Package 'PoSI'

July 21, 2025

Type Package

Title Valid Post-Selection Inference for Linear LS Regression

Version 1.1

Date 2020-11-04

Description In linear LS regression, calculate for a given design matrix the multiplier K of coefficient standard errors such that the confidence intervals [b - K*SE(b), b + K*SE(b)] have a guaranteed coverage probability for all coefficient estimates b in any submodels after performing arbitrary model selection.

Suggests MASS

License GPL-3

NeedsCompilation no

Repository CRAN

Author Andreas Buja [aut],
Kai Zhang [aut],
Wan Zhang [cre]

Maintainer Wan Zhang <wanz63@live.unc.edu>

Date/Publication 2020-11-18 10:30:07 UTC

Contents

PoSI-package 2
PoSI 3

Index **8**

PoSI-package *Valid Post-Selection Inference for Linear LS Regression*

Description

In linear LS regression, calculate for a given regressor matrix the multiplier K of coefficient standard errors such that the confidence intervals [b - K*SE(b), b + K*SE(b)] have a guaranteed coverage probability for all coefficient estimates b in any submodels after performing arbitrary model selection.

Details

Package:	PoSI
Type:	Package
Version:	1.1
Date:	2020-10-24
License:	GPL-3

Author(s)

Andreas Buja and Kai Zhang

Maintainers: Andreas Buja <buja.at.wharton@gmail.com>, Kai Zhang <zhangk@email.unc.edu> and Wan Zhang <wanz63@live.unc.edu>

References

"Valid Post-Selection Inference," Berk, R., Brown, L., Buja, A., Zhang, K., Zhao, L., The Annals of Statistics, 41 (2), 802–837~(2013).

See Also

lm, link{model.matrix}

Examples

```
data(Boston, package="MASS")
summary(PoSI(Boston[,-14]))
```

PoSI *Valid Post-Selection Inference for Linear LS Models*

Description

Used in calculating multipliers K of standard errors in linear LS reqression such that the confidence intervals

[b - K*SE(b), b + K*SE(b)]

have guaranteed coverage probabilities for all coefficient estimates b in any submodel arrived at after performing arbitrary model selection. The actual multipliers `K` are calculated by `summary`; `PoSI` returns an object of class "PoSI".

Usage

```
PoSI(X, modelSZ = 1:ncol(X), center = T, scale = T, verbose = 1,
     Nsim = 1000, bundleSZ = 100000, eps = 1e-08)

## S3 method for class 'PoSI'
summary(object, confidence = c(0.95, 0.99), alpha = NULL,
                        df.err = NULL, eps.PoSI = 1e-06, digits = 3, ...)
```

Arguments

`X`	a regressor matrix as returned, for example, by the function `model.matrix` when applied to a linear model object from the function `lm`; data frames are coerced to matrices
`modelSZ`	the model sizes to be included (default: `1:ncol(X)`). This argument permits 'sparse PoSI' with, e.g., `modelSZ=1:5` when only models up to size 5 have been searched, or 'rich PoSI' with, e.g., `modelSZ=(ncol(X)-2):ncol(X)` when only the removal of up to two regressors has been tried.
`center`	whether to center the columns of `X` (boolean, default: TRUE, in which case the intercept will be removed)
`scale`	whether to standardize the columns of X (boolean, default: TRUE; prevents problems from columns with vastly differing scales)
`verbose`	0: no printed reports during computations; 1: report bundle completion (default); 2: report each processed submodel (for debugging with small `ncol(X)`).
`Nsim`	the number of tests being simulated (default: 1000). `PoSI` is partly simulation-based; increase `Nsim` for greater precision at the cost of increased run time.
`bundleSZ`	number of tests to be processed simultaneously (default: 100000). Larger bundles are computationally more efficient but require more memory.
`eps`	threshold below which singular values of `X` will be considered to be zero (default: 1E-8). In cases of highly collinear columns in `X` this threshold determines the effective dimension of the column space of `X`.

object	an object of class "PoSI" as returned by the function PoSI
confidence	a numeric vector of values between 0 and 1 containing the confidence levels for which multipliers of standard error should be provided (default: c(.95,.99))
alpha	if specified, sets confidence = 1-alpha. (This argument is redundant with confidence; only one should be specified.)
df.err	error degrees of freedom for t-tests (default: NULL, performs z-tests)
eps.PoSI	precision to which standard error multipliers are computed (default: 1e-06)
digits	number of significant digits to which standard error multipliers are rounded (default: 3)
...	(other arguments)

