Parse PSMs

true

09 November, 2020

Abstract

Here, we parse the PSM-level PD output

Load libraries

```r
#### Load packages ####
library(camprotR)
library(tidyverse)
library(MSnbase)
```

Read in PSM data

```r
infiles <- Sys.glob('../raw/*gz')
names(infiles) <- gsub('_PSMs_txt.gz', '', basename(infiles))
psm <- infiles %>% lapply(read.delim)
print(infiles)
```

```
##                      LOPIT_DC_U2OS_Rep1                      LOPIT_DC_U2OS_Rep2
## "../raw/LOPIT_DC_U2OS_Rep1_PSMs_txt.gz" "../raw/LOPIT_DC_U2OS_Rep2_PSMs_txt.gz"
##                      LOPIT_DC_U2OS_Rep3                                Oconnell
## "../raw/LOPIT_DC_U2OS_Rep3_PSMs_txt.gz"           "../raw/Oconnell_PSMs_txt.gz"
```

Make the cRAP list for filtering

```r
get_fasta_ids <- function(fasta){
  # Load the FASTA
  bs.fasta <- Biostrings::fasta.index(fasta, seqtype = "AA")

  # Extract the UniProt accessions
  accessions <- bs.fasta %>%
    pull(desc) %>%
    stringr::str_extract_all("(?<=\\|).*?(?=\\|)") %>%
    unlist()

  accessions
}

crap.accessions <- get_fasta_ids('../shared_files/cRAP_FullIdentifiers.fasta')
```

Match species to uniprotID

```
hs.accessions <- get_fasta_ids(
  '../shared_files/h.sapiens_UP0000065640.fasta.gz')
sc.accessions <- get_fasta_ids(
  '../shared_files/s.cerevisiae_UP000002311.fasta.gz')

uniprot_2_species <- data.frame('id'=c(hs.accessions, sc.accessions),
                                'species'=c(rep('H.sapiens', length(hs.accessions)),
                                            rep('S.cerevisiae', length(sc.accessions))))
head(uniprot_2_species)
```

```
##       id   species
## 1 Q6ZSK4 H.sapiens
## 2 Q9Y263 H.sapiens
## 3 Q96RE7 H.sapiens
## 4 O43312 H.sapiens
## 5 Q9NP80 H.sapiens
## 6 Q15319 H.sapiens
```

Parse and filter PSMs to remove cRAP proteins

```
print(names(psm))
```

```
## [1] "LOPIT_DC_U2OS_Rep1" "LOPIT_DC_U2OS_Rep2" "LOPIT_DC_U2OS_Rep3"
## [4] "Oconnell"
```

```
psm_parsed <- psm %>% lapply(function(x){
  parse_features(x, TMT=TRUE, level='PSM',
                 crap_proteins=crap.accessions, unique_master=FALSE)
})
```

```
## Parsing features...

## 93514 features found from 9287 master proteins => Input

## 230 cRAP proteins supplied

## 691 proteins identified as 'cRAP associated'

## 92637 features found from 9218 master proteins => cRAP features removed

## 92053 features found from 9178 master proteins => associated cRAP features removed

## Parsing features...

## 95928 features found from 9225 master proteins => Input

## 230 cRAP proteins supplied

## 496 proteins identified as 'cRAP associated'

## 94984 features found from 9160 master proteins => cRAP features removed

## 94485 features found from 9118 master proteins => associated cRAP features removed

## Parsing features...
## 96855 features found from 9459 master proteins => Input

## 230 cRAP proteins supplied

## 557 proteins identified as 'cRAP associated'

## 96050 features found from 9395 master proteins => cRAP features removed

## 95297 features found from 9352 master proteins => associated cRAP features removed

## Parsing features...

## 141598 features found from 10717 master proteins => Input

## 230 cRAP proteins supplied

## 1259 proteins identified as 'cRAP associated'

## 140485 features found from 10672 master proteins => cRAP features removed

## 139567 features found from 10625 master proteins => associated cRAP features removed
```

Annotated the data with the species

```r
psm_parsed_annt <- psm_parsed %>% lapply(function(x){

 species_matches <- x %>% select(Protein.Accessions) %>%
  mutate(Protein.Accessions_sep=Protein.Accessions) %>%
  separate_rows(Protein.Accessions_sep) %>%
  merge(uniprot_2_species, by.x='Protein.Accessions_sep', by.y='id', all.x=TRUE) %>%
  group_by(Protein.Accessions) %>%
  summarise(all_species=paste0(unique(species), collapse='; ')) %>%
  mutate(species=ifelse(grepl(';', all_species), 'mixed', all_species))

 x %>% merge(species_matches, by='Protein.Accessions')
})
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
```

