Package 'carSurv'

May 8, 2026

Title Correlation-Adjusted Regression Survival (CARS) Scores

Version 1.0.0

Author Thomas Welchowski

Maintainer Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Description Contains functions to estimate the Correlation-Adjusted Regression Survival (CARS) Scores. The method is described in Welchowski, T. and Zuber, V. and Schmid, M., (2018), Correlation-Adjusted Regression Survival Scores for High-Dimensional Variable Selection, <doi:10.48550/arXiv.1802.08178>.

Depends R (>= 3.2.2)

Imports Rcpp, survival, corpcor, mboost, fdrtool

Suggests microbenchmark, mvtnorm

License GPL-3

LazyLoad yes

LinkingTo Rcpp

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-02-26 12:34:39 UTC

Contents

carSurv-package *Correlation-Adjusted Regression Survival Scores*

Description

Contains functions to estimate the Correlation-Adjusted Regression Survival (CARS) Scores. The main function is carSurvScore, which estimates CARS scores of each variable. The higher the absolute values of CARS scores, the higher the variable importance. Additionally there is the function carVarSelect to select cut-off thresholds to separate variables associated with survival from noise variables. There are two possible cut-off threshold options: False non-discovery rate q-values and empirical quantiles of the raw scores.

Details

Package: carSurv

Type: Package

Version: 1.0.0

Date: 2018-02-24

License: GPL-3

Author(s)

Thomas Welchowski (Maintainer) <welchow@imbie.meb.uni-bonn.de>

References

Welchowski, T. and Zuber, V. and Schmid, M., (2018), Correlation-Adjusted Regression Survival Scores for High-Dimensional Variable Selection, <arXiv:1802.08178>

Zuber, V. and Strimmer, K., (2011), High-Dimensional Regression and Variable Selection Using CAR Scores, Statistical Applications in Genetics and Molecular Biology

Schaefer, J. and Strimmer, K., (2005), A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics, Statistical Applications in Genetics and Molecular Biology

Van der Laan, M. J. and Robins, J. M., (2003), Unified Methods for Censored Longitudinal Data and Causality, Springer Series in Statistics Strimmer, K., (2008), A unified approach to false discovery rate estimation, BMC Bioinformatics

carSurvScore *Estimate Correlation-Adjusted Regression Survival (CARS) Scores*

Description

Estimates CARS scores. CARS scores measure the relative importance of each variable with respect to the survival times adjusted by IPC weighting.

Usage

```
carSurvScore(obsTime, obsEvent, X, maxIPCweight = 10, denom = "1/n")
```

Arguments

obsTime	Observed time points of a right censored survival process (numeric vector).
obsEvent	Observed event indicator of right censored survival process (numeric vector) 0=no event, 1=event
X	Data of design variables (numeric matrix). Must be already encoded.
maxIPCweight	Specifies the maximum possible weight, to ensure numerical stability.
denom	Specifies the denominator of the weighted sums. Two options are available: The default value "1/n" uses the sample size as denominator. Option "sum_w" uses the sum of all IPC weights in the denominator.

Details

CARS scores are defined as theta=P_X^(-1/2) P_(X, log(T)). The term P_X^(-1/2) is the inverse square root of the correlation matrix between covariates X. P_(X, log(T)) is the correlation vector between covariates and the logarithmic survival time adjusted for censoring by IPC weighting.

Value

Estimated CAR survival score of each variable (numeric vector).

Note

It is recommended to use default setting "denom=1/n" because in this case CARS scores are consistent. Furthermore the simulation results of "1/n" show lower root mean squared error of CARS scores with respect to the true parameter.

Author(s)

Thomas Welchowski

References

Welchowski, T. and Zuber, V. and Schmid, M., (2018), Correlation-Adjusted Regression Survival Scores for High-Dimensional Variable Selection, <arXiv:1802.08178>

Zuber, V. and Strimmer, K., (2011), High-Dimensional Regression and Variable Selection Using CAR Scores, Statistical Applications in Genetics and Molecular Biology

Schaefer, J. and Strimmer, K., (2005), A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics, Statistical Applications in Genetics and Molecular Biology

Van der Laan, M. J. and Robins, J. M., (2003), Unified Methods for Censored Longitudinal Data and Causality, Springer Series in Statistics

