LFQ comparison

True Ratios

The 48 UPS proteins (Sigma Aldrich) were spiked into an *E. coli* background. UPS1 has all 48 UPS proteins at the same concentration, while UPS2 has varying concentrations. We define the ratio as UPS2 / UPS1, so the true ratios run from 10 to 10^{-4} – in addition, 8 proteins have the same concentration in UPS2 and UPS1. The proteins and their associated UPS2/UPS1 ratios are defined below:

```
ups_protein_ids <- c(
    #ratio of 10
    "P00915", "P00918", "P01031", "P69905", "P68871", "P41159", "P02768", "P62988",
    # ratio of 1
    "P04040", "P00167", "P01133", "P02144", "P15559", "P62937", "Q06830", "P63165",
    # ratio of 0.1
    "P00709", "P06732", "P12081", "P61626", "Q15843", "P02753", "P16083", "P63279",
    # ratio of 0.01
    "P01008", "P61769", "P55957", "O76070","P08263","P01344","P01127","P10599",
    # ratio of 0.001
    "P99999","P06396","P09211","P01112","P01579","P02787","O00762","P51965",
    # ratio of 0.0001
    "P08758","P02741","P05413","P10145","P02788","P10636","P00441","P01375"
)

ups_true_df <- data.frame(prot = ups_protein_ids,
                          ratios = rep(
                            sapply(1:-4, function(factor){
                              return(10^factor)
                            }),
                            each = 8),
                          stringsAsFactors = FALSE)

# find the protein ids for proteins that are constant
ups_non_de <- filter(ups_true_df, ratios == 1)$prot
```

Quantification Results

We analyze 6 different result files on the peptide level, and 2 different result files on the protein level. The peptide files are as follows:

- moFF's peptide summary, produced from MaxQuant's `msms.txt` - run with/without MBR (2 files)
- FlashLFQ's peptide file: `msms_FlashLFQ_QuantifiedBaseSequences.tsv` - produced from MaxQuant's `msms.txt` and run with/without MBR (2 files)
- MaxQuant's peptide file: `peptides.txt`, from MaxQuant, with/without MBR (2 files)

The 2 protein files are MaxQuant's protein report (`proteinGroups.txt`), from MaxQuant run with/without MBR.

In the following, we read in all of the results files. In most cases, the protein IDs are in a list, separated by ";", so we use the function `get_protein()` to get the first protein in the list.

```
get_protein <- function(prot_vec){ sapply(prot_vec, function ( row ) { return(str_extract(row, "[A-Z0-9
```

We don't want any character columns to be read as factors:

```
options(stringsAsFactors = FALSE)
```

Let's also set up a function for getting the filename using a regular expression:

```
root <- "Jan_Quant_Paper_revised"
get_filename <- function(root, directory, regex){
  directory <- paste0("data/", root, directory)
  filename <- paste0(directory, list.files(directory, pattern=regex))
  print(filename)
  return(filename)
}
```

FlashLFQ (no normalization)

```
flash <- read.delim(get_filename(root, "/UPS/FLASHLFQ/MBR_without_norm/", ".+QuantifiedPeptides.tsv].tal
                    stringsAsFactors = FALSE) %>%
    rename(peptide = Sequence, prot = Protein.Groups) %>%
    select(peptide, prot, starts_with("Intensity"))
```

```
## [1] "data/Jan_Quant_Paper_revised/UPS/FLASHLFQ/MBR_without_norm/Galaxy339-[FlashLFQ_on_data_20,_data_
```

```
flash$prot <- get_protein(flash$prot)
flash_names <- names(flash)
flash$peptide <- gsub("<cmm>","",
                      gsub("<ox>","",
                           gsub("NH2-","", gsub("-COOH","", flash$peptide))))
#flash

# flash no mbr
flash_no_mbr <- read.delim(get_filename(root, "/UPS/FLASHLFQ/noMBR_without_norm/", ".+QuantifiedPeptides
                           stringsAsFactors = FALSE) %>%
    rename(peptide = Sequence, prot = Protein.Groups) %>%
    select(peptide, prot, starts_with("Intensity"))
```

```
## [1] "data/Jan_Quant_Paper_revised/UPS/FLASHLFQ/noMBR_without_norm/Galaxy351-[FlashLFQ_on_data_20,_dat
```

```
flash_no_mbr$prot <- get_protein(flash_no_mbr$prot)
flash_no_mbr$peptide <- gsub("<cmm>","",
                      gsub("<ox>","",
                           gsub("NH2-","", gsub("-COOH","", flash_no_mbr$peptide))))
#flash_no_mbr
```

