Package ‘fence’

October 13, 2022

Type Package

Title Using Fence Methods for Model Selection

Version 1.0

Date 2017-05-29

Author Jiming Jiang, Jianyang Zhao, J. Sunil Rao, Thuan Nguyen

Maintainer Thuan Nguyen <nguythua@ohsu.edu>

Description This method is a new class of model selection strategies, for mixed model selection, which includes linear and generalized linear mixed models. The idea involves a procedure to isolate a subgroup of what are known as correct models (of which the optimal model is a member). This is accomplished by constructing a statistical fence, or barrier, to carefully eliminate incorrect models. Once the fence is constructed, the optimal model is selected from among those within the fence according to a criterion which can be made flexible.
References:
1. Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692.
<DOI:10.1214/07-AOS517> <https://projecteuclid.org/euclid.aos/1216237296>.
2. Jiang J., Nguyen T., Rao J.S. (2009), A Simplified Adaptive Fence Procedure. Statistics and Probability Letters, 79, 625-629.
<DOI:10.1016/j.spl.2008.10.014> <https://www.researchgate.net/publication/23991417_A_simplified_adaptive_fence_procedure>
3. Jiang J., Nguyen T., Rao J.S. (2010), Fence Method for Nonparametric Small Area Estimation. Survey Methodology, 36(1), 3-11.
<http://publications.gc.ca/collections/collection_2010/statcan/12-001-X/12-001-x2010001-eng.pdf>.
4. Jiming Jiang, Thuan Nguyen and J. Sunil Rao (2011), Invisible fence methods and the identification of differentially expressed gene sets.
Statistics and Its Interface, Volume 4, 403-415.
<http://www.intlpress.com/site/pub/files/_fulltext/journals/sii/2011/0004/0003/SII-2011-0004-0003-a014.pdf>.
5. Thuan Nguyen & Jiming Jiang (2012), Restricted fence method for covariate selection in longitudinal data analysis.
Biostatistics, 13(2), 303-314.

<DOI:10.1093/biostatistics/kxr046> <https://academic.oup.com/biostatistics/article/13/2/303/263903/Restricted-fence-method-for-covariate-selection-in>.
6. Thuan Nguyen, Jie Peng, Jiming Jiang (2014), Fence Methods for Backcross Experiments. Statistical Computation and Simulation, 84(3), 644-662.
<DOI:10.1080/00949655.2012.721885> <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3891925/>.
7. Jiang, J. (2014), The fence methods, in Advances in Statistics, Hindawi Publishing Corp., Cairo.
<DOI:10.1155/2014/830821>.
8. Jiming Jiang and Thuan Nguyen (2015), The Fence Methods, World Scientific, Singapore.
<https://www.abebooks.com/9789814596060/Fence-Methods-Jiming-Jiang-981459606X/plp>.

License BSD_2_clause + file LICENSE

Depends R (>= 2.10)

Imports MASS, stats, lme4, ggplot2, compiler, sae, fields, grDevices, snowfall, snow

Suggests pscl

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-01 08:02:18 UTC

R topics documented:

`adaptivefence` *Adaptive Fence model selection*

Description

Adaptive Fence model selection

Usage

```
adaptivefence(mf, f, ms, d, lf, pf, bs, grid = 101, bandwidth)
```

Arguments

`mf`	function for fitting the model
`f`	formula of full model
`ms`	list of formula of candidates models
`d`	data
`lf`	measure lack of fit (to minimize)
`pf`	model selection criteria, e.g., model dimension
`bs`	bootstrap samples
`grid`	grid for c
`bandwidth`	bandwidth for kernel smooth function

Value

`models`	list all model candidates in the model space
`B`	list the number of bootstrap samples that have been used
`lack_of_fit_matrix`	list a matrix of Qs for all model candidates (in columns). Each row is for each bootstrap sample
`Qd_matrix`	list a matrix of QM - QM.tilde for all model candidates. Each row is for each bootrap sample
`bandwidth`	list the value of bandwidth
`model_mat`	list a matrix of selected models at each c values in grid (in columns). Each row is for each bootstrap sample
`freq_mat`	list a matrix of coverage probabilities (frequency/smooth_frequency) of each selected models for a given c value (index)
`c`	list the adaptive choice of c value from which the parsimonious model is selected
`sel_model`	list the selected (parsimonious) model given the adaptive c value

