Package 'CoxMK'

May 7, 2026

Type Package

Title A Model-X Knockoff Method for Genome-Wide Survival Association Analysis

Version 0.1.1

Author Yang Chen [aut, cre], Contributors [ctb]

Maintainer Yang Chen <yangchen5@stu.scu.edu.cn>

Description A genome-wide survival framework that integrates sequential conditional independent tuples and saddlepoint approximation method, to provide SNP-level false discovery rate control while improving power, particularly for biobank-scale survival analyses with low event rates. The method is based on model-X knockoffs as described in Barber and Candes (2015) <doi:10.1214/15-AOS1337> and fast survival analysis methods from Bi et al. (2020) <doi:10.1016/j.ajhg.2020.06.003>. A shrinkage algorithmic leveraging accelerates multiple knockoffs generation in large genetic cohorts. This CRAN version uses standard Cox regression for association testing. For enhanced performance on very large datasets, users may optionally install the 'SPACox' package from GitHub which provides saddlepoint approximation methods for survival analysis.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports Matrix, survival, irlba, stats, utils, gdsfmt, BEDMatrix

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2025-09-03 08:00:29 UTC

Contents

`calculate_w_statistics` *Calculate W Statistics for Knockoff Analysis*

Description

Computes W statistics by comparing test statistics from original variables with those from their knockoff counterparts. These statistics are used for variable selection with FDR control.

Usage

```
calculate_w_statistics(t_orig, t_knock, method = "median")
```

Arguments

`t_orig`	Vector of test statistics for original variables
`t_knock`	Vector or list of test statistics for knockoff variables. If a list, should contain M vectors of the same length as t_orig.
`method`	Method for computing W statistics: • "difference": W_j = T_j - max(T_{j,k}) (default) • "median": Uses Model-X knockoff median-based statistics • "ratio": W_j = T_j / max(T_{j,k})

Value

Vector of W statistics for variable selection

Examples

```
# Example with difference method
t_orig <- c(5.2, 3.1, 8.7, 2.4, 6.9)
t_knock <- list(
  c(2.1, 4.2, 3.3, 1.8, 2.9),
  c(1.9, 3.8, 4.1, 2.2, 3.1)
)

w_median <- calculate_w_statistics(t_orig, t_knock, method = "median")
```

```
w_diff <- calculate_w_statistics(t_orig, t_knock, method = "difference")
```

CoxMK *CoxMK: Cox Regression with Multiple Knockoffs*

Description

Main interface functions for Cox regression analysis with Multiple knockoffs. This package provides a complete workflow for survival analysis with variable selection using the multiple knockoffs methodology.

The workflow follows four main steps: 1. **Generate Knockoffs**: Create knockoff variables using `create_knockoffs` 2. **Fit Null Model**: Fit null Cox model using `fit_null_cox_model` 3. **Perform Testing**: Conduct association testing using `perform_association_testing` 4. **Apply Filter**: Select variables using `knockoff_filter`

Main Functions

- `cox_knockoff_analysis` - Complete knockoff analysis workflow
- `create_knockoffs` - Step 1: Generate knockoff variables
- `fit_null_cox_model` - Step 2: Fit null Cox model for testing
- `perform_association_testing` - Step 3: Perform association testing
- `knockoff_filter` - Step 4: Apply knockoff filter for variable selection

---

cox_knockoff_analysis *Complete Cox Knockoff Analysis Workflow*

Description

Performs a complete Multiple knockoff analysis following the four-step workflow: 1. Generate knockoff variables from PLINK data and save to GDS format 2. Fit null Cox model using optimized Cox regression for large-scale analysis 3. Perform SPA testing using original and knockoff variables 4. Apply knockoff filter for variable selection with FDR control

Usage

```
cox_knockoff_analysis(
  plink_prefix,
  time,
  status,
  covariates = NULL,
  sample_ids = NULL,
  null_model = NULL,
  gds_file = NULL,
```

```
  M = 5,
  fdr = 0.05,
  method = "median",
  output_dir = NULL
)
```