Checking the above hasn't altered `nrow`

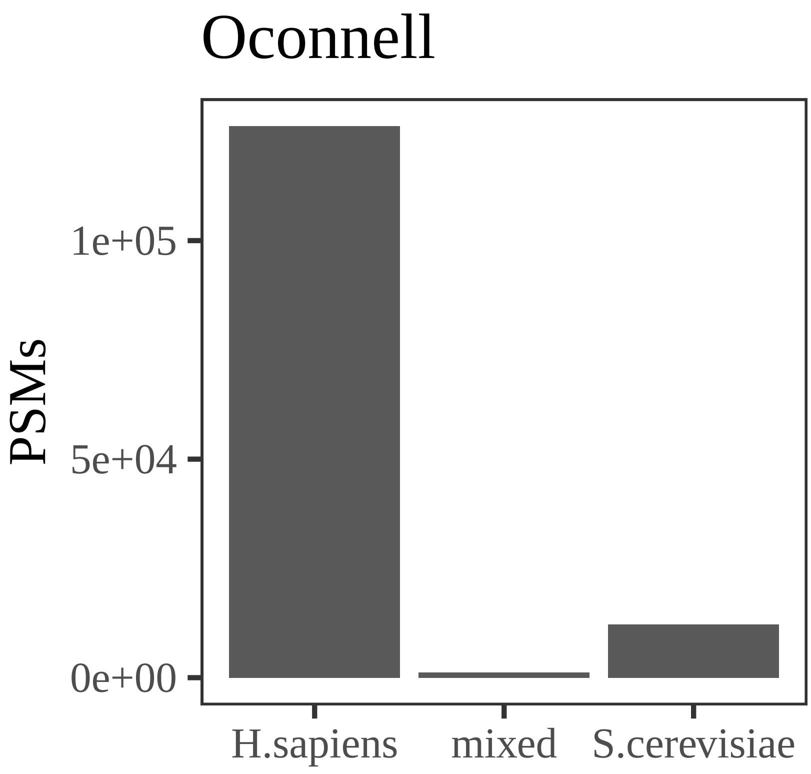
```r
psm_parsed %>% names() %>%
  sapply(function(x){
  print(nrow(psm_parsed[[x]]))
  nrow(psm_parsed_annt[[x]])
})
```

```
## [1] 92053
## [1] 94485
## [1] 95297
## [1] 139567
```

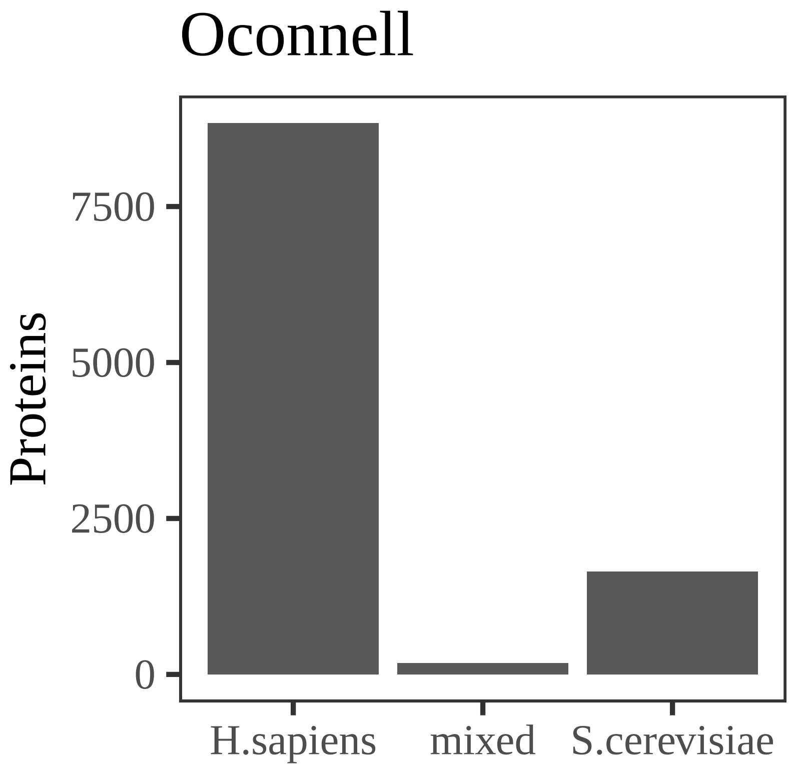
```
## LOPIT_DC_U2OS_Rep1 LOPIT_DC_U2OS_Rep2 LOPIT_DC_U2OS_Rep3           Oconnell
##              92053              94485              95297             139567
```

Summarise PSMs per species for Oconnell et al data

```
x <- 'Oconnell'

p1 <- psm_parsed_annt[[x]] %>%
  group_by(species) %>%
  tally() %>%
  ggplot(aes(species, n)) +
  geom_bar(stat='identity') +
  theme_camprot() +
  xlab('') +
  ylab('PSMs') +
  ggtitle(x)

p2 <- psm_parsed_annt[[x]] %>%
  select(Master.Protein.Accessions, species) %>%
  unique() %>%
  group_by(species) %>% tally() %>%
  ggplot(aes(species, n)) +
  geom_bar(stat='identity') +
  theme_camprot() +
  xlab('') +
  ylab('Proteins') +
  ggtitle(x)

print(p1)
```



```r
print(p2)
```



```r
ggsave(sprintf('../results/plots/%s_psm_n.png', gsub('AGC: ', '', x)), p1)
```

```
## Saving 6.5 x 4.5 in image
```

```r
ggsave(sprintf('../results/plots/%s_proteins_n.png', gsub('AGC: ', '', x)), p2)
```

```
## Saving 6.5 x 4.5 in image
```