FlashLFQ (with normalization)

```
flash_norm <- read.delim(get_filename(root, "/UPS/FLASHLFQ/MBR_with_norm/",
                                      ".+BayesianFoldChangeAnalysis.tsv].tabular"),
                         stringsAsFactors = FALSE) %>%
    rename(prot = Protein.Group)# %>%
```

```
## [1] "data/Jan_Quant_Paper_revised/UPS/FLASHLFQ/MBR_with_norm/Galaxy335-[FlashLFQ_on_data_20,_data_13
```

```
    #select(prot, Protein.Intensity.for.Treatment.Condition)
flash_norm$prot <- get_protein(flash_norm$prot)
#flash_names <- names(flash)
#flash_norm

# flash no mbr
flash_no_mbr_norm <- read.delim(get_filename(root, "/UPS/FLASHLFQ/noMBR_with_norm/",
                                             ".+BayesianFoldChangeAnalysis.tsv].tabular"),
                                stringsAsFactors = FALSE) %>%
    rename(prot = Protein.Group)
```

```
## [1] "data/Jan_Quant_Paper_revised/UPS/FLASHLFQ/noMBR_with_norm/Galaxy347-[FlashLFQ_on_data_20,_data_
```

```
flash_no_mbr_norm$prot <- get_protein(flash_no_mbr_norm$prot)
#flash_no_mbr_norm
```

moFF

```
moff <- read.delim(get_filename(root, "/UPS/moff/withMBR/", ".+peptide_summary].tabular"),
                   stringsAsFactors = FALSE)
```

```
## [1] "data/Jan_Quant_Paper_revised/UPS/moff/withMBR/Galaxy356-[moFF_on_data_133,_data_132,_and_others
```

```
# replace protein list with first protein
moff$prot <- get_protein(moff$prot)
moff_names <- names(moff)
moff <- moff[,order(colnames(moff))]
#moff
```

When moFF is run without MBR, we need to combine all of the individual peptide summaries, which we do with `join_all` from `plyr`.

```
# moff, no mbr
moff_files <- list.files(paste0('data/',root, '/UPS/moff/withoutMBR/peptide_summary/'),
                         full.names = TRUE)
#tail(unlist(str_split(moff_files[1], "/")), n=1)

moff_no_mbr <- lapply(moff_files, function(i) df = read.delim(i, as.is = TRUE) %>%
  rename(!!paste("sumIntensity_20130510_EXQ1_IgPa_QC_",
                 str_extract(i, "UPS[0-9_]+"), sep="") := sumIntensity_UPS_PSM_All_edited)) %>%
                       join_all(by=c("peptide", "prot"), type = "full")
moff_no_mbr$prot <- get_protein(moff_no_mbr$prot)
#moff_no_mbr
```

MaxQuant

Peptides

```
mq <- read.delim(paste0('data/', root, "/UPS/MAXQUANT/MQ_MBR/txt/peptides.txt"),
                 stringsAsFactors = FALSE) %>%
    rename(peptide = Sequence, prot = Proteins) %>%
    select(peptide, prot, starts_with("Intensity."))
mq$prot <- get_protein(mq$prot)
mq_names <- names(mq)
#mq
# mq no mbr
mq_no_mbr <- read.delim(paste0('data/', root, "/UPS/MAXQUANT/MQ_NoMBR/txt/peptides.txt"),
                        stringsAsFactors = FALSE) %>%
    rename(peptide = Sequence, prot = Proteins) %>%
    select(peptide, prot, starts_with("Intensity."))
mq_no_mbr$prot <- get_protein(mq_no_mbr$prot)
#mq_no_mbr
```