Author(s)

Jiming Jiang Jianyang Zhao J. Sunil Rao Thuan Nguyen

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiang J., Nguyen T., Rao J.S. (2009), A Simplified Adaptive Fence Procedure. Statistics and Probability Letters, 79, 625-629
- Thuan Nguyen, Jie Peng, Jiming Jiang (2014), Fence Methods for Backcross Experiments. Statistical Computation and Simulation, 84(3), 644-662

Examples

```
## Not run:
require(fence)

#### Example 1 #####
data(iris)
full = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + (1|Species)
test_af = fence.lmer(full, iris)
plot(test_af)
test_af$sel_model

#### Example 2 #####
r =1234; set.seed(r)
p=8; n=150; rho = 0.6
id = rep(1:50,each=3)
R = diag(p)
for(i in 1:p){
  for(j in 1:p){
     R[i,j] = rho^(abs(i-j))
  }
}
R = 1*R
x=mvrnorm(n, rep(0, p), R)  # all x's are time-varying dependence #
colnames(x)=paste('x',1:p, sep='')
tbetas = c(0,0.5,1,0,0.5,1,0,0.5)  # non-zero beta 2,3,5,6,8
epsilon = rnorm(150)
y = x%*%tbetas + epsilon
colnames(y) = 'y'
data = data.frame(cbind(x,y,id))
full = y ~ x1+x2+x3+x4+x5+x6+x7+x8+(1|id)
#X = paste('x',1:p, sep='', collapse='+')
#full = as.formula(paste('y~',X,'+(1|id)', sep=""))  #same as previous one
fence_obj = fence.lmer(full,data)   # it takes 3-5 min #
plot(fence_obj)
fence_obj$sel_model

## End(Not run)
```

`adaptivefence.fh` *Adaptive Fence model selection (Small Area Estmation)*

Description

Adaptive Fence model selection (Small Area Estmation)

Usage

```
adaptivefence.fh(mf, f, ms, d, lf, pf, bs, grid = 101, bandwidth, method)
```

Arguments

`mf`	Call function, for example: default calls: function(m, b) eblupFH(formula = m, vardir = D, data = b, method = "FH")
`f`	Full Model
`ms`	find candidate model, findsubmodel.fh(full)
`d`	Dimension number
`lf`	Measures lack of fit using function(res) -resfitgoodness[1]
`pf`	Dimensions of model
`bs`	Bootstrap
`grid`	grid for c
`bandwidth`	bandwidth for kernel smooth function
`method`	Method to be used. Fay-Herriot method is the default.

Details

In Jiang et. al (2008), the adaptive c value is chosen from the highest peak in the p* vs. c plot. In Jiang et. al (2009), 95% CI is taken into account while choosing such an adaptive choice of c. In Thuan Nguyen et. al (2014), the adaptive c value is chosen from the first peak. This approach works better in the moderate sample size or weak signal situations. Empirically, the first peak becomes highest peak when sample size increases or signals become stronger

Value

`models`	list all model candidates in the model space
`B`	list the number of bootstrap samples that have been used
`lack_of_fit_matrix`	list a matrix of Qs for all model candidates (in columns). Each row is for each bootstrap sample
`Qd_matrix`	list a matrix of QM - QM.tilde for all model candidates. Each row is for each bootrap sample
`bandwidth`	list the value of bandwidth

model_mat	list a matrix of selected models at each c values in grid (in columns). Each row is for each bootstrap sample
freq_mat	list a matrix of coverage probabilities (frequency/smooth_frequency) of each selected models for a given c value (index)
c	list the adaptive choice of c value from which the parsimonious model is selected
sel_model	list the selected (parsimonious) model given the adaptive c value

Note

- The current Fence package focuses on variable selection. However, Fence methods can be used to select other parameters of interest, e.g., tunning parameter, variance-covariance structure, etc.
- The number of bootstrap samples is suggested to be increased, e.g., B=1000 when the sample size is small, or signals are weak

Author(s)

Jiming Jiang Jianyang Zhao J. Sunil Rao Thuan Nguyen

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiang J., Nguyen T., Rao J.S. (2009), A Simplified Adaptive Fence Procedure. Statistics and Probability Letters, 79, 625-629
- Thuan Nguyen, Jie Peng, Jiming Jiang (2014), Fence Methods for Backcross Experiments. Statistical Computation and Simulation, 84(3), 644-662

