey.rds")
```

4. Combining all datasets

This won't be used downstream but is done to check whether there is clear separation between the 4 experiments conducted in the study. To do so, we focus only on those proteins that are present in all 4 treatments/conditions and combine expression from all 4 experiments for these 638 proteins.

```
# Define intersecting proteins

# For NI experiment
intersecting_ni_proteins <- intersect(rownames(total_ni_protein_quant),
    rownames(oops_ni_protein_quant))
print(length(intersecting_ni_proteins))
```

```
## [1] 1100
```

```
# For US experiment
intersecting_us_proteins <- intersect(rownames(total_us_protein_quant),
    rownames(oops_us_protein_quant))
print(length(intersecting_us_proteins))
```

```
## [1] 730
```

```
# For both experiments
intersecting_all_proteins <- intersect(intersecting_ni_proteins,
    intersecting_us_proteins)
print(length(intersecting_all_proteins))
```

```
## [1] 638
```

```
# Combining data across all experiments
total_ni_exprs <- exprs(total_ni_protein_quant_nd[intersecting_all_proteins,
    ])
colnames(total_ni_exprs) <- paste0("Total-NI_", pData(total_ni_protein_quant_nd)$Sample_name)

oops_ni_exprs <- exprs(oops_ni_protein_quant_nd[intersecting_all_proteins,
    ])
colnames(oops_ni_exprs) <- paste0("RBP-NI_", pData(oops_ni_protein_quant_nd)$Sample_name)

total_us_exprs <- exprs(total_us_protein_quant_nd[intersecting_all_proteins,
    ])
colnames(total_us_exprs) <- paste0("Total-US_", pData(total_us_protein_quant_nd)$Sample_name)

oops_us_exprs <- exprs(oops_us_protein_quant_nd[intersecting_all_proteins,
    ])
colnames(oops_us_exprs) <- paste0("RBP-US_", pData(oops_us_protein_quant_nd)$Sample_name)

# Mega expression matrix
combined_exprs <- cbind(total_ni_exprs, oops_ni_exprs,
    total_us_exprs, oops_us_exprs)
```

4a. Mega PCA across all samples

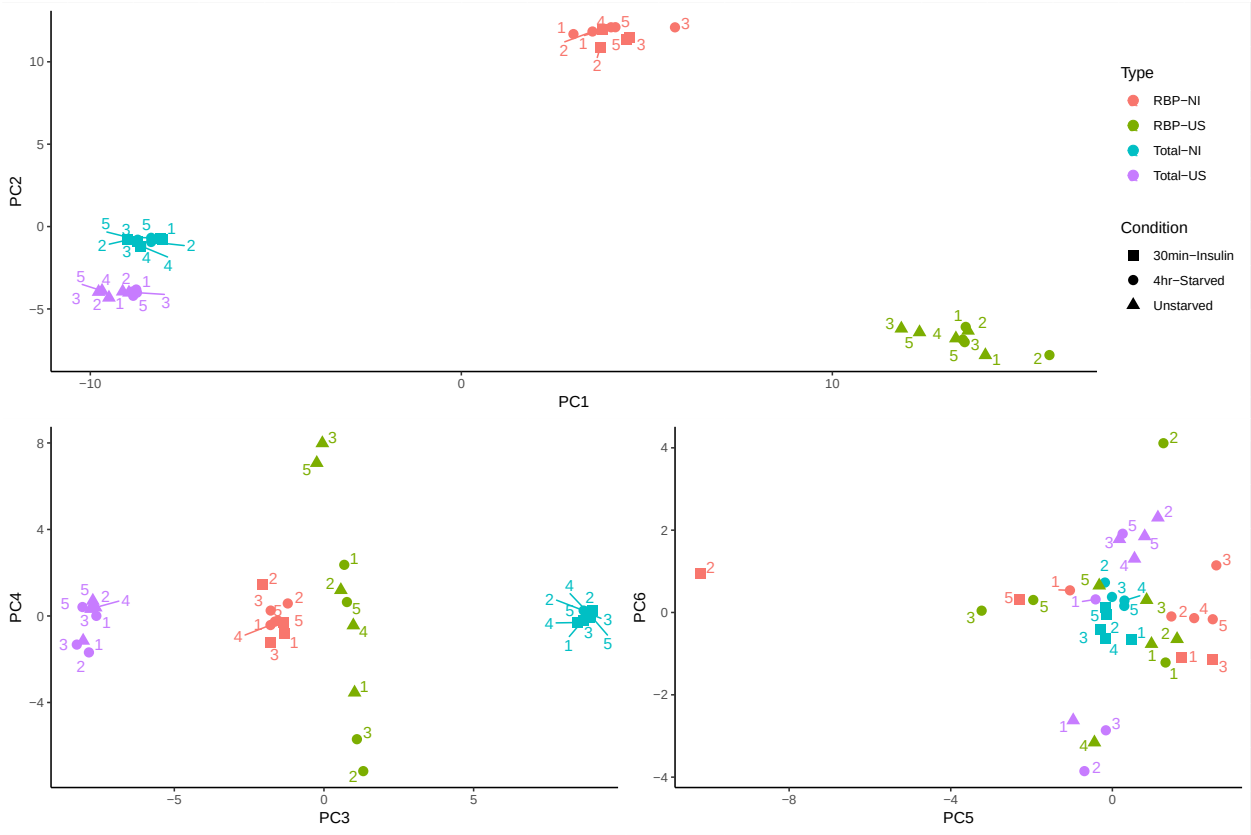
We plot PC1 vs PC2, PC3 vs PC4 and PC5 vs PC6 for these 20 samples below without the dodgey ones from above.Then we remake the plot having removed what we call the "dodgey" samples as defined above.