See Also

carVarSelect

Examples

```
########################################
# Simulate accelerated, failure time model

# Generate multivariate normal distributed covariates
noObs <- 100
noCovar <- 10
library(mvtnorm)
set.seed(190)
X <- rmvnorm(noObs, mean=rep(0, noCovar), sigma=diag(noCovar))

# Generate gamma distributed survival times
# Only the first 5 variables have an influence
eta <- 1 - 2 * X[,1] - X[,2] + X[,3] +
0.5 * X[,4] + 1.5 * X[,5]

# Function to generate survival times
genSurv <- function(x) {
set.seed(x)
rgamma(1, shape=2, scale=exp(eta[x]))
}

# Generate survival times
survT <- sapply(1:length(eta), genSurv)

# Generate exponential distributed censoring times
censT <- rexp(noObs, rate=1)

# Calculate event indicator
eventInd <- ifelse(survT <= censT, 1, 0)

# Calculate observed times
obsTime <- survT
obsTime[survT > censT] <- censT [survT > censT]
```

```
# Estimate CAR scores
carScores <- carSurvScore(obsTime=obsTime, obsEvent=eventInd, X=X)
carScores
```

`carVarSelect` *Variable selection with Correlation-Adjusted Regression Survival (CARS) Scores*

Description

Computes the CARS scores and selects significant variables. If the false non discovery rate (fndr) approach is used, significant and null variables are distinguished by an a priori defined q-value.

Usage

```
carVarSelect(carSurvScores, method = "fndr", plotDiag = FALSE,
  threshold = 0.05)
```

Arguments

`carSurvScores`	Estimated CAR survival scores of each variable (numeric vector). See function `carSurvScore`.
`method`	Gives the variable selection procedure. Default is "fndr", which is based on the false non-discovery-rate. The other option is "threshold", which selects only variables above a given empirical quantile.
`plotDiag`	Should diagnostic plots of the null distribution be plotted? Default is FALSE (logical scalar).
`threshold`	If method="threshold", then this specifies the quantile threshold of the CAR survival scores. Every score above this threshold is then a significant variable.

Value

Index giving the significant variables of the original data (integer vector).

Note

The quality of estimated, significant variables depends on the sample size and on the number of variables.

Author(s)

Thomas Welchowski

References

Strimmer, K., (2008), A unified approach to false discovery rate estimation, BMC Bioinformatics

See Also

carSurvScore

Examples

```
########################################
# Simulate accelerated, failure time model

# Generate multivariate normal distributed covariates
noObs <- 100
noCovar <- 250
library(mvtnorm)
set.seed(7903)
X <- rmvnorm(noObs, mean=rep(0, noCovar), sigma=diag(noCovar))

# Generate gamma distributed survival times
# Only the first 5 variables have an influence
eta <- 1 - 2 * X[,1] - X[,2] + X[,3] +
0.5 * X[,4] + 1.5 * X[,5]

# Function to generate survival times
genSurv <- function(x) {
set.seed(x)
rgamma(1, shape=2, scale=exp(eta[x]))
}

# Generate survival times
survT <- sapply(1:length(eta), genSurv)

# Generate exponential distributed censoring times
censT <- rexp(noObs, rate=1)

# Calculate event indicator
eventInd <- ifelse(survT <= censT, 1, 0)

# Calculate observed times
obsTime <- survT
obsTime[survT > censT] <- censT [survT > censT]

# Conduct variable selection using fndr
carScores <- carSurvScore(obsTime=obsTime, obsEvent=eventInd, X=X)
selectedVar <- carVarSelect(carSurvScores=carScores)
selectedVar

# Check true positive and true negative rate
TPR <- mean(c(1:5) %in% selectedVar)
TNR <- mean(c(6:250) %in% setdiff(6:250, selectedVar))
perf <- TPR + TNR -1
perf
```

`weightedCovarRcpp` *Estimate weighted covariance*

Description

Efficient C implementation of the sample covariance estimator. The denominator is defined as the sum of all weights.

Usage

```
weightedCovarRcpp(x, y, w)
```

Arguments

`x`	Covariate without weighting (numeric vector).
`y`	Response. The mean of the response contains weights (numeric vector).
`w`	Weights for averaging (numeric vector).

Value

Weighted variance (numeric scalar).