Examples

```
## Not run:
require(fence)
### example 1 ####
data("kidney")
data = kidney[-which.max(kidney$x),]     # Delete a suspicious data point #
data$x2 = data$x^2
data$x3 = data$x^3
data$x4 = data$x^4
data$D = data$sqrt.D.^2
plot(data$y ~ data$x)
full = y~x+x2+x3+x4
testfh = fence.sae(full, data, B=1000, fence="adaptive", method="F-H", D = D)
testfh$sel_model
testfh$c

## End(Not run)
```

`fence.lmer` *Fence model selection (Linear Mixed Model)*

Description

Fence model selection (Linear Mixed Model)

Usage

```
fence.lmer(full, data, B = 100, grid = 101, fence = c("adaptive",
  "nonadaptive"), cn = NA, REML = TRUE, bandwidth = NA,
  cpus = parallel::detectCores())
```

Arguments

`full`	formula of full model
`data`	data
`B`	number of bootstrap samples, parametric bootstrap is used
`grid`	grid for c
`fence`	a procedure of the fence method to be used. It's suggested to choose nonadaptive procedure if c is known; otherwise nonadaptive must be chosen
`cn`	cn value for nonadaptive
`REML`	Restricted Maximum Likelihood approach
`bandwidth`	bandwidth for kernel smooth function
`cpus`	Number of parallel computers

Details

In Jiang et. al (2008), the adaptive c value is chosen from the highest peak in the p* vs. c plot. In Jiang et. al (2009), 95% CI is taken into account while choosing such an adaptive choice of c. In Thuan Nguyen et. al (2014), the adaptive c value is chosen from the first peak. This approach works better in the moderate sample size or weak signal situations. Empirically, the first peak becomes highest peak when sample size increases or signals become stronger

Value

`models`	list all model candidates in the model space
`B`	list the number of bootstrap samples that have been used
`lack_of_fit_matrix`	list a matrix of Qs for all model candidates (in columns). Each row is for each bootstrap sample
`Qd_matrix`	list a matrix of QM - QM.tilde for all model candidates. Each row is for each bootrap sample
`bandwidth`	list the value of bandwidth

`model_mat` list a matrix of selected models at each c values in grid (in columns). Each row is for each bootstrap sample

`freq_mat` list a matrix of coverage probabilities (frequency/smooth_frequency) of each selected models for a given c value (index)

`c` list the adaptive choice of c value from which the parsimonious model is selected

`sel_model` list the selected (parsimonious) model given the adaptive c value

@note The current Fence package focuses on variable selection. However, Fence methods can be used to select other parameters of interest, e.g., tunning parameter, variance-covariance structure, etc.

Author(s)

Jiming Jiang Jianyang Zhao J. Sunil Rao Thuan Nguyen

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiang J., Nguyen T., Rao J.S. (2009), A Simplified Adaptive Fence Procedure. Statistics and Probability Letters, 79, 625-629
- Thuan Nguyen, Jie Peng, Jiming Jiang (2014), Fence Methods for Backcross Experiments. Statistical Computation and Simulation, 84(3), 644-662

Examples

```
require(fence)
library(snow)

#### Example 1 #####
data(iris)
full = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + (1|Species)
# Takes greater than 5 seconds to run
# test_af = fence.lmer(full, iris)
# test_af$c
# test_naf = fence.lmer(full, iris, fence = "nonadaptive", cn = 12)
# plot(test_af)
# test_af$sel_model
# test_naf$sel_model
```

`fence.NF` *Fence model selection (Nonparametric Model)*

Description

Fence model selection (Noparametric Model)

Usage

```
fence.NF(full, data, spline, ps = 1:3, qs = NA, B = 100, grid = 101,
  bandwidth = NA, lambda)
```

Arguments

full	formula of full model
data	data
spline	variable needed for spline terms
ps	order of power
qs	number of knots
B	number of bootstrap sample, parametric for lmer
grid	grid for c
bandwidth	bandwidth for kernel smooth function
lambda	A grid of lambda values