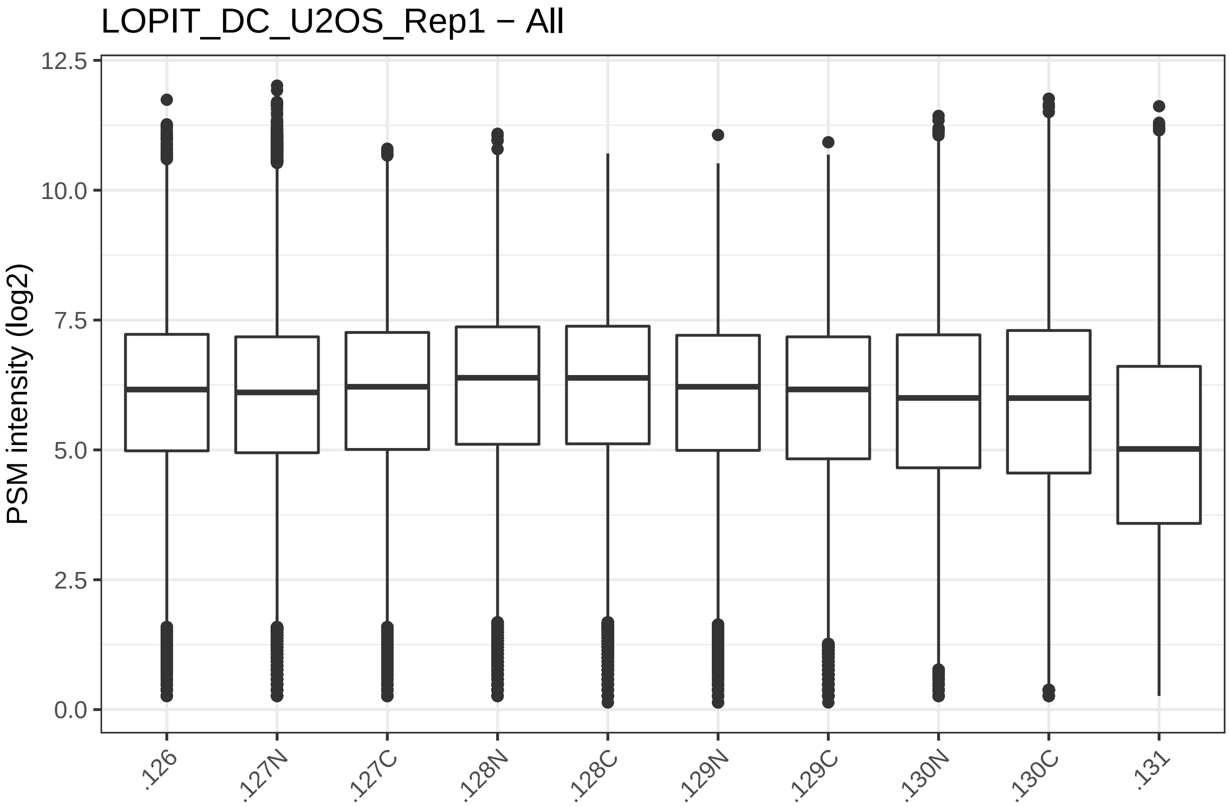
Make MSnSets

```r
psm_res <- psm_parsed_annt %>% lapply(function(x){
  # Abundance columns for TMT PD-output start with Abundance
  abundance_cols <- colnames(x)[grepl('Abundance.', colnames(x))]

  .e <- as.matrix(x[,abundance_cols])
  .f <- x[,setdiff(colnames(x), abundance_cols)]

  # update the column names to remove the 'Abundance.` prefix
  colnames(.e) <- gsub('Abundance.', '', colnames(.e))

  res <- MSnbase::MSnSet(exprs=.e, fData=.f)

  res
})
```

Plotting the distribution of tag intensities in each full dataset and the single species subsets. Note that the tag intensities for yeast fall into the 3 groups we expect given the experimental design.

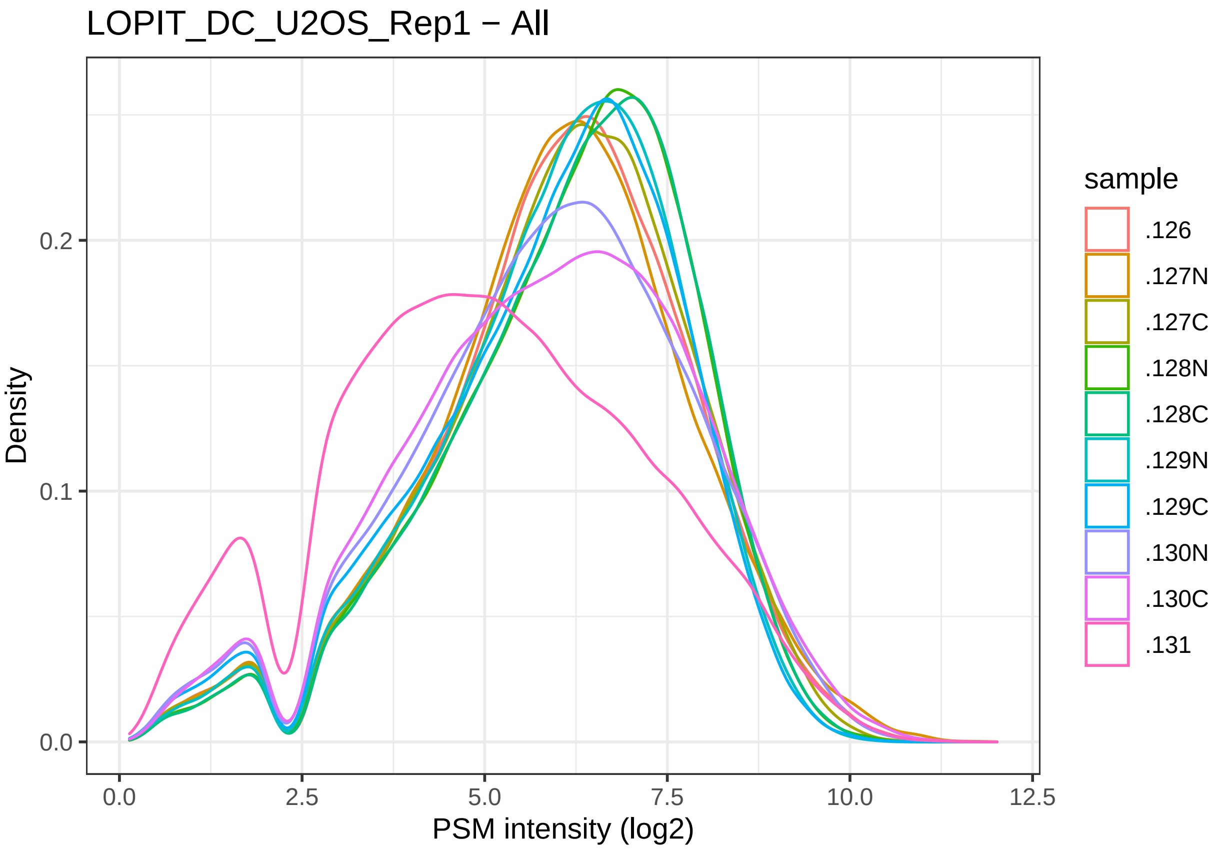
```r
psm_res %>% names() %>% lapply(function(x){

  all <- psm_res[[x]]
  if(x == 'Oconnell_PSMs.txt.gz'){
    hs <- all[fData(all)$species=='H.sapiens']
    sc <- all[fData(all)$species=='S.cerevisiae']

    slices <- list('All'=all, 'H.sapiens'=hs, 'S.cerevisiae'=sc)}
  else{ slices <- list('All'=all) }

  for(slice in names(slices)){
    p <- slices[[slice]] %>% log(base=2) %>% plot_quant() +
      ggtitle(sprintf('%s - %s', x, slice)) +
      ylab('PSM intensity (log2)')
    print(p)

    p <- slices[[slice]] %>% log(base=2) %>% plot_quant(method='density') +
      xlab('PSM intensity (log2)') +
      ggtitle(sprintf('%s - %s', x, slice))
    print(p)
  }
  return(NULL)
})
```

```
## Warning in if (method == "box") {: the condition has length > 1 and only the
## first element will be used
## Warning: Removed 21856 rows containing non-finite values (stat_boxplot).
```

