MDstatsDIAMS: Real Data Comparison of Methods for Testing Mean Differences

Namgil Lee*, Hojin Yoo†, Juhyoung Kim‡, Heejung Yang§

June 14, 2025

Contents

This vignette includes numerical comparison of the methods for testing mean differences using real DIA-MS data, which are given in either Spectronaut, MaxQuant, or Skyline report format. This vignette reproduces figures and tables in Section 3.2 Mass Spectrometry Data Analysis of the paper "*A Shrinkage-based Statistical Method for Testing Group Mean Differences in Quantitative Bottom-up Proteomics*" written by the authors.

Preparation

1. Set parameters to reduce analysis time

Because the real data size can be too large for the vignette, users can choose parameters to reduce analysis time.

```
# n_protein:
#   Size of a subset of proteins randomly selected.
#   If -1 or Inf, include all proteins. Default to 100.
n_protein <- Inf

# input_report_format:
#   User must choose one of "sn"(Spectronaut), "mq" (MaxQuant), or
#   "sk" (Skyline).
input_report_format <- "sk"

# result_save_folder:
#   Path to the folder to save ttest results.
result_save_folder <- "/Users/namgil/Documents/Projects/MDstatsDIAMS/vignettes/"

# remove_intermediate_reports:
#   If TRUE, remove report from environment if not used in further steps.
#   Its value does not influence the results.
#   Default is TRUE.
remove_intermediate_reports <- TRUE
```

*Kangwon National University, and Bionsight Inc., namgil.lee@kangwon.ac.kr
†Bionsight Inc.
‡Kangwon National University
§Kangwon National University, and Bionsight Inc., heejyang@kangwon.ac.kr

```
# control_condition,treat_condition:
#   Comparison between control and treatment. Select conditions
#   from {"DMSO", "10nM", "100nM", "100mM"} if Spectronaut;
#   from {"CON1", "CON4", "CON5", "CON8"} if MaxQuant or Skyline
if (input_report_format == "sn") {
  control_condition <- "DMSO"
  treat_condition <- "100mM"
} else {
  control_condition <- "CON1"
  treat_condition <- "CON8"
}

# subconditions:
#   List of four conditions for additional pairwise comparisons.
#   Set {"DMSO", "10nM", "100nM", "100mM"} if Spectronaut;
#   {"CON1", "CON4", "CON5", "CON8"} if MaxQuant or Skyline
if (input_report_format == "sn") {
  subconditions <- c("DMSO", "10nM", "100nM", "100mM")
} else {
  subconditions <- c("CON1", "CON4", "CON5", "CON8")
}
```

2. Load packages

```
library(dplyr)
library(MDstatsDIAMS)
library(pROC)
```

3. Load target list

Load an on-target protein list.

```
known_target_df <- read.csv(
  paste0(
    "https://zenodo.org/records/15653980/files/",
    "known_target_list_staurosporine_davis2011.csv"
  )
)

if (input_report_format == "sk") {
  protein_column = "ProteinName"
} else {
  protein_column = "UniprotID"
}
```

4. Load report data

Load and preprocess report data, and convert it into standard format. Spectronaut, MaxQuant, and Skyline reports can be converted into standard format directly.

- Reports are filtered by q-value < 0.01, fragment peak area > 1.
- Decoy proteins are removed from MaxQuant and Skyline.
- Top-3 fragment ions are selected in MaxQuant.

```
if (input_report_format == "sn") {
  ## Load Spectronaut report
  df_real <- arrow::read_parquet(
```

```
    paste0(
      "/Users/namgil/Documents/Projects/MDstatsDIAMS/data/",
      "lip_quant_staurosporine_hela_sn_report.parquet"
    )
  )

  df_filtered <- convert_sn_to_standard(df_real, filter_identified = TRUE)

} else if (input_report_format == "mq") {
  ## Load MaxQuant report
  mq_root <- paste0("/Users/namgil/Documents/Projects/MDstatsDIAMS/data/",
                    "lip_quant_staurosporine_hela_mq_report_4conds/")

  mq_evidence_path <- paste0(mq_root, "evidence.txt")
  mq_msms_path <- paste0(mq_root, "msms.txt")

  df_real <- read.delim(mq_evidence_path)
  mq_msms <- read.delim(mq_msms_path)

  ##  It is required to generate `annotation.txt' file in prior for **MSstats**.
  mq_an <- create_annotation_df(
    evidence = df_real,
    n_replicates_per_condition = 4
  )
  df_filtered <- convert_mq_to_standard(df_real, mq_msms, mq_an)

} else if (input_report_format == "sk") {
  ## Load Skyline report
  df_real <- arrow::read_parquet(
    paste0("/Users/namgil/Documents/Projects/MDstatsDIAMS/data/",
           "lip_quant_staurosporine_hela_sk_transition_4conds.parquet")
  )

  # Make an annotation data frame
  sk_an <- data.frame(
    Condition = rep(paste0("CON", c(1, 4, 5, 8)), each = 4),
    Replicate = paste0("StauroDoseResp-", c(1:4, 17:24, 33:36)),
    Run = paste0("StauroDoseResp-", c(1:4, 17:24, 33:36))
  )

  df_filtered <- convert_sk_to_standard(df_real, annotation = sk_an)
}
```

```
## Warning in convert_sk_to_standard(df_real, annotation = sk_an): NAs introduced
## by coercion
```

```
print(dim(df_filtered))
```

```
## [1] 16999987        7
```

