Real Data Example

Package Setup

```
library(glmnet)
```

```
Loading required package: Matrix
```

```
Warning: package 'Matrix' was built under R version 4.4.1
```

```
Loaded glmnet 4.1-8
```

```
library(MASS)
```

```
Warning: package 'MASS' was built under R version 4.4.1
```

```
library(patchwork)
```

```
Warning: package 'patchwork' was built under R version 4.4.1
```

```

Attaching package: 'patchwork'
```

```
The following object is masked from 'package:MASS':

    area
```

Real Data Input

```
y <- read.csv('./Onset of Labor/Training/DOS.csv')
X_CyTOF <- read.csv('./Onset of Labor/Training/CyTOF.csv')
X_Meta <- read.csv('./Onset of Labor/Training/Metabolomics.csv')
X_Proteomics <- read.csv('./Onset of Labor/Training/Proteomics.csv')
X_CyTOF = as.matrix(X_CyTOF[,-1])
X_Meta = as.matrix(X_Meta[,-1])
X_Proteomics = as.matrix(X_Proteomics[,-1])
y = y[, 2]
X = cbind(X_CyTOF, X_Meta, X_Proteomics)
y = y - mean(y)
X[is.na(X)] = 0
X = X[, colSums(abs(X)) != 0]
X = scale(X)
n = nrow(X)
p = ncol(X)
```

Preprocessing and Setup for SyNPar

```
binary_search = function(coef_real, coef_knockoff, q) {
  left = 0
  right = max(coef_real)
  while (abs(left - right) > 1e-8) {
    mid = ((left + right) / 2)
    num_negative = length(which(coef_knockoff >= mid)) + q/2
    num_positive = max(length(which(coef_real >= mid)), 1)
    FDP = num_negative / num_positive
    if (FDP > q) {
      left = mid
    } else {
      right = mid
    }
  }
  return(right)
}

correction_factor_estimation = function(X, y, beta_real, hat_real, residuals, best_lambda
  # beta_real: coef of the positive control = beta daggar = beta hat
  n = nrow(X)
  beta_real[which(beta_real>0)] = beta_real[which(beta_real>0)] + best_lambda
  beta_real[which(beta_real<0)] = beta_real[which(beta_real<0)] - best_lambda
  y_correction = X %*% beta_real + sample(residuals, replace = TRUE)
  X = scale(X)
  y_correction = y_correction - mean(y_correction)
  Signal_index_correction = which(beta_real != 0)

  #### correction true ######
  model_correction = glmnet(X, y_correction, alpha = 1, intercept = F, lambda = best_lamb
  coef_correction = coef(model_correction)
  coef_correction = coef_correction[-1] # remove the intercept
  # hat_sigma_corr = sqrt(sum((y - X %*% as.vector(coef_correction))^2) / max(1,(n - sum(
  coef_correction = abs(as.vector(coef_correction))
  # coef_correction = coef_correction[-which.max(coef_correction)]
  X_snp = X
  X_snp = scale(X_snp)
  y_snp = sample(residuals, replace = TRUE)
  y_snp = y_snp - mean(y_snp)
  model_snp = glmnet(X_snp, y_snp, alpha = 1, intercept = F, lambda = best_lambda)
  coef_snp = coef(model_snp)
  coef_snp = coef_snp[-1] # remove the intercept
  coef_snp = abs(as.vector(coef_snp))
  # coef_snp = coef_snp[-which.max(coef_snp)]
  left_correction = 0.05
  right_correction = max(coef_correction)/(best_lambda + sqrt(log(p))/sqrt(n))
  while (abs(right_correction - left_correction) > 1e-8) {
    mid = ((right_correction + left_correction) / 2)
    FDR_vector = c()
    # for (b in 1:B){
```

```
    # the “inflated” null coefficients after applying the candidate correction.
    corrected = abs(coef_snp) + mid * hat_real * (best_lambda + sqrt(log(p))/sqrt(n))
    # threshold computed for this candidate correction factor
    tau_mid = binary_search(coef_correction, corrected, q)
    # where the “corrected” real coefficients exceed the threshold tau_mid
    rejected_mid = which(coef_correction >= tau_mid)
    # The empirical false discovery proportion at threshold tau_mid
    FDR_mid = length(which(coef_correction[setdiff(1:p, Signal_index_correction)] >= tau_
    FDR_vector = c(FDR_vector, FDR_mid)
    # }
    FDR_mean = mean(FDR_vector)
    if (FDR_mean > q) { #
      left_correction = mid
    } else {
      right_correction = mid
    }
  }
  return(right_correction)
}

correction_factor_estimation_normal = function(X, y, beta_real, hat_real, best_lambda, q)
  # beta_real: coef of the positive control = beta daggar = beta hat
  n = nrow(X)
  beta_real[which(beta_real>0)] = beta_real[which(beta_real>0)] + best_lambda
  beta_real[which(beta_real<0)] = beta_real[which(beta_real<0)] - best_lambda
  y_correction = X %*% beta_real + rnorm(n, sd = hat_real)
  X = scale(X)
  y_correction = y_correction - mean(y_correction)
  Signal_index_correction = which(beta_real != 0)

  #### correction true ######
  model_correction = glmnet(X, y_correction, alpha = 1, intercept = F, lambda = best_lamb
  coef_correction = coef(model_correction)
  coef_correction = coef_correction[-1] # remove the intercept
  # hat_sigma_corr = sqrt(sum((y - X %*% as.vector(coef_correction))^2) / max(1,(n - sum(
  coef_correction = abs(as.vector(coef_correction))
  # coef_correction = coef_correction[-which.max(coef_correction)]
  X_snp = X
  X_snp = scale(X_snp)
  y_snp = rnorm(n, sd = hat_real)
  y_snp = y_snp - mean(y_snp)
  model_snp = glmnet(X_snp, y_snp, alpha = 1, intercept = F, lambda = best_lambda)
  coef_snp = coef(model_snp)
  coef_snp = coef_snp[-1] # remove the intercept
  coef_snp = abs(as.vector(coef_snp))
  # coef_snp = coef_snp[-which.max(coef_snp)]
  left_correction = 0.05
  right_correction = max(coef_correction)/(best_lambda + sqrt(log(p))/sqrt(n))
  while (abs(right_correction - left_correction) > 1e-8) {
    mid = ((right_correction + left_correction) / 2)
    FDR_vector = c()
```

```
    # for (b in 1:B){

    # the “inflated” null coefficients after applying the candidate correction.
    corrected = abs(coef_snp) + mid * hat_real * (best_lambda + sqrt(log(p))/sqrt(n))
    # threshold computed for this candidate correction factor
    tau_mid = binary_search(coef_correction, corrected, q)
    # where the “corrected” real coefficients exceed the threshold tau_mid
    rejected_mid = which(coef_correction >= tau_mid)
    # The empirical false discovery proportion at threshold tau_mid
    FDR_mid = length(which(coef_correction[setdiff(1:p, Signal_index_correction)] >= tau_
    FDR_vector = c(FDR_vector, FDR_mid)
    # }
    FDR_mean = mean(FDR_vector)
    if (FDR_mean > q) { #
      left_correction = mid
    } else {
      right_correction = mid
    }
  }
  return(right_correction)
}

fit = cv.glmnet(X, y, intercept = F, alpha = 1)
best_lambda <- 0.2 * fit$lambda.min
best_lasso_model <- glmnet(X, y, alpha = 1,intercept = F, lambda = best_lambda)
coef_real = coef(best_lasso_model)
coef_real = coef_real[-1]
coef_corr = coef_real
hat_sigma = sqrt(sum((y - X %*% as.vector(coef_real))^2) / max(n/2,(n - sum(coef_real !=
coef_real = abs(as.vector(coef_real))
residuals = min(n^(1/4), sqrt(n / max(n/2,(n - sum(coef_real != 0)))))*(y - X %*% as.vect

X_knockoff = X
X_knockoff = scale(X_knockoff)
beta_knockoff = rep(0, p)
```