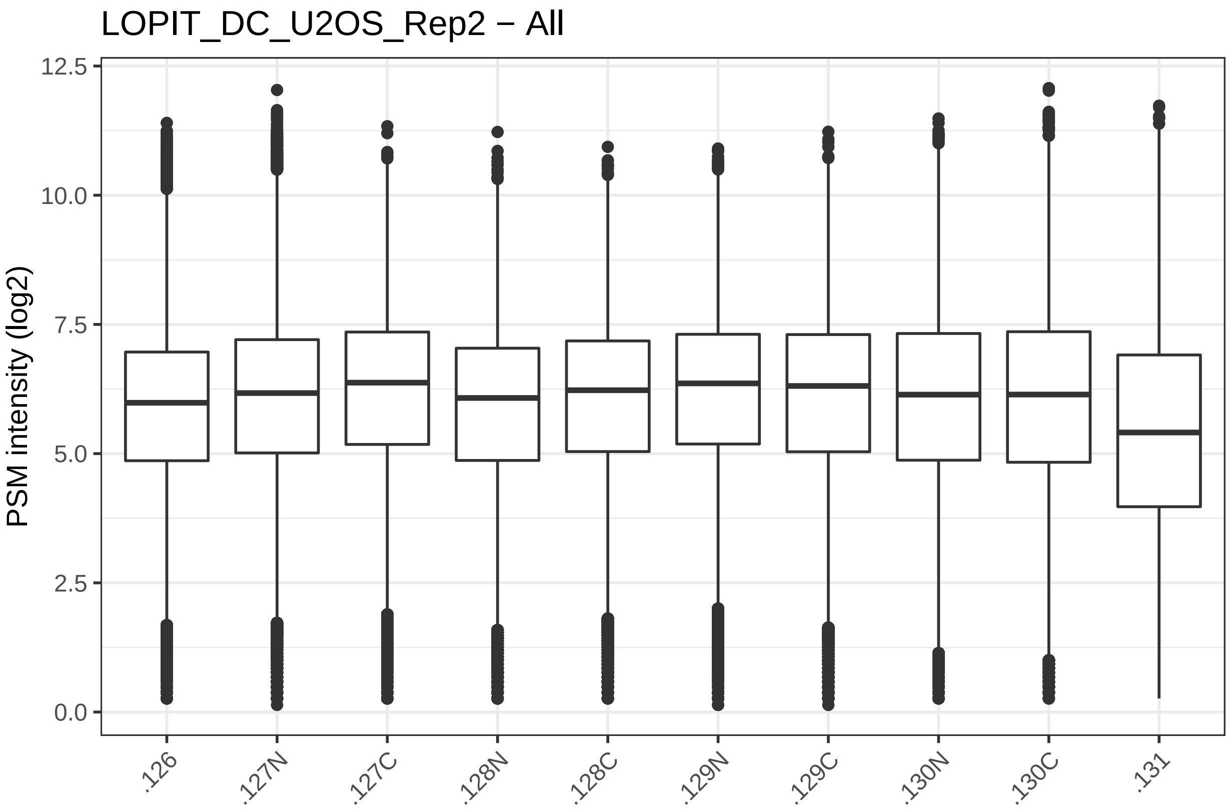


```
## Warning: Removed 21856 rows containing non-finite values (stat_density).
```