Arguments

plink_prefix	Character string. Path prefix for PLINK files (.bed, .bim, .fam)
time	Numeric vector. Survival times
status	Numeric vector. Censoring indicator (1 = event, 0 = censored)
covariates	Data frame or matrix. Covariate data (optional)
sample_ids	Character vector. Sample IDs to match with genetic data (optional)
null_model	Fitted Cox model object for null hypothesis (optional)
gds_file	Character string. Path to pre-generated GDS file with knockoffs (optional)
M	Integer. Number of knockoff copies to generate (default: 5)
fdr	Numeric. Target false discovery rate (default: 0.05)
method	Character. Method for computing W statistics ("median", "difference", "ratio")
output_dir	Character string. Directory to save intermediate results (default: NULL, uses tempdir())

Value

List containing:

selected_vars	Indices of selected variables
W_stats	W statistics for all variables
threshold	Knockoff threshold used
gds_file	Path to GDS file used
null_model	Fitted null Cox model
test_results	SPA test results

List containing:

- W_stats - Vector of W statistics for each variant
- selected_vars - Indices of selected variants
- knockoffs - Generated knockoff matrix (if gds_file not provided)
- summary - Summary statistics of the analysis

Examples

```
# Simple workflow example
# Load example data
extdata_path <- system.file('extdata', package = 'CoxMK')
plink_prefix <- file.path(extdata_path, 'sample')
pheno_data <- read.table(file.path(extdata_path, 'tte_phenotype.txt'),
                         header = TRUE, stringsAsFactors = FALSE)
covar_data <- read.table(file.path(extdata_path, 'covariates.txt'),
                         header = TRUE, stringsAsFactors = FALSE)
covar_data <- covar_data[, c("age", "sex", "bmi", "smoking")]

# Run complete analysis
result <- cox_knockoff_analysis(
  plink_prefix = plink_prefix,
  time = pheno_data$time,
  status = pheno_data$status,
  covariates = covar_data,
  M = 3,
  fdr = 0.1
)

# View results
print(result$selected_vars)
print(result$summary)
```

`create_knockoffs` *Create Multiple Knockoffs for Genetic Data*

Description

Generate knockoff variables for genotype data using the Multiple knockoff method with leveraging scores and clustering specifically optimized for genetic variant data.

Usage

```
create_knockoffs(
  X,
  pos,
  chr_info = NULL,
  sample_ids = NULL,
  M = 5,
  save_gds = TRUE,
  output_dir = NULL,
  start = NULL,
  end = NULL,
  corr_max = 0.75,
  maxN_neighbor = Inf,
```

```
  maxBP_neighbor = 1e+05,
  n_AL = floor(10 * nrow(X)^(1/3) * log(nrow(X))),
  thres_ultrarare = 25,
  R2_thres = 1,
  prob_eps = 1e-12,
  irlba_maxit = 1500
)
```

Arguments

X	A sparse matrix (n x p) of genotype data where n is the number of samples and p is the number of SNPs. Typically coded as 0, 1, 2 for genotype dosages.
pos	A numeric vector of SNP positions (in base pairs) for linkage disequilibrium-aware knockoff generation.
chr_info	Optional chromosome information. Can be either: (1) A data frame with chromosome information from BIM file containing a column named "chr" or "CHR" with chromosome numbers, or (2) A vector of chromosome numbers directly. Chromosome information will be automatically extracted.
sample_ids	A character vector of sample IDs (default: NULL, will generate)
M	Number of knockoff copies to generate (default: 5). More copies can improve statistical power but increase computational cost.
save_gds	Whether to save knockoffs to GDS format (default: TRUE)
output_dir	Directory to save GDS files (default: NULL, uses tempdir())
start	Start position for file naming (default: min(pos))
end	End position for file naming (default: max(pos))
corr_max	Maximum correlation threshold for clustering variants (default: 0.75). Higher values create fewer, larger clusters.
maxN_neighbor	Maximum number of neighboring variants to consider for each variant (default: Inf).
maxBP_neighbor	Maximum base pair distance to consider variants as neighbors (default: 100,000 bp).
n_AL	Number of samples to use for adaptive lasso fitting (default: automatically determined based on sample size).
thres_ultrarare	Minimum minor allele count threshold for variant inclusion (default: 25).
R2_thres	R-squared threshold for model fitting (default: 1).
prob_eps	Minimum probability value to prevent numerical issues (default: 1e-12).
irlba_maxit	Maximum iterations for truncated SVD (default: 1500).