```
# Mega PCA of all samples
pca <- prcomp(t(combined_exprs))
loadings <- pca$sdev^2

# Annotate samples to enable PCA labelling
projections <- data.frame(pca$x) %>% tibble::rownames_to_column("sample") %>%
    separate(sample, into = c("Type", "Condition",
        "Replicate"), sep = "_")

# Plotting the PCA with labels so we can tell which
# samples are most variable.
library(ggrepel)
p <- projections %>% ggplot(aes(PC1, PC2, shape = Condition,
    colour = Type)) + geom_point(size = 3) + geom_text_repel(aes(label = Replicate)) +
    scale_shape_manual(values = c(15, 16, 17, 18)) +
```

```
    theme_classic()

p2 <- p + aes(PC3, PC4) + theme(legend.position = "none")
p3 <- p + aes(PC5, PC6) + theme(legend.position = "none")

pdf("../plots/Mega-PCA-of-all-20-samples-4-conditions-4-replicates-no-dodgey-samples.pdf",
    paper = "a4r", width = 14, height = 8)
plot(loadings/sum(loadings))
pt = arrangeGrob(p, p2, p3, nrow = 2, layout_matrix = rbind(c(1,
    1), c(2, 3)))
plot(pt)
dev.off()
```

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

```
plot(pt)
```



Looking at the PCA plots, albeit using a subset of the total data, overall, the experiment seems to have worked pretty well. PC1 separates the Total from the OOPS/RBP samples very very clearly. PC2 seems to separate the 4 protein extractions - Total NI, Total US, RBP NI and RBP US. PC3 does thsi separation better than PC2. Within the first three components, we capture 40% (PC1) + 25% (PC2) + 18% (PC3) = 83% of the variation in the data arising mainly from the fact that they are Total vs RBP samples and also protein extractions. The intra-condition variability is very very low i.e the replicates, for the most part, cluster tightly together with the exception of the RBP-US samples.