Value

models	list all model candidates with p polynomial degrees and q knots in the model space
Qd_matrix	list a matrix of QM - QM.tilde for all model candidates. Each row is for each bootrap sample
bandwidth	list the value of bandwidth
model_mat	list a matrix of selected models at each c values in grid (in columns). Each row is for each bootstrap sample
freq_mat	list a matrix of coverage probabilities (frequency/smooth_frequency) of each selected models for a given c value (index)
c	list the adaptive choice of c value from which the parsimonious model is selected
lambda	penalty (or smoothing) parameter estimate given selected p and q
sel_model	list the selected (parsimonious) model given the adaptive c value
beta.est.u	A list of coefficient estimates given a lambda value
f.x.hat	A vector of fitted values obtained from a given lambda value and beta.est.u

@note The current Fence method in Nonparametric model focuses on one spline variable. This method can be extended to a general case with more than one spline variables, and includes non-spline variables.

Author(s)

Jiming Jiang Jianyang Zhao J. Sunil Rao Bao-Qui Tran Thuan Nguyen

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiang J., Nguyen T., Rao J.S. (2009), A Simplified Adaptive Fence Procedure. Statistics and Probability Letters, 79, 625-629
- Jiang J., Nguyen T., Rao J.S. (2010), Fence Method for Nonparametric Small Area Estimation. Survey Methodology, 36, 1, 3-11

Examples

```
## Not run:
require(fence)
n = 100
set.seed(1234)
x=runif(n,0,3)
y = 1-x+x^2- 2*(x-1)^2*(x>1) + 2*(x-2)^2*(x>2) + rnorm(n,sd=.2)
lambda=exp((c(1:60)-30)/3)
data=data.frame(cbind(x,y))
test_NF = fence.NF(full=y~x, data=data, spline='x', ps=c(1:3), qs=c(2,5), B=1000, lambda=lambda)
plot(test_NF)
summary <- summary(test_NF)
model_sel <- summary[[1]]
model_sel
lambda_sel <- summary[[2]]
lambda_sel

## End(Not run)
```

---

fence.sae *Fence model selection (Small Area Estmation)*

Description

Fence model selection (Small Area Estmation)

Usage

```
fence.sae(full, data, B = 100, grid = 101, fence = c("adaptive",
  "nonadaptive"), cn = NA, method = c("F-H", "NER"), D = NA,
  REML = FALSE, bandwidth = NA, cpus = parallel::detectCores())
```

Arguments

`full`	formular of full model
`data`	data
`B`	number of bootstrap sample, parametric for lmer
`grid`	grid for c

`fence`	fence method to be used, e.g., adaptive, or nonadaptive. It's suggested to choose nonadaptive procedure if c is known; otherwise nonadaptive must be chosen
`cn`	cn for nonadaptive
`method`	Select method to use
`D`	vector containing the D sampling variances of direct estimators for each domain. The values must be sorted as the variables in formula. Only used in FH model
`REML`	Restricted Maximum Likelihood approach
`bandwidth`	bandwidth for kernel smooth function
`cpus`	Number of parallel computers

Details

In Jiang et. al (2008), the adaptive c value is chosen from the highest peak in the p* vs. c plot. In Jiang et. al (2009), 95% CI is taken into account while choosing such an adaptive choice of c. In Thuan Nguyen et. al (2014), the adaptive c value is chosen from the first peak. This approach works better in the moderate sample size or weak signal situations. Empirically, the first peak becomes highest peak when sample size increases or signals become stronger

Value

`models`	list all model candidates in the model space
`B`	list the number of bootstrap samples that have been used
`lack_of_fit_matrix`	list a matrix of Qs for all model candidates (in columns). Each row is for each bootstrap sample
`Qd_matrix`	list a matrix of QM - QM.tilde for all model candidates. Each row is for each bootrap sample
`bandwidth`	list the value of bandwidth
`model_mat`	list a matrix of selected models at each c values in grid (in columns). Each row is for each bootstrap sample
`freq_mat`	list a matrix of coverage probabilities (frequency/smooth_frequency) of each selected models for a given c value (index)
`c`	list the adaptive choice of c value from which the parsimonious model is selected
`sel_model`	list the selected (parsimonious) model given the adaptive c value

Note

- The current Fence package focuses on variable selection. However, Fence methods can be used to select other parameters of interest, e.g., tunning parameter, variance-covariance structure, etc.
- The number of bootstrap samples is suggested to be increased, e.g., B=1000 when the sample size is small, or signals are weak

Author(s)