Proteins

With the MaxQuant protein reports, we are going to do a t-test on each row of the data matrix, rather than using `PECA` as for the peptide-level analysis. Therefore, we filter the results to keep only the proteins that had values in 3 out of 4 replicates for both groups, which is the minimum necessary for a robust T-test.

```
mq_prot_analysis <- function(proteinGroups, name){
    #don't remove contaminants, because some UPS proteins are potential contam
    mq_prot <- read.delim(proteinGroups,
                      na.strings = c("0", "", "NA"),
                      stringsAsFactors = FALSE) %>%
        filter(is.na(Reverse)) %>% # if not decoy, is na
        mutate(prot = get_protein(Protein.IDs),
               ups_prot_id = str_split(prot, "ups\\|", simplify = TRUE)[, 1]) %>%
        mutate(ups = prot %in% ups_protein_ids, de = ups & !(prot %in% ups_non_de)) %>%
        select(prot, ups, de, starts_with("LFQ"), ups_prot_id)

    # filter to 3 or more observations
    mq_prot$enough_obs <- rowSums(is.na(mq_prot[4:7])) <= 1 &
        rowSums(is.na(mq_prot[8:11])) <= 1
    mq_prot_filt <- filter(mq_prot, enough_obs) %>% select(-enough_obs)

    # t testing
    mq_ts <- rep(0, nrow(mq_prot_filt))
    for (i in 1:nrow(mq_prot_filt)){
        mq_ts[i] <- t.test(x = mq_prot_filt[i, 4:7], y = mq_prot_filt[i, 8:11])$p.value
    }
    mq_prot_filt$p.fdr <- p.adjust(mq_ts, method = "fdr")
    mq_prot_filt$sr <- rowMeans(mq_prot_filt[, 8:11], na.rm = TRUE)/
        rowMeans(mq_prot_filt[, 4:7], na.rm = TRUE)
    # mq_roc <- roc(mq_prot_filt$de, mq_prot_filt$p.fdr)
    joined_mq <- filter(mq_prot_filt, ups) %>%
        left_join(ups_true_df, by = c("ups_prot_id" = "prot"))
    rmsle <- sqrt(mean((log10(joined_mq$sr) - log10(joined_mq$ratios))^2, na.rm = TRUE))
```

```
    mq_data <- data.frame("norm_methods" = "MaxLFQ", "quant_methods" = name, rmsle = rmsle)
    list("full_df" = mq_prot_filt, "rmlse" = mq_data, "joined_mq" = joined_mq)
}

# mq prot
mq_data <- mq_prot_analysis(paste0('data/', root, "/UPS/MAXQUANT/MQ_MBR/txt/proteinGroups.txt"), "MaxQua

# mq prot, no mbr
mq_no_mbr_data <- mq_prot_analysis(paste0('data/', root, "/UPS/MAXQUANT/MQ_NoMBR/txt/proteinGroups.txt"
```