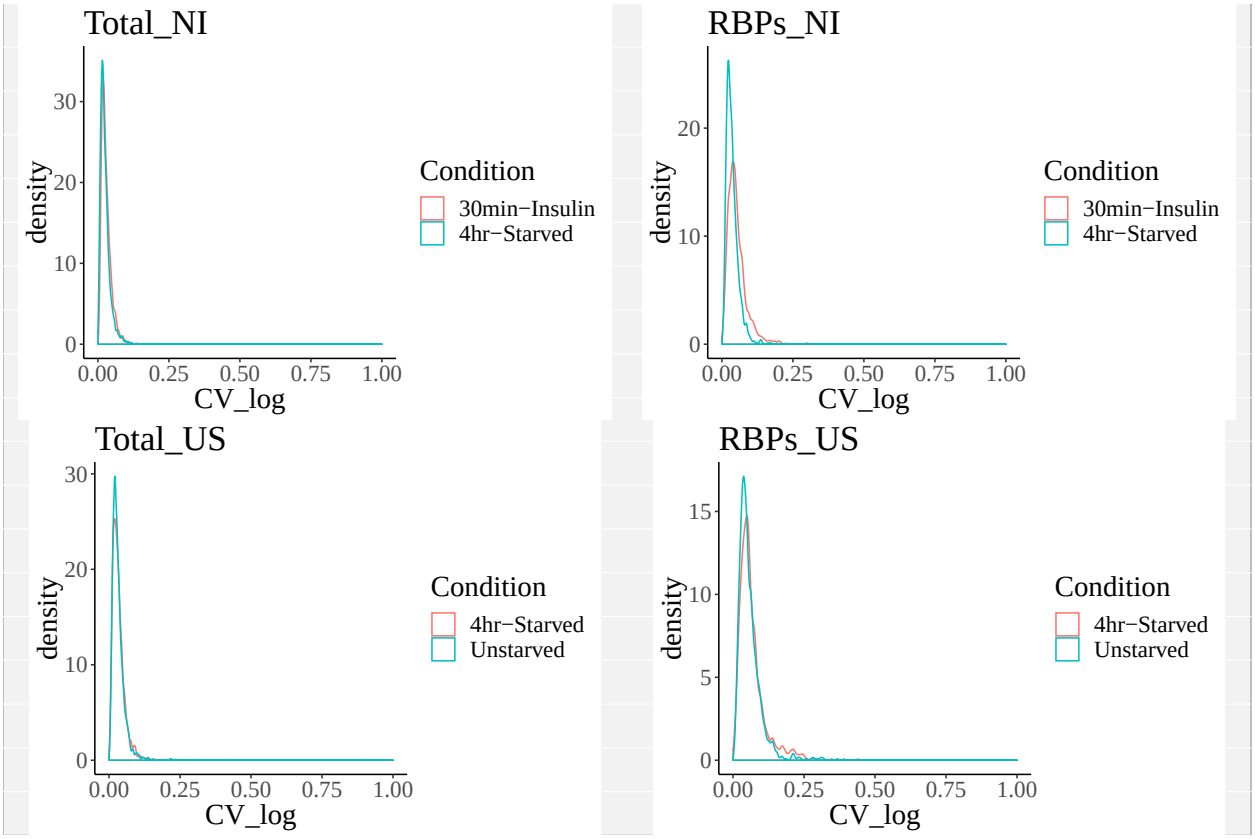
4b. Coefficient of variation

Below, we plot the distributions of coefficient of variance for each of the datasets. This is just an additional QC step that the data all looks OK and that there is no big difference in the variance within a condition since this could invalidate the assumptions of heteroscedasticity (same variance per group) that we will make in the modeling of protien abundance later. Note: MR, 02/09/2019. Don't fully understand this.

```
p_ni_total <- plotCVs(total_ni_protein_quant_nd) +
    ggtitle("Total_NI") + xlim(0, 1)
# print(p_ni_total) print(p_ni_total + aes(CV_log))

p_ni_oops <- plotCVs(oops_ni_protein_quant_nd) + ggtitle("RBPs_NI") +
    xlim(0, 1)
# print(p_ni_oops) print(p_ni_oops + aes(CV_log))

p_us_total <- plotCVs(total_us_protein_quant_nd) +
    ggtitle("Total_US") + xlim(0, 1)
# print(p_us_total) print(p_us_total + aes(CV_log))

p_us_oops <- plotCVs(oops_us_protein_quant_nd) + ggtitle("RBPs_US") +
    xlim(0, 1)
# print(p_us_oops) print(p_us_oops + aes(CV_log))

g = arrangeGrob(p_ni_total + aes(CV_log), p_ni_oops +
    aes(CV_log), p_us_total + aes(CV_log), p_us_oops +
    aes(CV_log), nrow = 2)
pdf("../plots/CV-plots-all-4-experiments-no-dodgey-samples.pdf",
    paper = "a4r", width = 14, height = 8)
plot(g)
dev.off()
```

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

```
plot(g)
```



The CVs look as good as we would expect for TMT. 50% of the CVs should be < 0.2 after normalisation - see here. Also the OOPS data is not notably more variable which is good to see.