Package 'CombMSC'

January 4, 2019

Type Package

Title Combined Model Selection Criteria

Version 1.4.2.1

Date 2008-02-24

Author Andrew K. Smith

Maintainer Andrew K. Smith <andrewsmith81@gmail.com>

Description Functions for computing optimal convex combinations of
model selection criteria based on ranks, along with utility
functions for constructing model lists, MSCs, and priors on
model lists.

License GPL-2

Depends R(>= 2.4.0)

Suggests lattice, rgl

Repository CRAN

Date/Publication 2019-01-04 18:23:10 UTC

NeedsCompilation no

R topics documented:

CombMSC-package   *Combined Model Selection Criteria*

Description

Functions to compute optimal convex combinations of model selection criteria, as well as utility functions to construct model lists and priors, and several common model selection criteria.

Details

Package:	CombMSC
Type:	Package
Version:	1.4
Date:	2007-11-27
License:	GPL v3

The most important function is TMC. The other user-level functions are mainly subsidiary, and are used, for example, to construct objects to be passed as arguments to TMC.

Author(s)

Andrew K. Smith

Maintainer: Andrew K. Smith <andrewsmith81@gmail.com>

Adj.Rsq   *Adjusted R-Squared*

Description

Convenience function to calculate R-Squared from an object inheriting from class lm.

AIC.fmo	*AIC*

Description

New method for fmo objects used internally by TMC.

Author(s)

Andrew K. Smith

BIC	*Akaike Information Criterion*

Description

Convenience function (wrapper for AIC).

compare	*Compare 2 msc objects*

Description

Gives a graphical comparison of 2 msc objects by plotting the differences of their response function values.

Details

This is intended to help visualize the effect of tweaking a parameter or two in the call to TMC. The resulting plot may not be meaningful for radically different objects. In particular, the function will not work if the msc.List argument or the stepSize argument differ between the two objects, since then the resulting graphs are no longer comparable.

Author(s)

Andrew K. Smith

See Also

TMC

Examples

```
## Not run:
example(TMC)
result3 <- update(result, par.Sigma = 2)
compare(result, result3)
## End(Not run)
```

Cp	*Mallows Cp*

Description

Calculates the Mallows Cp criterion of a lm object

Usage

```
Cp(object, a = 1, b = 0, ...)
```

Arguments

`object`	object passed from TMC containing the fitted lm object
`a`	constant multiplier for the Cp term itself
`b`	constant multiplier for the closeness of Cp to p.
`...`	other arguments

Details

In the model selection criteria for TMC, we require a single quantity to be minimized or maximized, but the common interpretation of Cp is that we want Cp to be "small and close to p." Therefore, we have introduced arguments a and b which allow the user to customize, if desired, the relative weights of the two criteria. By default, the function simply calculates the raw Cp value, and therefore the closeness of Cp to p is of no consequence. However, if one wishes the final criterion value (which will eventually be ranked in TMC) to take closeness to p into account as well, one can easily modify the MSC function in the `msc.List` argument to adjust the constant multipliers. The final value returned will be $a * Cp + b * |Cp - p|$.

Value

$a * Cp + b * |Cp - p|$

Author(s)

Andrew K. Smith

`fit.Models` *Generic model fitting*

Description

A default for the `model.List` argument of [TMC](TMC) which will correctly fit time series, regression, and GLM models.

Usage

```
fit.Models(fmla, ...)
```

Arguments

`fmla`	A generic model specification object.
`...`	Other arguments, which will depend on the type of model involved.

Details

This function is intended to be used internally in [TMC](TMC) as argument `fit.Model`. It will be applied to each element of the argument `model.List` during each iteration. Methods are included for lm, glm, and time-series models, and new methods can easily be written for different types of models.

Author(s)

Andrew K. Smith

`gen.Data` *Simulate Data Sets*

Description

Simulates data sets from a given regression or time series model. For internal use in TMC.

---

gen.Params   *Simulate parameters*

Description

Simulates parameters for either a regression formula or a time series model specified by a numeric vector.