Jiming Jiang Jianyang Zhao J. Sunil Rao Thuan Nguyen

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiang J., Nguyen T., Rao J.S. (2009), A Simplified Adaptive Fence Procedure. Statistics and Probability Letters, 79, 625-629
- Thuan Nguyen, Jie Peng, Jiming Jiang (2014), Fence Methods for Backcross Experiments. Statistical Computation and Simulation, 84(3), 644-662

Examples

```
require(fence)
library(snow)
### example 1 ####
data("kidney")
data = kidney[-which.max(kidney$x),]     # Delete a suspicious data point #
data$x2 = data$x^2
data$x3 = data$x^3
data$x4 = data$x^4
data$D = data$sqrt.D.^2
plot(data$y ~ data$x)
full = y~x+x2+x3+x4
# Takes more than 5 seconds to run
# testfh = fence.sae(full, data, B=100, fence="adaptive", method="F-H", D = D)
# testfh$sel_model
# testfh$c
```

IF.lm — *Invisible Fence model selection (Linear Model)*

Description

Invisible Fence model selection (Linear Model)

Usage

```
IF.lm(full, data, B = 100, cpus = 2, lftype = c("abscoef", "pvalue"))
```

Arguments

`full`	formula of full model
`data`	data
`B`	number of bootstrap sample, parametric for lm
`cpus`	number of parallel computers
`lftype`	subtractive measure type, e.g., absolute value of coefficients, p-value, t-value, etc.

Details

This method (Jiang et. al, 2011) is motivated by computational expensive in complex and high dimensional problem. The idea of the method–there is the best model in each dimension (in model space). The boostrapping determines the coverage probability of the selected model in each dimensions. The parsimonious model is the selected model with the highest coverage probabily (except the one for the full model, always probability of 1.)

Value

`full`	list the full model
`B`	list the number of bootstrap samples that have been used
`freq`	list the coverage probabilities of the selected model for each dimension
`size`	list the number of variables in the parsimonious model
`term`	list variables included in the full model
`model`	list the variables selected in-the-order in the parsimonious model

@note The current Invisible Fence focuses on variable selection. The current routine is applicable to the case in which the subtractive measure is the absolute value of the coefficients, p-value, t-value. However, the method can be extended to other subtractive measures. See Jiang et. al (2011) for more details.

Author(s)

Jiming Jiang Jianyang Zhao J. Sunil Rao Thuan Nguyen

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiming Jiang, Thuan Nguyen and J. Sunil Rao (2011), Invisible fence methods and the identification of differentially expressed gene sets. Statistics and Its Interface, Volume 4, 403-415.

Examples

```
library(fence)
library(MASS)
library(snow)
r =1234; set.seed(r)
p=10; n=300; rho = 0.6
R = diag(p)
for(i in 1:p){
  for(j in 1:p){
     R[i,j] = rho^(abs(i-j))
  }
}
R = 1*R
x=mvrnorm(n, rep(0, p), R)
colnames(x)=paste('x',1:p, sep='')
X = cbind(rep(1,n),x)
```

```
tbetas = c(1,1,1,0,1,1,0,1,0,0,0)  # non-zero beta 1,2,4,5,7
epsilon = rnorm(n)
y = as.matrix(X)%*%tbetas + epsilon
colnames(y) = 'y'
data = data.frame(cbind(X,y))
full = y ~ x1+x2+x3+x4+x5+x6+x7+x8+x9+x10
# Takes greater than 5 seconds (~`17 seconds) to run
# obj1 = IF.lm(full = full, data = data, B = 100, lftype = "abscoef")
# sort((names(obj1$model$coef)[-1]))
# obj2 = IF.lm(full = full, data = data, B = 100, lftype = "pvalue")
# sort(setdiff(names(data[c(-1,-12)]), names(obj2$model$coef)))
```

IF.lmer *Invisible Fence model selection (Linear Mixed Model)*

Description

Invisible Fence model selection (Linear Mixed Model)

Usage

```
IF.lmer(full, data, B = 100, REML = TRUE, method = c("marginal",
  "conditional"), cpus = parallel::detectCores(), lftype = c("abscoef",
  "tvalue"))
```

Arguments

`full`	formula of full model
`data`	data
`B`	number of bootstrap sample, parametric for lmer
`REML`	Restricted maximum likelihood estimation
`method`	choose either marginal (e.g., GEE) or conditional model
`cpus`	Number of parallel computers
`lftype`	subtractive measure type, e.g., absolute value of coefficients, p-value, t-value, etc.