Value

If save_gds is TRUE, returns the path to the saved GDS file. Otherwise, returns a list of M matrices, each of the same dimensions as X, containing knockoff variables.

`fit_cox_model_from_files` *Step 2: Fit Cox Model from Files*

Description

Implements Step 2 of the CoxMK workflow: fitting a null Cox proportional hazards model by reading phenotype and covariate data from files. This function is designed for batch processing and large-scale analysis where data is stored in separate files.

Usage

```
fit_cox_model_from_files(
  phenotype_file,
  covariate_file,
  output_file = NULL,
  use_spacox = TRUE
)
```

Arguments

`phenotype_file`	Path to CSV file with columns: IID, time, status
`covariate_file`	Path to CSV file with columns: IID, covar1, covar2, ...
`output_file`	Path to RDS file to save the fitted null model (default: temporary directory)
`use_spacox`	Legacy parameter, kept for compatibility (ignored)

Value

Invisible path to the output file

Examples

```
# Prepare example data files
pheno_data <- data.frame(
  IID = paste0("ID", 1:100),
  time = rexp(100, 0.1),
  status = rbinom(100, 1, 0.3)
)
covar_data <- data.frame(
  IID = paste0("ID", 1:100),
  age = rnorm(100, 50, 10),
  sex = rbinom(100, 1, 0.5)
)

# Use temporary directory for file operations to comply with CRAN policies
temp_dir <- tempdir()
pheno_file <- file.path(temp_dir, "phenotype.csv")
covar_file <- file.path(temp_dir, "covariates.csv")
```

```
output_file <- file.path(temp_dir, "null_model.rds")

write.csv(pheno_data, pheno_file, row.names = FALSE)
write.csv(covar_data, covar_file, row.names = FALSE)

# Step 2: Fit null Cox model from files
fit_cox_model_from_files(
  phenotype_file = pheno_file,
  covariate_file = covar_file,
  output_file = output_file
)

# Load the fitted model for Step 3
model_info <- readRDS(output_file)
```

`fit_null_cox_model` *Fit Null Cox Model*

Description

Fit Null Cox Model

Usage

```
fit_null_cox_model(time, status, covariates = NULL, use_spacox = TRUE)
```

Arguments

`time`	Numeric vector of survival times
`status`	Numeric vector of censoring indicators
`covariates`	Optional covariate data frame
`use_spacox`	Logical, whether to use SPACox (ignored in CRAN version)

Value

Fitted Cox model object

knockoff_filter *Apply Knockoff Filter for Variable Selection*

Description

Applies the knockoff filter to select variables while controlling the false discovery rate (FDR) at a specified level.

Usage

```
knockoff_filter(W, fdr = 0.1, offset = 1)
```

Arguments

W	Vector of W statistics from calculate_w_statistics
fdr	Target false discovery rate (default: 0.1)
offset	Offset parameter for knockoff filter (default: 1)

Value

Vector of indices of selected variables

Examples

```
# Generate some example W statistics
W <- c(2.1, -0.5, 3.8, -1.2, 4.5, 0.3, -2.1, 1.9)

# Apply knockoff filter
selected <- knockoff_filter(W, fdr = 0.1)
print(selected)  # Indices of selected variables
```

load_knockoff_gds *Load Knockoff Data from GDS File*

Description

Load Knockoff Data from GDS File

Usage

```
load_knockoff_gds(gds_file)
```

Arguments

gds_file	Character string. Path to the GDS file containing knockoff data

`perform_association_testing` *Perform Association Testing*

Description

Perform Association Testing

Usage

```
perform_association_testing(X, null_model)
```

Arguments

`X`	Genotype matrix
`null_model`	Fitted null Cox model

Value

List with test statistics and p-values

Index