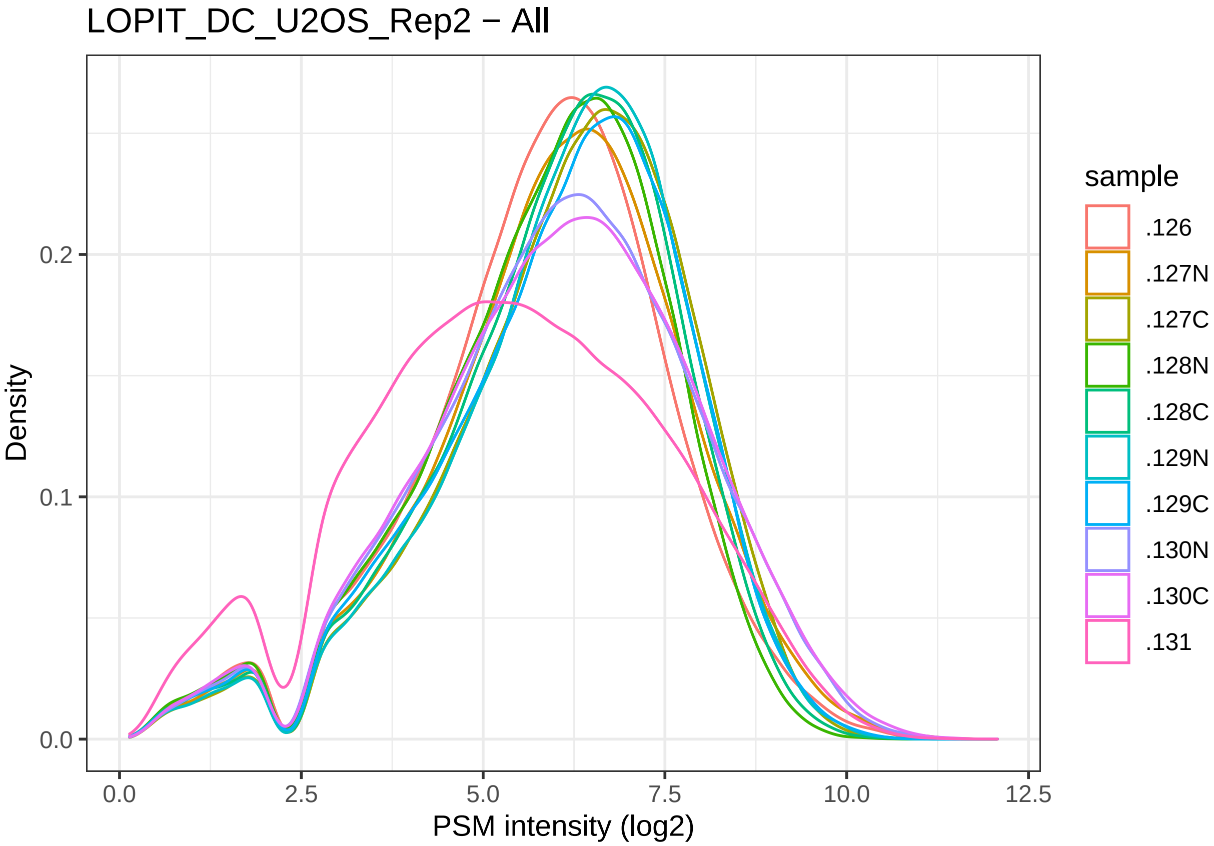
```
## Warning in if (method == "box") {: the condition has length > 1 and only the
## first element will be used
```



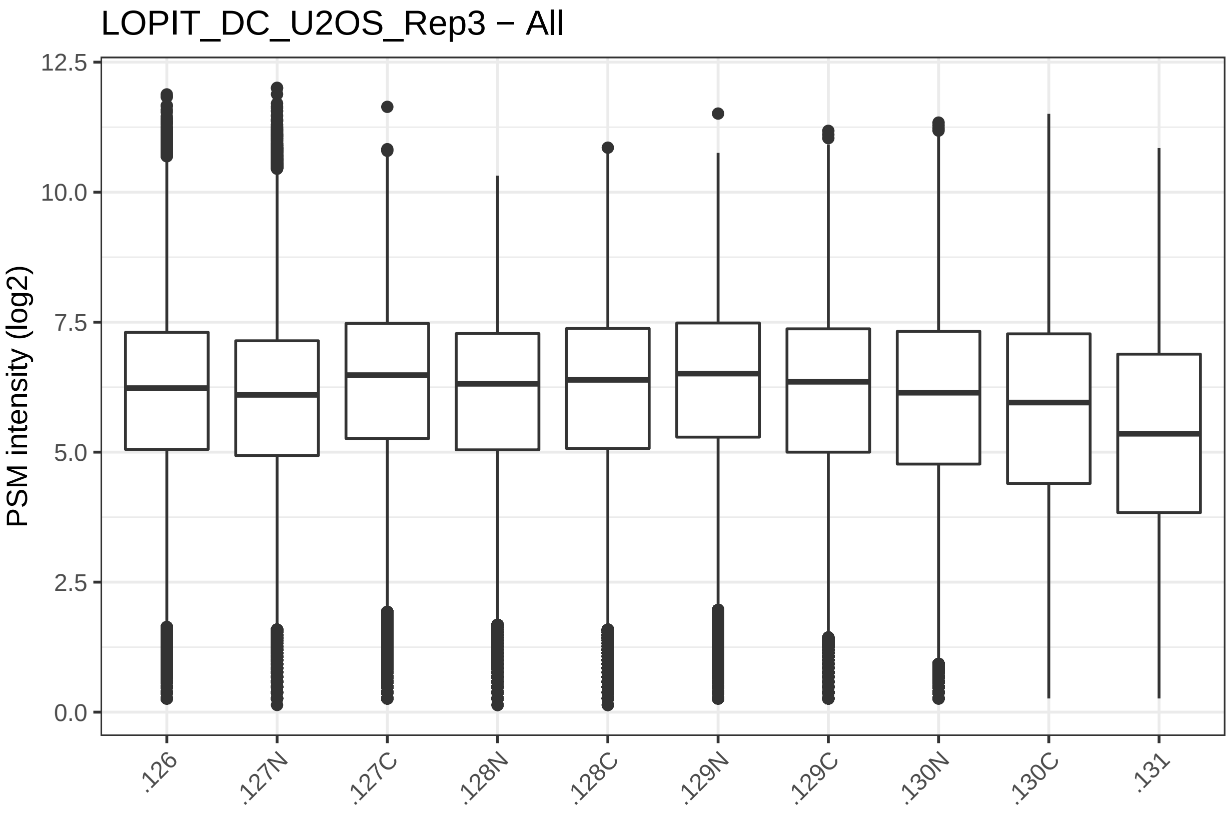
```
## Warning: Removed 20041 rows containing non-finite values (stat_boxplot).
```