Details

This method (Jiang et. al, 2011) is motivated by computational expensive in complex and high dimensional problem. The idea of the method–there is the best model in each dimension (in model space). The boostrapping determines the coverage probability of the selected model in each dimensions. The parsimonious model is the selected model with the highest coverage probabily (except the one for the full model, always probability of 1.)

Value

full	list the full model
B	list the number of bootstrap samples that have been used
freq	list the coverage probabilities of the selected model for each dimension
size	list the number of variables in the parsimonious model
term	list variables included in the full model
model	list the variables selected in-the-order in the parsimonious model

@note The current Invisible Fence focuses on variable selection. The current routine is applicable to the case in which the subtractive measure is the absolute value of the coefficients, p-value, t-value. However, the method can be extended to other subtractive measures. See Jiang et. al (2011) for more details.

Author(s)

Jiming Jiang Jianyang Zhao J. Sunil Rao Thuan Nguyen

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiming Jiang, Thuan Nguyen and J. Sunil Rao (2011), Invisible fence methods and the identification of differentially expressed gene sets. Statistics and Its Interface, Volume 4, 403-415.

Examples

```
require(fence)
library(snow)
library(MASS)
data("X.lmer")
data = data.frame(X.lmer)
# non-zero beta I.col.2, I.col.3a, I.col.3b, V5, V7, V8, V9
beta = matrix(c(0, 1, 1, 1, 1, 0, 0.1, 0.05, 0.25, 0), ncol = 1)
set.seed(1234)
alpha = rep(rnorm(100), each = 3)
mu = alpha + as.matrix(data[,-1]) %*% beta
data$id = as.factor(data$id)
data$y = mu + rnorm(300)
raw = "y ~ (1|id)+I.col.2+I.col.3a+I.col.3b"
for (i in 5:10) {
    raw = paste0(raw, "+V", i)
}
full = as.formula(raw)
# The following output takes more than 5 seconds (~70 seconds) to run.

# obj1.lmer = IF.lmer(full = full, data = data, B = 100, method="conditional",lftype = "abscoef")
# sort(obj1.lmer$model)

# obj2.lmer = IF.lmer(full = full, data = data, B = 100, method="conditional",lftype = "tvalue")
```

```
# sort(obj2.lmer$model)

# Similarly, the following scenarios can be run

# obj2.lmer = IF.lmer(full = full, data = data, B = 100, method="conditional",lftype = "tvalue")
# sort(obj2.lmer$model)
# obj1.lm = IF.lmer(full = full, data = data, B = 100, method="marginal", lftype = "abscoef")
# sort(names(obj1.lm$model$coefficients[-1]))
# obj2.lm = IF.lmer(full = full, data = data, B = 100, method="marginal", lftype = "tvalue")
# sort(names(obj2.lm$model$coefficients[-1]))
```

---

invisiblefence *Invisible Fence model selection*

Description

Invisible Fence model selection

Usage

```
invisiblefence(mf, f, d, lf, bs)
```

Arguments

mf	Call function, for example: default calls: function(m, b) eblupFH(formula = m, vardir = D, data = b, method = "FH")
f	Full model
d	Dimension number
lf	Measures lack of fit using function(res) -resfitgoodness[1]
bs	Bootstrap

Details

This method (Jiang et. al, 2011) is motivated by computational expensive in complex and high dimensional problem. The idea of the method–there is the best model in each dimension (in model space). The boostrapping determines the coverage probability of the selected model in each dimensions. The parsimonious model is the selected model with the highest coverage probabily (except the one for the full model, always probability of 1.)

Value

full	list the full model
B	list the number of bootstrap samples that have been used
freq	list the coverage probabilities of the selected model for each dimension
size	list the number of variables in the parsimonious model

`term` list variables included in the full model

`model` list the variables selected in-the-order in the parsimonious model

@note The current Invisible Fence focuses on variable selection. The current routine is applicable to the case in which the subtractive measure is the absolute value of the coefficients, p-value, t-value. However, the method can be extended to other subtractive measures. See Jiang et. al (2011) for more details.