Details

This function simply simulates whatever number of parameters is required of the model. In the case of regression, this means one parameter for each variable in the RHS of the formula, while for a time series, it means one parameter for each component in the AR, MA, SAR, and SMA parts of the model.

Value

A numeric vector of parameters, of the appropriate length as determined by the argument fmla.

Author(s)

Andrew K. Smith

---

holdout.SS   *Holdout Sample Prediction Errors*

Description

Calculates either the sum of squares, mean absolute value, or median absolute value of the set of holdout sample prediction errors, for regression or time series models. This is only for use as a criterion to be included in an msc.List argument to TMC, and cannot be used by itself.

Details

This is intended to provide a more accurate estimate of out-of-sample prediction error. The choice of whether to use the sum of squares, median absolute value, or mean absolute value will depend on the particular application, and is analogous to the choice of doing least-squares versus least-median-residual regression. (Note, however, that there is no substantial computational difference between these criteria.)

Author(s)

Andrew K. Smith

---

make.Model.List.Reg   *Make Model Lists*

Description

Generates a list of regression formulae or numeric vectors for time series. Each element of the list is either a formula for which a regression model, or a tsm object for a time series model.

Usage

```
make.Model.List.Reg(fram, max.Size = min(8, dim(fram)[2]), no.Intercepts = FALSE, GLM = FALSE)
```

Arguments

fram	data.frame of covariates.
max.Size	The maximum number of terms to be included in the model.
no.Intercepts	Logical indicating whether or not to include model formulae with a " + 0" at the end. By default, these will not be included, so all models will implicitly include an intercept term unless otherwise specified.
GLM	Logical indicating whether the formulas are to be interpreted as specifying an lm (FALSE) or a glm (TRUE) type of model.

Details

The function will generate all possible model formulae with LHS equal to a dummy variable y, and the RHS given by all possible subsets of the column names of var.Frame, subject to the restriction that all subsets are no larger than max.Size. For time series, max.Size is a vector of length 6, specifying the maximum of each of the 6 components (AR, I, MA, SAR, SI, SMA) to generate.

Value

A list of either model formulae or numeric vectors.

Author(s)

Andrew K. Smith

See Also

TMC

---

num.Terms   *Number of terms (parameters) in a model*

Description

For a regression formula or a numeric vector specifying a time series model, calculates the number of parameters needed to specify the model. Mainly used to construct weight vectors based on the complexity of the models.

Author(s)

Andrew K. Smith

---

plot.msc   *Display an msc object*

Description

Provides several options for displaying the output of TMC.

Usage

```
## S3 method for class 'msc'
plot(x, sumfun.Index = "def", fancy = FALSE, grayscale = FALSE, draw.Grid = TRUE, ...)
```

Arguments

x	An msc object, outputted by TMC.
sumfun.Index	This is the index number of the summary function you wish to plot. Available summary functions are listed in the summary.Functions component of the msc object.
fancy	If TRUE, will generate a 3-D plot (based on package rgl) of the results. Only works if the objects msc.List component is of length exactly 3.
grayscale	If true, will make the plots grayscale instead of color. Useful for printing. Only effective if fancy is FALSE.
draw.Grid	Draws a grid in the background of each Lattice plot. Only effective if fancy is FALSE.
...	Other graphical parameters.

Details

If, as we recommend, msc.List is of length exactly 3, the plot will display as a function of 2 independent variables. This is because, although there are technically 3 independent variables (the weights of the 3 distinct MSCs,) they are related by the equation $weight_1 + weight_2 + weight_3 = 1$. Hence the function is defined only on the simplex joining the 3 vectors (0,0,1), (0,1,0), (1,0,0), and so may be viewed as a 2-dimensional set. Thus, in the plot, each corner represents a pure msc, and the interior of the triangle represents the various convex combinations of them in an intuitive way.

For an msc.List of length 4, the plot produces a trellis of plots by conditioning on the weight of the 4th msc. For msc.List length larger than 4, there is no default plotting method available — instead try using `summary.msc`.

Author(s)

Andrew K. Smith

PRESS	*PRESS*

Description

Convenience function to calculate the PRESS criterion value of a fitted lm object.