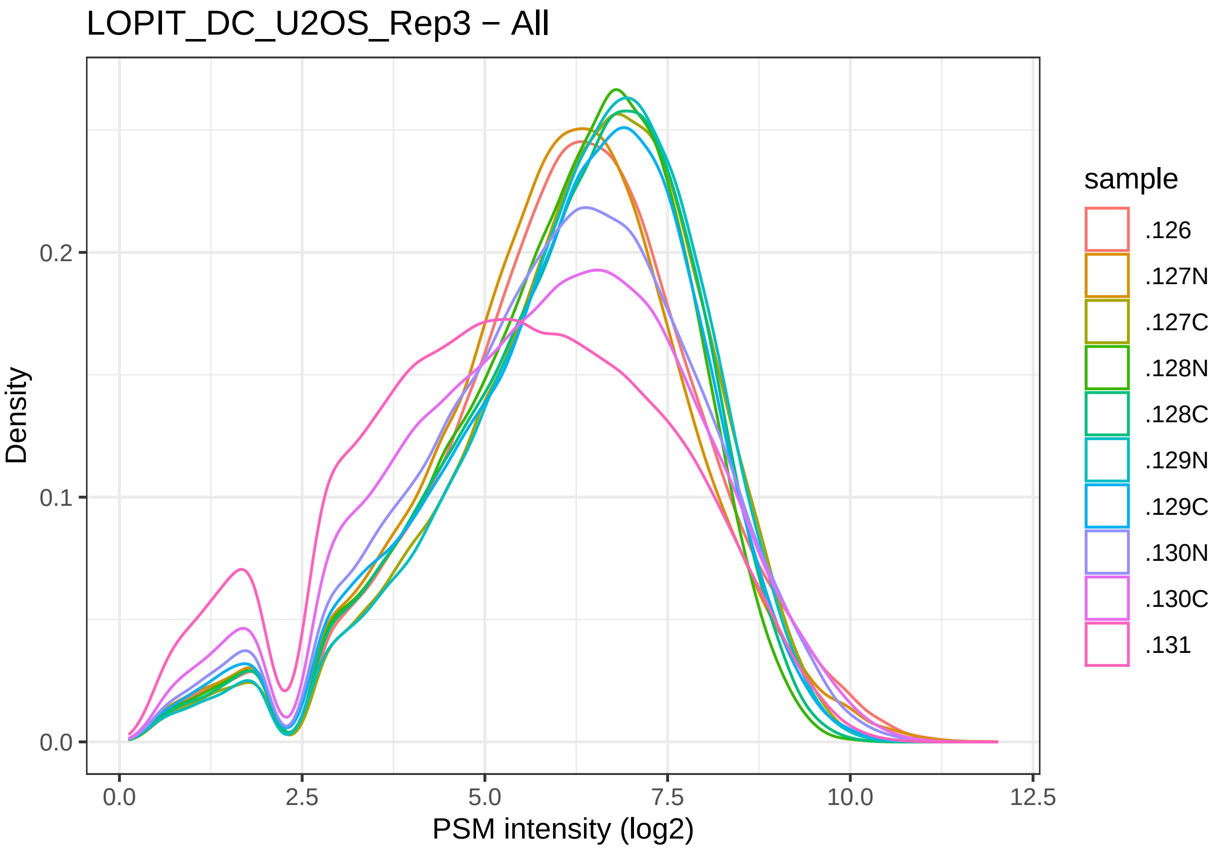
```
## Warning: Removed 20041 rows containing non-finite values (stat_density).

## Warning: the condition has length > 1 and only the first element will be used
```



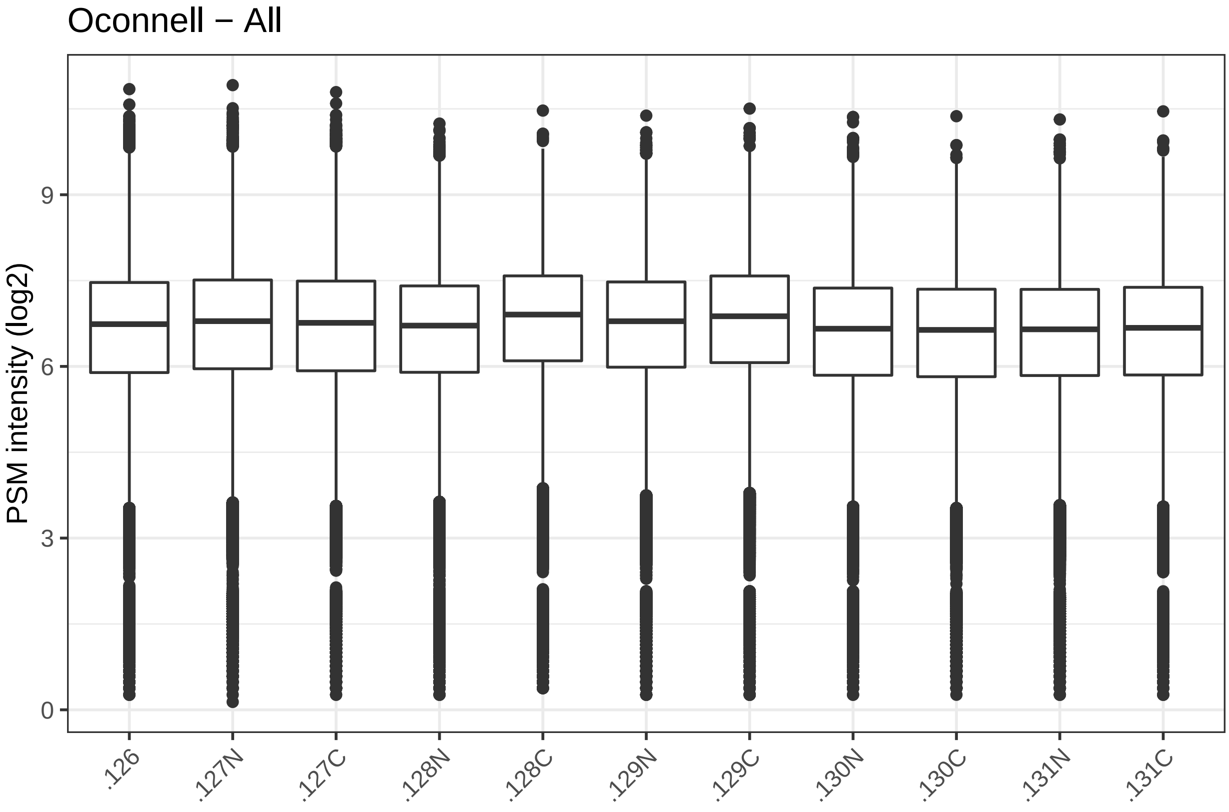
```
## Warning: Removed 22683 rows containing non-finite values (stat_boxplot).
```