Author(s)

Jiming Jiang Jianyang Zhao J. Sunil Rao Thuan Nguyen

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiming Jiang, Thuan Nguyen and J. Sunil Rao (2011), Invisible fence methods and the identification of differentially expressed gene sets. Statistics and Its Interface, Volume 4, 403-415.

Examples

```
## Not run:
data("X.lmer")
data = data.frame(X.lmer)
beta = matrix(c(0, 1, 1, 1, 1, 0, 0.1, 0.05, 0.25, 0), ncol = 1)
set.seed(1234)
alpha = rep(rnorm(100), each = 3)
mu = alpha + as.matrix(data[,-1]) %*% beta
data$id = as.factor(data$id)
data$y = mu + rnorm(300)
raw = "y ~ (1|id)+I.col.2+I.col.3a+I.col.3b"
for (i in 5:10) {
    raw = paste0(raw, "+V", i)
}
full = as.formula(raw)
obj1.lmer = IF.lmer(full = full, data = data, B = 100, method="conditional",lftype = "abscoef")
obj1.lmer$model$coefficients
obj2.lmer = IF.lmer(full = full, data = data, B = 100, method="conditional",lftype = "tvalue")
obj2.lmer$model$coefficients
obj1.lm = IF.lmer(full = full, data = data, B = 100, method="marginal", lftype = "abscoef")
obj1.lm$model$coefficients
obj2.lm = IF.lmer(full = full, data = data, B = 100, method="marginal", lftype = "tvalue")
obj2.lm$model$coefficients

## End(Not run)
```

kidney *kidney*

Description

Data used for kidney example

Usage

```
kidney
```

Format

A data frame with 4 variables

nonadaptivefence *Nonadaptive Fence model selection*

Description

Nonadaptive Fence model selection

Usage

```
nonadaptivefence(mf, f, ms, d, lf, pf, cn)
```

Arguments

mf	function for fitting the model
f	formula of full model
ms	list of formula of candidates models
d	data
lf	measure lack of fit (to minimize)
pf	model selection criteria, e.g., model dimension
cn	given a specific c value

Value

models	list all model candidates in the model space
lack_of_fit	list a vector of Qs for all model candidates
formula	list the model of the selected parsimonious model
sel_model	list the selected (parsimonious) model given the adaptive c value

Author(s)

Jiming Jiang Jianyang Zhao J. Sunil Rao Thuan Nguyen

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiang J., Nguyen T., Rao J.S. (2009), A Simplified Adaptive Fence Procedure. Statistics and Probability Letters, 79, 625-629
- Thuan Nguyen, Jie Peng, Jiming Jiang (2014), Fence Methods for Backcross Experiments. Statistical Computation and Simulation, 84(3), 644-662

Examples

```
## Not run:
require(fence)

#### Example 1 #####
data(iris)
full = Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + (1|Species)
test_naf = fence.lmer(full, iris, fence = "nonadaptive", cn = 12)
test_naf$sel_model

## End(Not run)
```

`plot.AF` *Plot Adaptive Fence model selection*

Description

Plot Adaptive Fence model selection

Usage

```
## S3 method for class 'AF'
plot(x = res, ...)
```

Arguments

`x`	Object to be plotted
`...`	Additional arguments. CNot currently used.

plot.NF *Plot Nonparametric Fence model selection*

Description

Plot Nonparametric Fence model selection

Usage

```
## S3 method for class 'NF'
plot(x = res, ...)
```

Arguments

x	Object to be plotted
...	Additional arguments. CNot currently used.

RF *Adaptive Fence model selection (Restricted Fence)*

Description

Adaptive Fence model selection (Restricted Fence)

Usage

```
RF(full, data, groups, B = 100, grid = 101, bandwidth = NA,
  plot = FALSE, method = c("marginal", "conditional"), id = "id",
  cpus = parallel::detectCores())
```

Arguments

full	formula of full model
data	data
groups	A list of formulas of (full) model in each bins (groups) of variables
B	number of bootstrap sample, parametric for lmer
grid	grid for c
bandwidth	bandwidth for kernel smooth function
plot	Plot object
method	either marginal (GEE) or conditional approach is selected
id	Subject or cluster id variable
cpus	Number of parallel computers