Details

Example of use of PoSI multipliers: In the Boston Housing data shown below, the 0.95 multiplier is 3.593. If after arbitrary variable selection we decide, for example, in favor of the submodel

summary(lm(medv ~ rm + nox + dis + ptratio + lstat, data=Boston))

the regressor rm (e.g.) has a coefficient estimate of 4.16 with a standard error of 0.41; hence the 0.95 PoSI confidence interval is found by

4.16 + c(-1,+1) * 3.593 * 0.41

which is (2.69, 5.63) after rounding. Similar intervals can be formed for any regressor in any submodel. The resulting confidence procedure has a 0.95 family-wise guarantee of containing the true coefficient even after arbitrary variable selection in any submodel one might arrive at.

The computational limitations of the PoSI method are in the exponential growth of the number of t/z-tests that are being computed:

(1) If p=ncol(X) and all submodels are being searched (modelSZ=1:p), the number of tests is p*2^(p-1). Example: p=20; modelSZ=1:20 ==> # tests = 10,485,760

(2) If only models of sizes modelSZ=m are being searched, the number of tests is sum(m*choose(p,m)). Example: p=50; m=1:5 ==> # tests = 11,576,300

Thus limiting PoSI to small submodel sizes such as modelSZ=1:5 ("sparse PoSI") puts larger p=ncol(X) within reach.

PoSI computations are partly simulation-based and require specification of a number Nsim of random unit vectors to be sampled in the column space of X. Large Nsim yields greater precision but requires more memory. The memory demands can be lowered by decreasing bundleSZ at the cost of some efficiency. bundleSZ determines how many tests are simultaneously processed.

Value

PoSI returns an object of class "PoSI" whose only use is to be the first argument to the function summary.

summary returns a matrix containing in its first column the two-sided PoSI standard error multipliers K for the specified confidence levels or Type-I error probabilities. Additionally, in the second and third column, it returns standard error multipliers based on the Bonferroni and Scheffe methods which are more conservative than the PoSI method: PoSI < Bonferroni < Scheffe (sometimes Bonferroni > Scheffe).

References

"Valid Post-Selection Inference," by Berk, R., Brown, L., Buja, A., Zhang, K., Zhao,L., The Annals of Statistics, 41 (2), 802-837 (2013).