Differential expression analysis and ratio estimation

Below is the function that is used on all of the peptide reports to normalize and then test for differential expression. The R/Bioconductor package `limma` is used for normalization, and the package `PECA` is used to "roll-up" from peptides to proteins and to test for differential expression (using a modified t-test). Then, `roc` from the `pROC` package is used to create a ROC curve, from which the AUC is estimated.

```
peptides_normalize_and_test_de <- function(df, quant_method, int_col_vec, id_col_name,
                                           norm_method, grp1_col_name, grp2_col_name){
  if (str_detect(quant_method, "norm")){
    df_peca <- df %>% rename(p = False.Discovery.Rate,
                             slr = Protein.Log2.Fold.Change)
  } else {
    if (norm_method == "vsn"){
      intensities <- 2^limma::normalizeVSN(as.matrix(df[, int_col_vec]))
    } else if (norm_method != 'NA') {
      intensities <- 2^(limma::normalizeBetweenArrays(log2(as.matrix(df[, int_col_vec])), method = norm
    } else {
      intensities <- as.matrix(df[, int_col_vec])
    }

    df_norm <- data.frame(prot = df$prot, intensities, stringsAsFactors = FALSE)
    df_peca <- PECA::PECA_df(df_norm, id = id_col_name, samplenames1 = grp1_col_name, samplenames2 = grp
                             test = "modt")
    df_peca$prot <- rownames(df_peca)
  }

  # ups proteins
  df_peca$ups <- as.numeric(df_peca$prot %in% ups_protein_ids)
  # ups proteins that are not DE
  for (i in 1:nrow(df_peca)){
    df_peca$de[i] <- ifelse(df_peca$prot[i] %in% ups_non_de, 0, df_peca$ups[i])
  }

  df_peca_ratio <- df_peca %>%
    filter(ups == 1) %>%
    left_join(ups_true_df, by = "prot") %>%
    mutate(sr = 2^slr) # peca returns log2 fold change
  sqr_err <- (log10(df_peca_ratio$sr) - log10(df_peca_ratio$ratios))^2
  rmsle <- sqrt(mean(sqr_err, na.rm = TRUE))

  return(list("peca" = df_peca_ratio,
```

```
              "full_peca" = df_peca,
              "rmsle" = rmsle))
}
```

```
norm_methods <- c("cyclicloess", "scale", "quantile", "vsn")
quant_methods <- c("moFF",  "moFF_no_MBR", "FlashLFQ",
                   "FlashLFQ_no_MBR", "MaxQuant", "MaxQuant_no_MBR")
quants <- list("moFF" = moff,  "moFF_no_MBR" = moff_no_mbr,
               "FlashLFQ" = flash, "FlashLFQ_no_MBR" = flash_no_mbr,
               "MaxQuant" = mq, "MaxQuant_no_MBR" = mq_no_mbr,
               "FlashLFQ_norm" = flash_norm, "FlashLFQ_no_MBR_norm" = flash_no_mbr_norm)
method_list <- expand.grid(list(norm_methods = norm_methods, quant_methods = quant_methods), stringsAsFa
method_list <- rbind(method_list,
                     c('NA', 'FlashLFQ_norm'),
                     c('NA', 'FlashLFQ_no_MBR_norm'))
results <- vector(length = nrow(method_list), mode = "list")
# aucs <- rep(0, nrow(method_list))
rmses <- rep(0, nrow(method_list))
for (i in 1:nrow(method_list)){
   if (i != 14){
    df <- quants[[method_list$quant_methods[i]]]
    names_df <- names(df)
    ith_result <- peptides_normalize_and_test_de(df,
                                                 quant_method = method_list$quant_methods[i],
                                                 int_col_vec = 3:10,
                                                 id_col_name = 'prot',
                                                 norm_method =  method_list$norm_methods[i],
                                                 grp1_col_name = names_df[7:10],
                                                 grp2_col_name = names_df[3:6])
    results[[i]] <- ith_result
    # aucs[i] <- ith_result$roc$auc
    rmses[i] <- ith_result$rmsle
   }
}
all_df <- cbind(method_list, rmsle = rmses)
```

Results

RMSLE

Combined Levels

```
fc_accuracy <- rbind(all_df, mq_data$rmlse, mq_no_mbr_data$rmlse) %>%
    rename("Normalization" = "norm_methods",
           "Tool" = "quant_methods")
fc_accuracy
```

```
##    Normalization               Tool     rmsle
## 1    cyclicloess               moFF 1.4190035
## 2          scale               moFF 2.7693164
```

```
## 3       quantile                 moFF 3.3490518
## 4            vsn                 moFF 2.7600588
## 5    cyclicloess          moFF_no_MBR 1.7468269
## 6          scale          moFF_no_MBR 1.7530628
## 7       quantile          moFF_no_MBR 1.7620451
## 8            vsn          moFF_no_MBR 1.7471202
## 9    cyclicloess             FlashLFQ 1.4414896
## 10         scale             FlashLFQ 1.6727528
## 11      quantile             FlashLFQ 1.7704304
## 12           vsn             FlashLFQ 1.6611123
## 13   cyclicloess      FlashLFQ_no_MBR 0.3955577
## 14         scale      FlashLFQ_no_MBR 0.0000000
## 15      quantile      FlashLFQ_no_MBR 3.4003876
## 16           vsn      FlashLFQ_no_MBR 2.4845968
## 17   cyclicloess              MaxQuant 0.5695361
## 18         scale              MaxQuant 3.5071639
## 19      quantile              MaxQuant 4.0913438
## 20           vsn              MaxQuant 3.5003773
## 21   cyclicloess      MaxQuant_no_MBR 0.3870962
## 22         scale      MaxQuant_no_MBR 4.1826888
## 23      quantile      MaxQuant_no_MBR 4.3085499
## 24           vsn      MaxQuant_no_MBR 4.1651244
## 25            NA        FlashLFQ_norm 1.4412886
## 26            NA FlashLFQ_no_MBR_norm 2.1015551
## 27        MaxLFQ              MaxQuant 0.5192335
## 28        MaxLFQ      MaxQuant_no_MBR 0.4988500
```

```
write.table(fc_accuracy, file = "results/fold_change_accuracy.tabular",
            quote = FALSE,
            row.names = FALSE,
            sep = '\t')
```