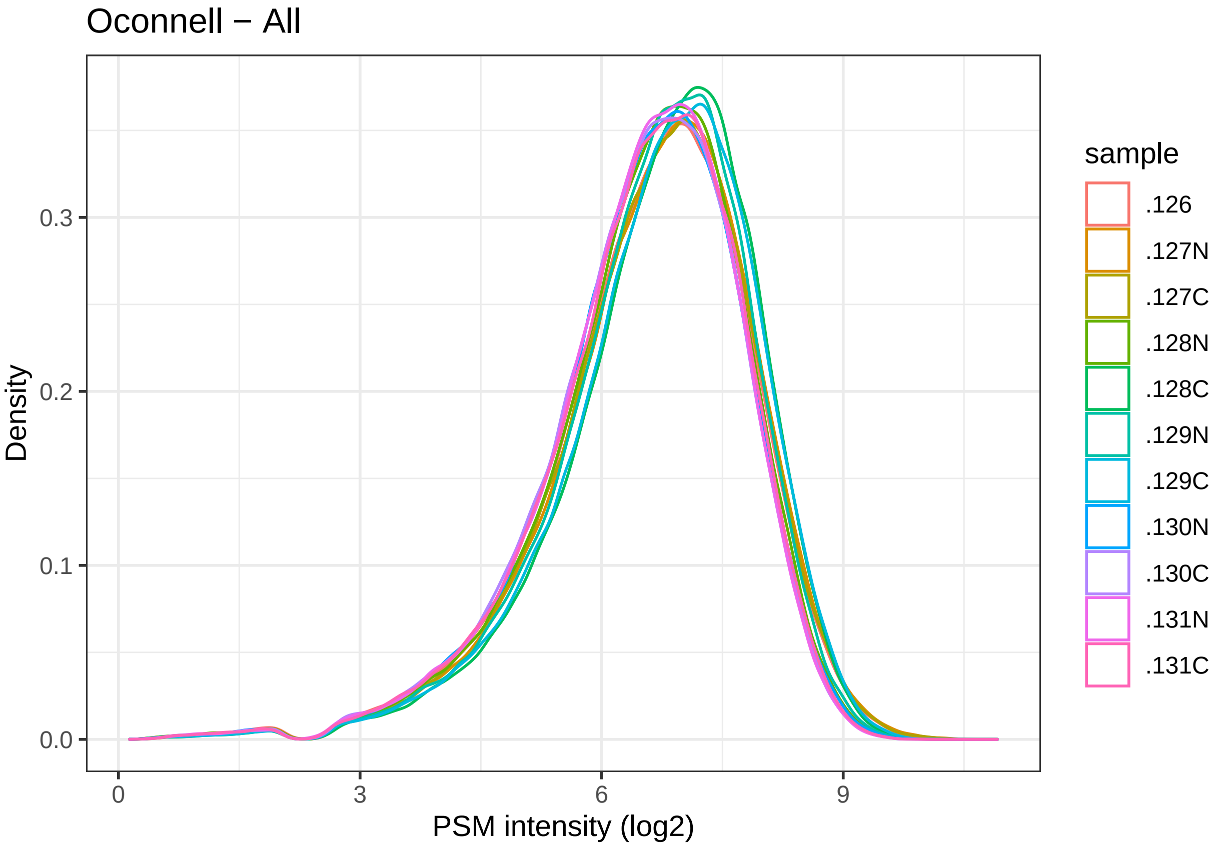
```
## Warning: Removed 22683 rows containing non-finite values (stat_density).

## Warning: the condition has length > 1 and only the first element will be used
```



```
## Warning: Removed 3751 rows containing non-finite values (stat_boxplot).
```



```
## Warning: Removed 3751 rows containing non-finite values (stat_density).
```



```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
```