Examples

```
## Not run:
# Boston Housing data from http://archive.ics.uci.edu/ml/datasets/Housing
data(Boston, package="MASS")
UL.Boston <- PoSI(Boston[,-14]) # 4.7 sec (**)
summary(UL.Boston)
    ##     K.PoSI K.Bonferroni K.Scheffe
    ## 95%  3.593        4.904     4.729
    ## 99%  4.072        5.211     5.262

## End(Not run)


# Just 1 predictor:
X.1 <- as.matrix(rnorm(100))
UL.max.1 <- PoSI(X.1)
summary(UL.max.1)             # Assuming sigma is known
    ##     K.PoSI K.Bonferroni K.Scheffe
    ## 95%  1.960        1.960     1.960
    ## 99%  2.576        2.576     2.576
summary(UL.max.1, df.err=4)  # sigma estimated with 4 dfs
    ##     K.PoSI K.Bonferroni K.Scheffe
    ## 95%  2.776        2.776     2.776
    ## 99%  4.604        4.604     4.604


# small N and automatic removal of intercept:
p <- 6;  N <- 4
X.small <- cbind(1,matrix(rnorm(N*p), ncol=p))
UL.max.small <- PoSI(X.small, modelSZ=c(4,3,1), Nsim=1000, bundleSZ=5, verbose=2)
summary(UL.max.small, df.err=4)
    ##     K.PoSI K.Bonferroni K.Scheffe
    ## 95%  4.226        9.256     4.447
    ## 99%  6.731       13.988     7.077

## Not run:
# Orthogonal regressors:
p <- 10; N <- 10
X.orth <- qr.Q(qr(matrix(rnorm(p*N), ncol=p)))
UL.max.orth <- PoSI(X.orth, Nsim=10000) # 2.8 sec (**)
summary(UL.max.orth)
    ##     K.PoSI K.Bonferroni K.Scheffe
    ## 95%  3.448        4.422     4.113
    ## 99%  3.947        4.758     4.655

## End(Not run)
```

```
## Not run:
# Large p=50, small N=20, models up to size 4:  1.3min
p <- 50;  N <- 20
X.p50.N20 <- matrix(rnorm(p*N), ncol=p)
UL.max.p50.N20 <- PoSI(X.p50.N20, Nsim=1000, bundleSZ=100000, modelSZ=1:4) # 1.3 min (*)
summary(UL.max.p50.N20)
##     K.PoSI K.Bonferroni K.Scheffe
## 95%  4.309        5.448     5.490
## 99%  4.769        5.728     6.016

## End(Not run)

## Not run:
# The following is modeled on a real data example:
p <- 84;  N <- 2758
X.84 <- matrix(rnorm(p*N), ncol=p)
# --- (1) Rich submodels: sizes m=84 and m=83 with more simulations (10,000) for precision
UL.max.84 <- PoSI(X.84, Nsim=1000, bundleSZ=10000, modelSZ=c(p-1,p))  # 2 sec (*)
summary(UL.max.84)
    ##     K.PoSI K.Bonferroni K.Scheffe
    ## 95%  3.494        4.491    10.315
    ## 99%  3.936        4.823    10.819

## End(Not run)

## Not run:
# --- (2) Sparse submodels: p=84, model size m=4, in p=d=84 dimensions:
# WARNING:  17 minutes (*)
UL.max.84.4 <- PoSI(X.84, Nsim=1000, bundleSZ=100000, modelSZ=4)
summary(UL.max.84.4)
    ##     K.PoSI K.Bonferroni K.Scheffe
    ## 95%  3.553        5.804    10.315
    ## 99%  3.966        6.068    10.819
summary(UL.max.84.4, df.err=2758-84-1)
    ##     K.PoSI K.Bonferroni K.Scheffe
    ## 95%  3.557        5.823    10.338
    ## 99%  3.972        6.089    10.855

## End(Not run)


## Not run:
# Big experiment: full large PoSI for p=20
# WARNING: 13 minutes (*)
p <- 20;  N <- 1000
X.p20 <- matrix(rnorm(N*p), ncol=p)
UL.max.p20 <- PoSI(X.p20, bundleSZ=100000)
summary(UL.max.p20)
    ##     K.PoSI K.Bonferroni K.Scheffe
    ## 95%  3.163        5.855     5.605
    ## 99%  3.626        6.117     6.129
summary(UL.max.p20, df.err=1000-21)
    ##     K.PoSI K.Bonferroni K.Scheffe
```

```
    ## 95%  3.171        5.908     5.624
    ## 99%  3.638        6.177     6.160

## End(Not run)


## Not run:
# Big experiment: sparse large PoSI with p=50 and m=1:5:
## WARNING: 22 minutes (*)
p <- 50;  N <- 1000;  m <- 1:5
X.p50 <- matrix(rnorm(N*p), ncol=p)
UL.max.p50.m5 <- PoSI(X.p50, bundleSZ=100000, modelSZ=m)
summary(UL.max.p50.m5)
    ##    K.PoSI K.Bonferroni K.Scheffe
    ## 95%  3.548        5.871     8.216
    ## 99%  4.041        6.133     8.727

## End(Not run)

## Not run:
# Exchangeable Designs:
# function to create exchangeable designs:
design.exch <- function(p,a) { (1-a)*diag(p) + a*matrix(1,p,p) }
# example:
p <- 12;  a <- 0.5;
X.exch <- design.exch(p=p, a=a)
UL.exch <- PoSI(X.exch, verbose=0, modelSZ=1:p) # 2 sec (**)
summary(UL.exch)
    ##    K.PoSI K.Bonferroni K.Scheffe
    ## 95%  3.635        4.750     4.436
    ## 99%  4.129        5.066     4.972

## End(Not run)


# (*) Elapsed times were measured in R version 3.1.3, 32 bit,
#     on a processor Intel(R) Core(TM), 2.9 GHz, under Windows 7.
#     2015/04/16

# (**) Elapsed times were measured in R version 4.0.2, 64 bit,
#      on a processor Intel(R) Core(TM), 1.80 GHz, under Windows 10.
#      2020/10/26
```

Index

∗ **Family-wise error**
PoSI, 3
∗ **LS Regression**
PoSI, 3
∗ **LS regression**
PoSI-package, 2
∗ **Model selection**
PoSI, 3
∗ **Post-selection inference**
PoSI, 3
∗ **family-wise error**
PoSI-package, 2
∗ **model selection**
PoSI-package, 2
∗ **post-selection inference**
PoSI-package, 2

lm, *2*

PoSI, 3
PoSI-package, 2

summary.PoSI *(*PoSI*)*, 3