Individual ratio levels

```
all_himedlo <- data.frame(t(sapply(c(1:(length(results) + 1)), function(i){
  if (i >= length(results)){
    norm_method <- "MaxLFQ"
    if (i == length(results)){
      quant_method <- "MaxQuant"
      peca_loc <- mq_data$joined_mq
    } else {
      quant_method <- "MaxQuant_no_MBR"
      peca_loc <- mq_no_mbr_data$joined_mq
    }
  } else {
    if (i > 13){ i <- i + 1 }
    norm_method <- method_list$norm_methods[i]
    quant_method <- method_list$quant_methods[i]
    peca_loc <- results[[i]]$peca
  }
  results <- sapply(c(1:-4), function(exp){
    de_level <- peca_loc %>% filter(ratios == 10^exp)
```

```
    num <- nrow(filter(de_level, !is.na(sr)))
    rsmle <- sqrt(mean((log10(de_level$ratios) - log10(de_level$sr))^2, na.rm = TRUE))
    return(c(rsmle,num))
  })
  return(c(norm_method, quant_method, results))
})))
headers <- unlist(sapply(1:-4, function(exp){
  ratio <- as.character(10^exp)
  return(list(ratio, paste0(ratio, 'n')))
}))
colnames(all_himedlo) <- c('norm_methods', 'quant_methods', headers)
all_himedlo
```

```
##    norm_methods        quant_methods                10 10n
## 1   cyclicloess                 moFF 0.499765780595977   8
## 2         scale                 moFF 0.655842551928926   8
## 3      quantile                 moFF 0.478242868479334   8
## 4           vsn                 moFF 0.632327080781435   8
## 5   cyclicloess          moFF_no_MBR 0.782482582832545   8
## 6         scale          moFF_no_MBR 0.795518057707186   8
## 7      quantile          moFF_no_MBR 0.752062708350108   8
## 8           vsn          moFF_no_MBR 0.785944936845805   8
## 9   cyclicloess             FlashLFQ 0.510888811941642   8
## 10        scale             FlashLFQ 0.460279415152784   8
## 11     quantile             FlashLFQ 0.382316120268724   8
## 12          vsn             FlashLFQ  0.43441976401007   8
## 13  cyclicloess      FlashLFQ_no_MBR 0.560916920452103   8
## 14     quantile      FlashLFQ_no_MBR 0.568717877961413   8
## 15          vsn      FlashLFQ_no_MBR 0.584635380266367   8
## 16  cyclicloess             MaxQuant 0.498700553312687   8
## 17        scale             MaxQuant 0.485212551246664   8
## 18     quantile             MaxQuant 0.490694683905359   8
## 19          vsn             MaxQuant 0.464013913577068   8
## 20  cyclicloess      MaxQuant_no_MBR 0.504255902012661   8
## 21        scale      MaxQuant_no_MBR 0.503561960856652   8
## 22     quantile      MaxQuant_no_MBR 0.446634191599251   8
## 23          vsn      MaxQuant_no_MBR 0.456725699795152   8
## 24           NA        FlashLFQ_norm 0.493216344171544   8
## 25           NA FlashLFQ_no_MBR_norm  0.53482353846093   8
## 26       MaxLFQ             MaxQuant 0.478157078789622   8
## 27       MaxLFQ      MaxQuant_no_MBR 0.495311512147928   8
##                    1 1n               0.1 0.1n              0.01 0.01n
## 1  0.278470216863194  8 0.312019535251112    8  1.53376765739783     3
## 2  0.651853286716037  8   2.1857165647639    8  4.05133310173945     8
## 3  0.386321838719755  8  3.61223377949582    8    5.058370452265     8
## 4   0.64985434365947  8  2.15992593974206    8  4.05088698109852     8
## 5  0.461463469463412  8 0.478077123260117    8  1.30371523995456     8
## 6  0.467533219802104  8 0.480578313996916    8   1.2971676222551     8
## 7  0.411746387119096  8 0.508702860018184    8  1.31646775933973     8
## 8  0.461554632708126  8 0.485696471150098    8  1.28783366072801     8
## 9  0.268328462623579  8 0.243865594220424    8 0.275595546147071     8
## 10 0.352860328427553  8    0.937015026483    8   2.1278335345936     8
## 11 0.304443871231701  8 0.881100658741605    8  2.36700900200388     8
```