5. Randomly select a subset of the predefined number of proteins to reduce analysis time.

```
if (
  n_protein < 0
  || is.infinite(n_protein)
```

```
  || n_protein >= n_distinct(df_filtered$protein_id)
) {
  df_subset <- df_filtered
} else {
  set.seed(111)
  protein_subset <- sample(unique(df_filtered$protein_id), n_protein)
  df_subset <- df_filtered %>% filter(protein_id %in% protein_subset)
}

print(dim(df_subset))
```

```
## [1] 16999987        7
```

6. Remove the large report data from the R environment that will not be used in the next steps.

Comparison Between Control and Treatment

1. Run statistical methods for comparing the means between two conditions, control (DMSO) and treatment (100 uM).

- Select a subset of the report for two conditions, control (DMSO) and treatment (100 uM)

```
df_two_conds <- df_subset %>%
  filter(condition %in% c(control_condition, treat_condition))

print(paste("Comparing mean differences between two conditions:",
            control_condition,
            treat_condition))
```

```
## [1] "Comparing mean differences between two conditions: CON1 CON8"
```

- Run statistical methods.

```
## Compute ttest result comparing two conditions ##
save_path <- paste0(
  result_save_folder,
  input_report_format, "_",
  "ttest_result_n", n_protein, "_", control_condition, "_", treat_condition,
  ".RData"
)

if (file.exists(save_path)) {
  load(save_path)
} else {
  # Run methods
  ttest_result <- run_ttests(
    df_two_conds,
    method_names = NULL,
    boot_denom_eps = 0.3,
    base_condition = control_condition
  )
  save(ttest_result, file = save_path)
}
```

2. Append local FDR score and on-target information

```r
lfdr_result <- compute_lfdr_result(
  ttest_result,
  known_target_df = known_target_df,
  is_target_column = "is_target",
  protein_column = protein_column
)
```

```
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
##
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
##
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
##
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
##
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
```

3. Compute sensitivity and specificity from the local fdr results

```r
protein_sensitivity_result <- compute_sensitivity_result(
  lfdr_result,
  q_value_column = "lfdr",
  is_target_column = "is_target",
  group_column = c("experiment", "protein_id")
)
```

- Print sensitivity and specificity in a brief form

```r
print(paste("Comparison:", control_condition, "/", treat_condition))
```

```
## [1] "Comparison: CON1 / CON8"
```

```r
spec_levels <- seq(0.8, 0.2, -0.2)
sens_matrix <- matrix(
  NA, length(protein_sensitivity_result), length(spec_levels),
  dimnames = list(names(protein_sensitivity_result), spec_levels)
)

for (level in spec_levels) {
  for (method in names(protein_sensitivity_result)) {
    sensitivity_table <- protein_sensitivity_result[[method]][[1]]
```

```
    spec_id_at_level <- max(which(sensitivity_table$specificity > level))
    sens_at_level <- sensitivity_table$sensitivity[spec_id_at_level]
    sens_matrix[method, as.character(level)] <- sens_at_level
  }
}

print(round(sens_matrix, 2))
```

```
##              0.8  0.6  0.4  0.2
## msstatslip  0.20 0.40 0.60 0.80
## rots        0.20 0.41 0.60 0.80
## paired      0.14 0.36 0.58 0.79
## independent 0.20 0.40 0.60 0.80
## shrinkage   0.20 0.40 0.60 0.80
```