SyNPar (param)

```
y_knockoff = X_knockoff %*% beta_knockoff + rnorm(n, sd = hat_sigma)
y_knockoff = y_knockoff - mean(y_knockoff)

### Lasso: Fit model to synthetic data (X, tilde_y) ###
knockoff_lasso_model <- glmnet(X_knockoff, y_knockoff, intercept = F, alpha = 1, lambda =
coef_knockoff = coef(knockoff_lasso_model)
coef_knockoff = coef_knockoff[-1]
nu_vec = c()

B = 5
```

```
for (b in 1:B) {
  set.seed(b)
  nu = correction_factor_estimation_normal(X, y, coef_corr, hat_sigma, best_lambda, 0.1)
  nu_vec = c(nu_vec, nu)
}
nu = max(nu_vec)

### Find the threshold tau_q for given FDR level q ###
# coef_real is beta_hat from real data
coef_knockoff = abs(coef_knockoff) + nu*hat_sigma*(best_lambda + log(p)/sqrt(n))

threshold = binary_search(coef_real, coef_knockoff, 0.1)
reject = which(coef_real >= threshold)

X_selected_snp = X[, reject]
snp_model = lm(y~X_selected_snp)
snp_coef <- coef(snp_model)

# output
num_feature <- length(reject)
r_squared <- summary(snp_model)$adj.r.squared

cat("Number of selected features by SyNPar (param):", num_feature, "\n")
```

```
Number of selected features by SyNPar (param): 10
```

```
cat("R-squared of SyNPar (param):", r_squared, "\n")
```

```
R-squared of SyNPar (param): 0.8915747
```

SyNPar (non-param)

```
y_knockoff = X_knockoff %*% beta_knockoff + sample(residuals, replace = TRUE)
y_knockoff=y_knockoff - mean(y_knockoff)


knockoff_lasso_model <- glmnet(X_knockoff, y_knockoff, intercept = F, alpha = 1, lambda =
coef_knockoff = coef(knockoff_lasso_model)
coef_knockoff = coef_knockoff[-1]
nu_vec = c()

B = 5
for (b in 1:B) {
  set.seed(b)
  nu = correction_factor_estimation(X, y, coef_corr, hat_sigma, residuals, best_lambda, 0
  nu_vec = c(nu_vec, nu)
}
```

```
nu = max(nu_vec)

coef_knockoff = abs(coef_knockoff) + nu*hat_sigma*(best_lambda + log(p)/sqrt(n))


threshold = binary_search(coef_real, coef_knockoff, 0.1)
reject = which(coef_real >= threshold)

X_selected_snp = X[, reject]
snp_model = lm(y~X_selected_snp)
snp_coef <- coef(snp_model)

# output
num_feature <- length(reject)
r_squared <- summary(snp_model)$adj.r.squared

cat("Number of selected features by SyNPar (non-param):", num_feature, "\n")
```

```
Number of selected features by SyNPar (non-param): 7
```

```
cat("R-squared of SyNPar (non-param):", r_squared, "\n")
```

```
R-squared of SyNPar (non-param): 0.8623458
```