Author(s)

Andrew K. Smith

---

print.msc	*Default print method for an msc object*

Description

Displays the call component of the msc object and a list of the names of the components of the object.

Author(s)

Andrew K. Smith

---

print.summary.msc   *Default print method for a summary of an msc object*

Description

Prints the most relevant information from an msc object. Gives the values of all functions in the summary.Functions component of the object evaluated at all of the corners (convex combinations with all weights equal to 0 or 1) , and then gives the top 5 convex combinations for each summary function, along with the values of all summary functions corresponding to those combinations.

Author(s)

Andrew K. Smith

---

print.tsm   *Default print method for a tsm object*

Description

Displays all the components of the time series model defined by the object in a tabular format.

Author(s)

Andrew K. Smith

---

sarima.Sim   *Simulate from a specified SARIMA model*

Description

Generates data from a SARIMA model specified by a numeric vector of length 6.

Usage

```
sarima.Sim(n = 20, period = 12, model, seasonal, rand.Gen.Fun = rnorm, rand.Gen.Seas = rnorm)
```

Arguments

n	Number of data points to generate, given in terms of the number of periods.
period	Length of the period (e.g., for yearly data, this should be 12.)
model	A list containing the AR, I, and MA components.
seasonal	A list containing the seasonal AR, I, and MA components.
rand.Gen.Fun	The function which generates the innovations for the nonseasonal components
rand.Gen.Seas	The function which generates the innovations for the seasonal components

Note

Based on code by Vincent Zoonekynd, licensed under the Creative Commons License.

Author(s)

Andrew K. Smith

---

sgnf	*Significance of an msc object*

Description

A convenience function which calculates, for each summary function of the msc object, the difference between the best pure MSC and the best combined MSC (i.e., how much better we can make each summary function by considering combined MSC instead of only the pure MSCs.) The results may be useful if one is interested in testing the hypothesis that pure MSCs are as good as any convex combination of MSCs.

Author(s)

Andrew K. Smith

---

splitTrainTest	*Split a time series into training and testing sets*

Description

Given a time series, this function splits it into two, depending on the length of numTrain. The first numTrain observations go to the training set, the remainder into the testing set, while retaining the time series attributes of both objects and correctly adjusting the start times and frequencies of both sets.

Usage

```
splitTrainTest(dat, numTrain = length(dat) - 10)
```

Arguments

dat	Time series
numTrain	The number of observations to keep in the training set

Author(s)

Andrew K. Smith

---

subsets   *Generate all the combinations of k out of n elements*

Description

Generates all different subsets of size r chosen from n different elements.

Usage

```
subsets(n, r, v = 1:n)
```

Arguments

n	Numeric. Number of elements to choose from.
r	Numeric. Size of the desired subsets.
v	Vector. Numeric or character vector of size n with the labels of the n elements to choose from.

Value

A matrix of dimension $(N \times r)$, where N is the total number of different combinations of r elements chosen from n possible.

Note

This particular version of the function was taken from a message from Bill Venables to 'r-help' list on Sun, 17 Dec 2000.

Documentation based on subsets.Rd file in package BHH2 by Ernesto Barrios.

Author(s)

Bill Venables <Bill.Venables@cmis.csiro.au>

References

Venables, Bill. "Programmers Note", R-News, Vol 1/1, Jan. 2001. http://cran.r-project.org/doc/Rnews

See Also

combinations of the **gtools** package.

Examples

```
subsets(5,3)
subsets(5,3,letters)
subsets(5,3,c(10,20,30,50,80))
```

TMC | *Compare model selection criteria*

Description

Computes convex combinations of model selection criteria. The function is very customizable, allowing the user to specify what type of model is to be tested, which criteria are to be used, and many other options described below.