Details

In Jiang et. al (2008), the adaptive c value is chosen from the highest peak in the p* vs. c plot. In Jiang et. al (2009), 95% CI is taken into account while choosing such an adaptive choice of c. In Thuan Nguyen et. al (2014), the adaptive c value is chosen from the first peak. This approach works better in the moderate sample size or weak signal situations. Empirically, the first peak becomes highest peak when sample size increases or signals become stronger

Value

`models`	list all model candidates in the model space
`B`	list the number of bootstrap samples that have been used
`lack_of_fit_matrix`	list a matrix of Qs for all model candidates (in columns). Each row is for each bootstrap sample
`Qd_matrix`	list a matrix of QM - QM.tilde for all model candidates. Each row is for each bootrap sample
`bandwidth`	list the value of bandwidth
`model_mat`	list a matrix of selected models at each c values in grid (in columns). Each row is for each bootstrap sample
`freq_mat`	list a matrix of coverage probabilities (frequency/smooth_frequency) of each selected models for a given c value (index)
`c`	list the adaptive choice of c value from which the parsimonious model is selected
`sel_model`	list the selected (parsimonious) model given the adaptive c value

Note

bandwidth = (cs[2] - cs[1]) * 3. So it's chosen as 3 times grid between two c values.

References

- Jiang J., Rao J.S., Gu Z., Nguyen T. (2008), Fence Methods for Mixed Model Selection. The Annals of Statistics, 36(4): 1669-1692
- Jiang J., Nguyen T., Rao J.S. (2009), A Simplified Adaptive Fence Procedure. Statistics and Probability Letters, 79, 625-629
- Thuan Nguyen, Jiming Jiang (2012), Restricted fence method for covariate selection in longitudinal data analysis. Biostatistics, 13(2), 303-314
- Thuan Nguyen, Jie Peng, Jiming Jiang (2014), Fence Methods for Backcross Experiments. Statistical Computation and Simulation, 84(3), 644-662

Examples

```
## Not run:
r =1234; set.seed(r)
n = 100; p=15; rho = 0.6
beta = c(1,1,1,0,1,1,0,1,0,0,1,0,0,0,0)  # non-zero beta 1,2,3,V6,V7,V9,V12
id = rep(1:n,each=3)
```

```
V.1 = rep(1,n*3)
I.1 = rep(c(1,-1),each=150)
I.2a = rep(c(0,1,-1),n)
I.2b = rep(c(0,-1,1),n)
x = matrix(rnorm(n*3*11), nrow=n*3, ncol=11)
x = cbind(id,V.1,I.1,I.2a,I.2b,x)
R = diag(3)
for(i in 1:3){
 for(j in 1:3){
   R[i,j] = rho^(abs(i-j))
 }
}
e=as.vector(t(mvrnorm(n, rep(0, 3), R)))
y = as.vector(x[,-1]%*%beta) + e
data = data.frame(x,y)
raw = "y ~ V.1 + I.1 + I.2a +I.2b"
for (i in 6:16) { raw = paste0(raw, "+V", i)}; full = as.formula(raw)
bin1="y ~ V.1 + I.1 + I.2a +I.2b"
for (i in 6:8) { bin1 = paste0(bin1, "+V", i)}; bin1 = as.formula(bin1)
bin2="y ~ V9"
for (i in 10:16){ bin2 = paste0(bin2, "+V", i)}; bin2 = as.formula(bin2)
# May take longer than 30 min since there are two stages in this RF procedure
obj1.RF = RF(full = full, data = data, groups = list(bin1,bin2), method="conditional")
obj1.RF$sel_model
obj2.RF = RF(full = full, data = data, groups = list(bin1,bin2), B=100, method="marginal")
obj2.RF$sel_model

## End(Not run)
```

summary.AF — *Summary Adaptive Fence model selection*

Description

Summary Adaptive Fence model selection

Usage

```
## S3 method for class 'AF'
summary(object = res, ...)
```

Arguments

`object`	Object to be summarized
`...`	addition arguments. Not currently used

summary.NF — *Summary Nonparametric Fence model selection*

Description

Summary Nonparametric Fence model selection

Usage

```
## S3 method for class 'NF'
summary(object = res, ...)
```

Arguments

`object`	Object to be summarized
`...`	addition arguments. Not currently used

X.lmer — *X.lmer*

Description

Data used in the example for X.lmer

Usage

```
data(X.lmer)
```

Format

A data frame with 10 variables:

Index