Note

If all weights are set to 1, the denominator is identical to n. There are no safety checks of input arguments.

Author(s)

Thomas Welchowski

Examples

```
# Simulate two random vectors
set.seed(3975)
x <- rnorm(100)
set.seed(-3975)
y <- rnorm(100)
# Calculate variance with standard R function
# Rescaling ensures that both calculations use same denominator "n"
covarEst <- cov(x, y) * (100-1) / 100
# Calculate weighted variance with equal weights
equalW <- rep(1, 100)
weightCovarEst <- weightedCovarRcpp(x=x, y=y, w=equalW)
# Output comparison
all.equal(covarEst, weightCovarEst)
# Runtime comparison
library(microbenchmark)
```

```
microbenchmark(Default=cov(x, y), New=weightedCovarRcpp(x=x, y=y, w=equalW), times=25)
# -> New method is multiple times faster
```

`weightedCovarRcppN` *Estimate weighted covariance*

Description

Efficient C implementation of the sample covariance estimator. The denominator is defined as the sample size.

Usage

```
weightedCovarRcppN(x, y, w)
```

Arguments

`x`	Covariate without weighting (numeric vector).
`y`	Response. The mean of the response contains weights (numeric vector).
`w`	Weights for averaging (numeric vector).

Value

Weighted variance (numeric scalar).

Note

There are no safety checks of input arguments.

Author(s)

Thomas Welchowski

Examples

```
# Simulate two random vectors
set.seed(3975)
x <- rnorm(100)
set.seed(-3975)
y <- rnorm(100)
# Calculate variance with standard R function
# Rescaling ensures that both calculations use same denominator "n"
covarEst <- cov(x, y) * (100-1) / 100
# Calculate weighted variance with equal weights
equalW <- rep(1, 100)
weightCovarEst <- weightedCovarRcppN(x=x, y=y, w=equalW)
# Output comparison
all.equal(covarEst, weightCovarEst)
# Runtime comparison
```

```
library(microbenchmark)
microbenchmark(Default=cov(x, y), New=weightedCovarRcpp(x=x, y=y, w=equalW), times=25)
# -> New method is multiple times faster
```

`weightedVarRcpp` *Estimate weighted variance*

Description

Efficient C implementation of the sample variance estimator. The denominator is defined as sum of all weights.

Usage

```
weightedVarRcpp(y, w)
```

Arguments

`y`	Response. The mean of the response contains weights (numeric vector).
`w`	Weights for averaging (numeric vector).

Value

Weighted variance (numeric scalar).

Note

If all weights are set to 1, the denominator is identical to n. There are no safety checks of input arguments.

Author(s)

Thomas Welchowski

Examples

```
# Simulate a random vector
set.seed(3975)
x <- rnorm(100)
# Calculate variance with standard implementation
# Rescaling ensures that both calculations use same denominator "n"
varEst <- var(x) * (100-1) / 100
# Calculate weighted variance with equal weights
equalW <- rep(1/100, 100)
weightVarEst <- weightedVarRcpp(y=x, w=equalW)
# Output comparison
all.equal(varEst, weightVarEst)
# Runtime comparison
library(microbenchmark)
```

```
microbenchmark(Default=var(x), New=weightedVarRcppN(y=x, w=equalW), times=25)
# -> New method is multiple times faster
```

`weightedVarRcppN` *Estimate weighted variance*

Description

Efficient C implementation of the sample variance estimator. The denominator is defined as the sample size.

Usage

```
weightedVarRcppN(y, w)
```

Arguments

`y`	Response. The mean of the response contains weights (numeric vector).
`w`	Weights for averaging (numeric vector).

Value

Weighted variance (numeric scalar).

Note

There are no safety checks of input arguments.

Author(s)

Thomas Welchowski

Examples

```
# Simulate a random vector
set.seed(3975)
x <- rnorm(100)
# Calculate variance with standard implementation
varEst <- var(x) * (100-1) / 100
# Calculate weighted variance with equal weights
equalW <- rep(1, 100)
weightVarEst <- weightedVarRcppN(y=x, w=equalW)
# Output comparison
all.equal(varEst, weightVarEst)
# Runtime comparison
library(microbenchmark)
microbenchmark(Default=var(x), New=weightedVarRcppN(y=x, w=equalW), times=25)
# -> New method is multiple times faster
```

Index