Simulation Examples for Cox Models

Package Setup

```
if (!require("devtools", quietly = TRUE))
  install.packages("devtools")
```

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

```
devtools::install_github("anonstats123/Nullstrap")
```

```
Skipping install of 'Nullstrap' from a github remote, the SHA1 (4519a73e) has not changed
since last install.
  Use `force = TRUE` to force installation
```

```
library(PRROC)
library(Nullstrap)
library(MASS)
```

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

```
library(knockoff)
library(survival)
```

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

```
library(simsurv)
```

Simulation Data Generation

```
n <- 400
p <- 200
nonzero_coefs <- 30
Amp <- 5
beta <- rep(0, p)
beta[1:nonzero_coefs] <- Amp
names(beta) = paste0("X", 1:p)
rho <- 0.6
Theta.8 <- toeplitz(rho^(0:(p - 1)))
X <- mvrnorm(n, rep(0, p), Sigma = Theta.8)
X <- scale(X)/(sqrt(n))
colnames(X) = paste0("X", 1:p)
Signal_index <- 1:nonzero_coefs
true_labels <- beta != 0
surv_time <- simsurv(lambdas = 1, gammas = 1, betas = beta, x = as.data.frame(X))
```

```
surv_time = surv_time$eventtime
status <- rep(1, n)
time <- surv_time
y <- cbind(time = time, status = status)
```

Statistical Metrics Function

```
fdp = function(selected) sum(beta[selected] == 0) / max(1, length(selected))
power = function(selected) sum(beta[selected] != 0) / sum(beta != 0)
aupr = function(statistic) pr.curve(scores.class0 = statistic,
                                    weights.class0 = true_labels,
                                    curve = FALSE)$auc.integral
```

Nullstrap

```
result_Nullstrap <- nullstrap_filter(
  X, y, fdr_value = 0.1, best_lambda = NULL, B_reps = NULL, model_type = "cox"
)
```

```
Loading required package: Matrix

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

Loaded glmnet 4.1-8

Warning: package 'eha' was built under R version 4.4.1

Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
Warning in densfun(x, parm[1], parm[2], ...): NaNs produced
```

```
Nullstrap_FDR <- length(which(result_Nullstrap$statistic[setdiff(1:p, Signal_index)] >=
                              result_Nullstrap$threshold)) / max(length(result_Nullstrap$s
Nullstrap_Power <- length(which(result_Nullstrap$statistic[Signal_index] >=
                                result_Nullstrap$threshold)) / length(Signal_index)
Nullstrap_AUPR <- aupr(result_Nullstrap$statistic)
cat("Nullstrap FDR:", Nullstrap_FDR, "\n")
```

```
Nullstrap FDR: 0
```

```
cat("Nullstrap Power:", Nullstrap_Power, "\n")
```

```
Nullstrap Power: 0.9333333
```

```
cat("Nullstrap AUPR:", Nullstrap_AUPR, "\n")
```

```
Nullstrap AUPR: 0.9865315
```

Model-X

```
mx_knockoff = create.second_order(X)
mx_statistic = stat.glmnet_coefdiff(X, mx_knockoff, y,
                                    family = "cox")
```

```
Warning in glmnet(x, y, weights = weights, offset = offset, lambda = lambda, :
Cox model has no intercept
```

```
mx_thres = knockoff.threshold(mx_statistic, fdr = 0.1)
mx_selected = which(mx_statistic >= mx_thres)
mx_FDR = fdp(mx_selected)
mx_Power = power(mx_selected)
mx_AUPR = aupr(abs(mx_statistic))
cat("Model-X FDR:", mx_FDR, "\n")
```

```
Model-X FDR: 0
```

```
cat("Model-X Power:", mx_Power, "\n")
```

```
Model-X Power: 0
```

```
cat("Model-X AUPR:", mx_AUPR, "\n")
```

```
Model-X AUPR: 0.9478386
```

Fixed-X

```
fx_knockoff = create.fixed(X)$Xk
fx_statistic = stat.glmnet_lambdasmax(X, fx_knockoff, y,
                                      family = "cox")
```

```
Warning in glmnet::glmnet(X, y, lambda = lambda, intercept = intercept, : Cox
model has no intercept
```

```
fx_thres = knockoff.threshold(fx_statistic, fdr = 0.1)
fx_selected = which(fx_statistic >= fx_thres)
fx_FDR = fdp(fx_selected)
fx_Power = power(fx_selected)
fx_AUPR = aupr(abs(fx_statistic))
cat("Fixed-X FDR:", fx_FDR, "\n")
```

```
Fixed-X FDR: 0
```

```
cat("Fixed-X Power:", fx_Power, "\n")
```

```
Fixed-X Power: 0
```

```
cat("Fixed-X AUPR:", fx_AUPR, "\n")
```

```
Fixed-X AUPR: 1
```