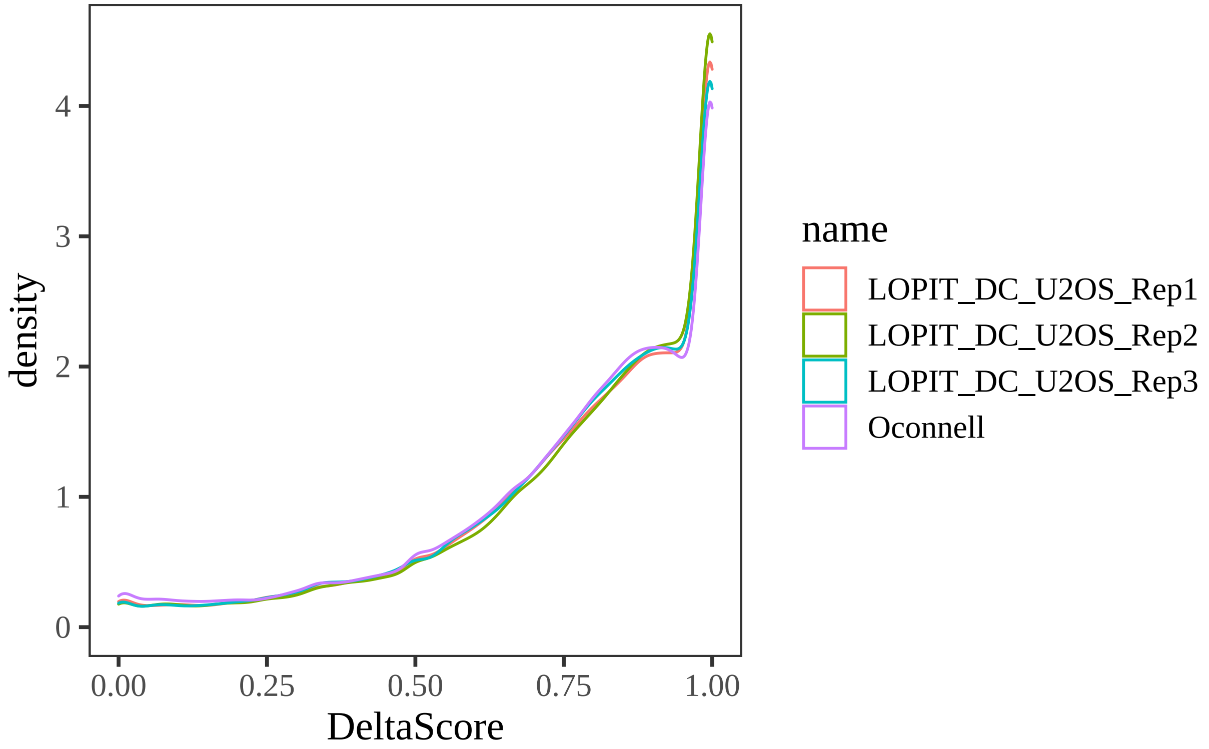
Below, we compare the Delta score and isolation interference distributions for each dataset

```
psm_metrics <-psm_res %>% names() %>% lapply(function(x){
  fData(psm_res[[x]])[,c('DeltaScore', 'Isolation.Interference....')] %>% mutate(name=x)
}) %>% do.call(what='rbind')
```

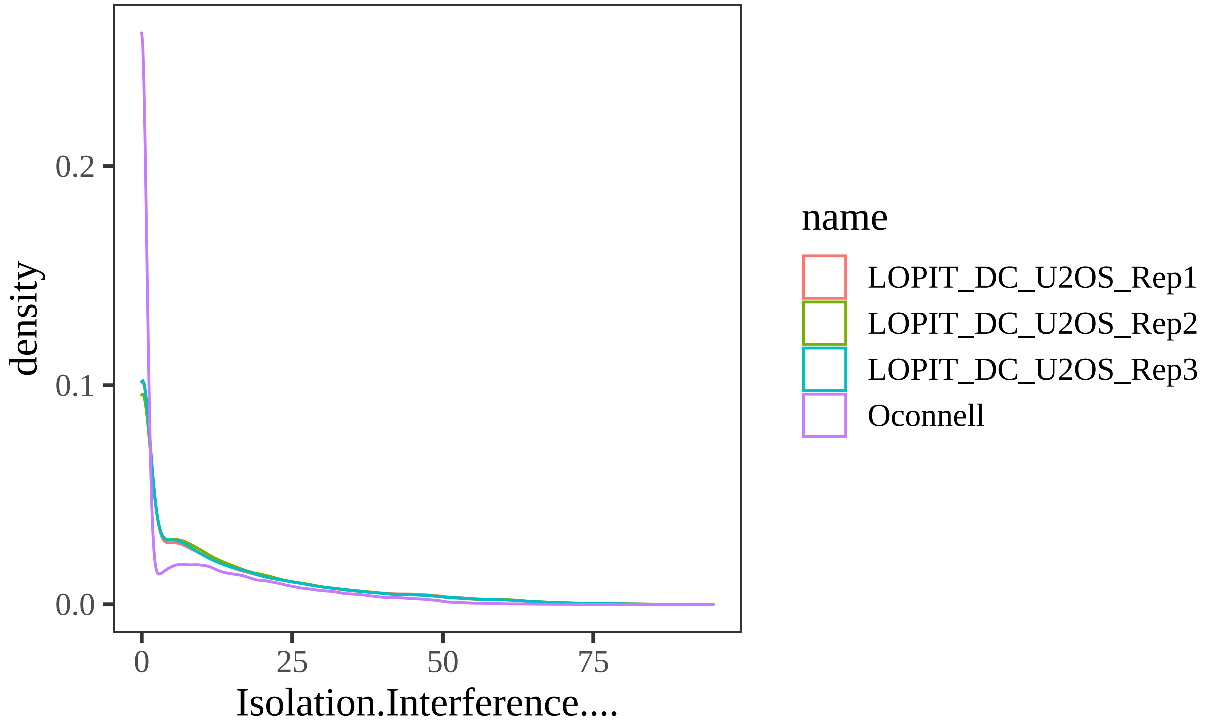
```
p <- psm_metrics %>%
  ggplot(aes(DeltaScore, colour=name)) +
  geom_density() +
  theme_camprot(base_size=15)
```

```r
print(p)
```

```
## Warning: Removed 103 rows containing non-finite values (stat_density).
```



```r
print(p + aes(Isolation.Interference....))
```



Save for downstream notebooks

```r
saveRDS(psm_res, '../results/psm_res.rds')
```