4. Print contingency table at fixed significance level

```
## [1] "---------- Significance level: 0.10 ----------"
```

```
## $`CON1/CON8`
##   Rejected msstatslip   rots paired independent shrinkage
## 1    FALSE     103623 122491 121472      119807     47317
## 2     TRUE          0      0      0           0         0
```

```
## [1] "---------- Significance level: 0.05 ----------"
```

```
## $`CON1/CON8`
##   Rejected msstatslip   rots paired independent shrinkage
## 1    FALSE     103623 122491 121472      119807     47317
## 2     TRUE          0      0      0           0         0
```

```
## [1] "---------- Significance level: 0.01 ----------"
```

```
## $`CON1/CON8`
##   Rejected msstatslip   rots paired independent shrinkage
## 1    FALSE     103623 122491 121472      119807     47317
## 2     TRUE          0      0      0           0         0
```

Testing Across Multiple Conditions

- Aggregate p-values of consecutive comparisons

```
protein_df_agg <- list()
aucs <- c()

### Collect p-values
num_cond <- length(subconditions)
for (i in 1:(num_cond - 1)) {
  ## Compute p-value from consecutive conditions
  control_condition <- subconditions[i]
  treat_condition <- subconditions[i + 1]

  df_two_conds <- df_subset %>%
    filter(condition %in% c(control_condition, treat_condition))

  ## Compute ttest result comparing two conditions ##
  save_path <- paste0(
    result_save_folder,
```

```
    input_report_format, "_",
    "ttest_result_n", n_protein, "_", control_condition, "_", treat_condition,
    ".RData"
  )
  if (file.exists(save_path)) {
    load(save_path)
  } else {
    ttest_result <- run_ttests(
      df_two_conds,
      method_names = NULL,
      boot_denom_eps = 0.3,
      base_condition = control_condition
    )
    save(ttest_result, file = save_path)
  }
  ##

  ## Compute "signed" p-value in precursor level
  for (method in names(ttest_result)) {
    signed_pvalue_table <- ttest_result[[method]][[1]] %>%
      select(experiment, protein_id, precursor_id, p.value, estimate) %>%
      mutate(comparison_1 = control_condition,
             comparison_2 = treat_condition,
             sign_of_test = sign(estimate)) %>%
      filter(!is.na(p.value))

    # protein_df_agg[[method]] columns: experiment, protein_id, precursor_id,
    #   p.value, estimate, comparison_1, comparison_2, sign_of_test
    protein_df_agg[[method]] <- rbind(
      protein_df_agg[[method]],
      signed_pvalue_table
    )
  }
}

# Aggregate all the comparisons for each precursor
for (method in names(protein_df_agg)) {
  protein_df_agg[[method]] <- protein_df_agg[[method]] %>%
    group_by(experiment, protein_id, precursor_id) %>%
    mutate(sign_of_minimal_pvalue = sign_of_test[which.min(p.value)]) %>%
    group_by(experiment, protein_id, precursor_id) %>%
    mutate(prod_recomputed_pvalue = prod(
      0.5 - sign_of_test * sign_of_minimal_pvalue * (0.5 - 0.5 * p.value),
      na.rm = TRUE)) %>%
    group_by(experiment, protein_id, precursor_id) %>%
    summarise(p.value = sum(prod_recomputed_pvalue * 2, na.rm = TRUE))
} ## experiment, protein_id, precursor_id, p.value

# Aggregate to protein level
for (method in names(protein_df_agg)) {
  protein_df_agg[[method]] <- protein_df_agg[[method]] %>%
    group_by(experiment, protein_id) %>%
    summarise(p.value = min(p.value, na.rm = TRUE))
```

```
}

# Append on-target information
for (method in names(protein_df_agg)) {
  protein_df_agg[[method]]$is_target <- (
    protein_df_agg[[method]]$protein_id %in% known_target_df[[protein_column]]
  )

  roc_prot <- pROC::roc(
    protein_df_agg[[method]]$is_target,
    1 - protein_df_agg[[method]]$p.value,
    auc = TRUE,
    plot = FALSE,
    quiet = TRUE
  )
  aucs[method] <- roc_prot$auc
}

print(paste(c("Conditions:", subconditions), collapse = " "))
```

```
## [1] "Conditions: CON1 CON4 CON5 CON8"
```

```
print("AUCs:")
```

```
## [1] "AUCs:"
```

```
print(aucs)
```

```
##  msstatslip        rots      paired independent   shrinkage
##   0.5021906   0.4998240   0.5111022   0.5158224   0.5207458
```