Usage

```
TMC(num.Iter = 50, data.Size = 100, make.Data = gen.Data, make.Params = gen.Params, model.List, weight
```

Arguments

num.Iter
: The number of iterations. This will be the total number of times that the entire loop described in the Details section will be executed.

data.Size
: For time-series (and possibly other extended types), the size of each simulated data set.

make.Data
: The name (not quoted) of a function used to simulate data. Must take the results of make.Params as only argument. A sensible default is gen.Data for time series and regression.

make.Params
: The name (not quoted) of a function used to simulate parameters. Must take a single model as its only argument. A sensible default is gen.Params for time series and regression.

model.List
: A list of candidate models. The true model will be chosen from this list in each iteration, and the MSC values of every model in this list will then be calculated, from which the rank of the true model is computed. Utility functions for constructing such model lists are make.Model.List.Reg and make.Model.List.TS.

weight.Vector
: A numeric vector, the same length as model.List, of the weights (probabilities) of each model. Used to choose the true model at each iteration. Need not be scaled, but must be nonnegative. To construct a vector of weights for individual models based on a prior distribution on the number of terms (or complexity) of the underlying model, use weightsGivenSize. Another possible utility function, which weights models of *only* a specified size, is weight.Only.N.

msc.List
: A list of model selection criterion functions. The length must be more than 1, but should not be much larger than 3 to avoid computational overflow. The recommended number of MSCs is 3. Each function must take a fitted model object (produced by fit.Model) as its only argument. Commonly used functions include AIC, BIC, and for time series models, holdout.Mean for mean absolute deviation on a holdout sample, and holdout.Med for the median absolute deviation. This list, however, is by no means exhaustive and new MSC functions can easily be written – see details below.

fit.Model
: The function used to fit the models defined by model.List. Whenever possible, we recommend that this be a built-in R function, e.g., lm or arima.

stepSize	The mesh of the grid of convex combinations. Bear in mind the number of convex combinations will be roughly proportional to $\left(\frac{1}{stepSize}\right)^{length(msc.List)}$, so don't make stepSize too small, especially if msc.List is longer than 3!
sumstats	The summary functions of the distributions of ranks. Used for graphical displays of the final msc object. Note that the average and also all the summary functions generated by thresholds (see below) are automatically included in the final object, so there is no need to put them in this list.
huge	Required to be set to TRUE if the matrix of convex combinations will be larger than roughly 500000. To avoid unexpectedly long calculations.
var.Frame	For models with covariates, this should be the data.frame containing them. For other models, it is ignored.
par.Sigma	This argument may be passed to make.Params, and is the standard deviation used in gen.Params.lmFormula, for example.
data.Sigma	An optional argument to be passed to make.Data.
barebones	For large computations, we recommend this be set to TRUE. It will throw away the individual ranks at each iteration, updating only the summary functions, in order to reduce space requirements. If barebones is TRUE, summary.Functions is restricted to pre-defined functions which can be updated dynamically, such as mean, and cannot include functions which require the whole sample, such as median.
allow.Negs	If TRUE, the matrix of convex combinations will be expanded to include linear combinations with negative weights. Greatly increases computation, and is rarely helpful.
thresholds	Must be a numeric vector. Included as a simple way to generate summary functions — for each element k of this vector, the summary function P(Rank > k) will be computed and included in the final object. Note that if barebones is set to TRUE, the elements of thresholds are the ONLY summary functions the user can specify (this must be enforced to ensure that the barebones routine does not need to keep track of all the ranks from individual iterations, but instead can retain only the updated summary function values.
test.Size	The size of the subset of each sample to be used as a holdout sample. Ordinarily, this is set to 0, but for certain MSCs, namely those whose names begin with "holdout", it needs to be set to a nonzero number to be useful. A common rule of thumb is to set the size to be roughly ten percent of the total sample size. Note, however, that whenever this argument is nonzero, the function will slow down considerably, since it is then forced to fit all models twice (once with the full sample, once with only the "training" sample, without the holdout sample included.)
scale.Frame	Logical indicating whether var.Frame should be scaled first. If true, each column will be centered by its mean and divided by its standard deviation.
use.Ranks	Logical. If TRUE, then in each iteration, the msc values for each criterion will be scaled by taking ranks. If FALSE, then they will be scaled by standardizing instead.
...	Other arguments to be passed to other functions.