```
## 12 0.328269476874641  8 0.921936357161216    8  2.11317432248027     8
## 13 0.262158217376765  8 0.250137343378881    6               NaN     0
## 14   1.1142250947294  8   3.8136608485023    8  5.31864180709647     8
## 15  1.07208422751307  8  2.50045975425825    8  3.99646004697719     8
## 16 0.273537369125465  8 0.310840646723892    8 0.346259696496987     2
## 17 0.372441069624199  8  2.95227063328949    8  4.79448164724293     8
## 18 0.483308646752027  8  3.90664407319439    8  5.91256435039776     8
## 19 0.352684572364788  8  2.92879696147734    8  4.78346225593294     8
## 20 0.294601722093659  8 0.320242486460555    7               NaN     0
## 21 0.859788740870613  8  4.66540646777645    8  6.10198840451231     8
## 22  0.31787314583878  8  4.13320682048537    8  6.48954526261285     8
## 23 0.837089152889346  8  4.65510681987377    8  6.07854508337686     8
## 24 0.304430527336509  8 0.179281766173467    8 0.376229145394222     8
## 25 0.274014177199538  8 0.195990724553794    8  1.42929416713932     8
## 26 0.278466653906413  8  0.32759604142564    8   0.8296029789004     6
## 27 0.284955058356451  8  0.38547294684958    8  1.15523947614295     2
##                0.001 0.001n            1e-04 1e-04n
## 1   2.02849300967587      1 3.22112982562865      5
## 2   3.78602403635763      8 3.10403005963744      8
## 3   4.08481906932269      8 3.40555974785365      8
## 4   3.77370230346755      8 3.09347789839291      8
## 5    2.2972815849169      8 3.20584832311987      8
## 6   2.29488213072729      8 3.22612080107838      8
## 7    2.3363435640621      8 3.23161220918535      8
## 8   2.28860563210295      8 3.21740559596843      8
## 9   2.28984127609049      7 2.97360867163536      6
## 10  2.89923335931309      8 1.62512067753862      8
## 11   2.9161959134172      8 1.91948505285709      8
## 12  2.88547654199212      8 1.61797260666961      8
## 13               NaN      0              NaN      0
## 14  3.95059994050211      8 3.06131945253642      8
## 15  2.43069777246912      8 2.72321129361047      8
## 16               NaN      0 2.27091651122646      1
## 17  4.80300495055989      8 4.31917052208521      8
## 18  5.28678933112306      8  4.6679512909205      8
## 19  4.80638051375504      8 4.31456806147413      8
## 20               NaN      0              NaN      0
## 21  4.94537890093568      8 4.52983988090323      8
## 22  5.38148269798933      8 4.78781095935231      8
## 23   4.9259449257491      8 4.50517367370035      8
## 24  2.24501643611551      8 2.62947662004302      8
## 25                 3      8 3.88030775692923      8
## 26 0.522107155010824      2 0.65633642178688      4
## 27               NaN      0              NaN      0
```

```
write.table(all_himedlo, file="results/high_low_fold_change_accuracy_new.tab",
            sep="\t", quote=FALSE, row.names=FALSE)
```