Details

The basic algorithm is as follows:

1. Choose a true model from model.List, by simulating a random entry using the weights (if present) given in weight.Vector.

2. Simulate parameters for that model by calling make.Params with argument true.Model as given above.

3. Simulated data from true.Model with params given above by calling function make.Data.

4. Fit all models in model.List to the simulated data set.

5. Calculated the model selection criteria in msc.List to each fitted model, and take the ranks of these values (within each individual MSC.)

6. For each convex combination in the grid implicitly defined by stepSize, calculated the convex combination of ranks of the different MSCs for each model in model.List

7. Among these values, calculate the rank of true.Model.

After these steps have been iterated num.Iter times, the summary functions specified in sumstats, as well as the average and threshold functions defined by thresholds, are computed for each convex combination.

New model selection functions, or additional methods for existing ones, can easily be written. The object initially passed to each such function will be of class "fmo", a class used internally in TMC. An fmo object will contain at least the components

`full` the fitted model object resulting from applying `fit.Model` to the full data set generated by `gen.Data`

`train` the fitted model object resulting from applying `fit.Model` to only the training part of the data set (that is, the data set less any observations held out for msc functions involving a holdout sample.) If `test.Size = 0`, this is NULL.

`test.Frame` the matrix of covariates associated with the holdout sample, if any. If `test.Size = 0`, this is NULL.

`test.Vector` the actual vector of observations held out as a test sample. If `test.Size = 0`, this is NULL.

`S2` an unbiased estimate of residual variance in regression models, included only for convenience in calculating `Cp` to avoid recalculating for every criterion.

Thus, to write a new model selection criterion function, one should create a generic function with a method for class "fmo", and further methods for whatever classes of model objects for which one can actually compute the criterion directly. The method for class "fmo" is typically very simple, and usually involves calling another method of the same function on some part of the fmo object itself, typically the `full` component for ordinary model selection criteria or the `train` component for criteria involving a holdout sample. For example, see `PRESS`.

`gen.Data`, `gen.Params`, and `fit.Models` are intended to be sensible defaults, but they certainly need not be the only functions one uses for simulating parameters, data, and fitting models. New methods can easily be written for all three such functions. It is recommended that, to do this, one creates a new class, create a list of model specifications (e.g., model formulae or order specifications) of this new class, and then write methods for `gen.Params`, etc. for this new class.

Value

An object of class msc, or an object of class barebones, which inherits from msc, if barebones is TRUE. Contains the following components:

call   The matched call

Sum.Stats   A data.frame, with each row representing a convex combination of MSCs. The first 3 columns give the weights corresponding to the combination, and the remaining columns give the values of all summary statistics corresponding to the combination.

var.Frame   For models containing covariates, a data.frame containing them.

error.Iterations
                Iteration numbers in which the attempt to fit the true model to the simulated data set resulted in an error, thus making it impossible to compute a rank.

num.Errors   The length of error.Iterations.

time.Taken   The total length of time to complete the call.

simulated.Models
                The formula corresponding to the true model chosen in each iteration.

simulation.Attempts
                The number of attempts needed, during each iteration, to simulate data successfully. Mainly useful for diagnostic purposes when simulation of time series results in non-stationary data.

In addition, if barebones is FALSE, the following components will also be included:

ranks.Mat   A matrix containing the ranks corresponding to each combination for every iteration. One can use this, for example, to calculate the values of new summary functions.

simulated.Data A list of data vectors simulated at each iteration.

simulated.Parameters
                A list of vectors containing the simulated parameters from each iteration.

simulated.Models
                A list of the actual models chosen (from the prior given by weight.Vector). Each will be an element of model.List.

Plus several other components which are taken directly from the call, for convenience in later processing.

Author(s)

Andrew K. Smith

References

A more complete description of the algorithm used, as well as a discussion of its properties and illustrations of its potential utility, can be found at http://www.isye.gatech.edu/~asmith/combmsc.pdf.

Examples

```
# Regression example
vars <- rnorm(60)
dim(vars)<- c(20,3)
vars <- data.frame(vars)

result <- TMC(num.Iter = 3, model.List = make.Model.List.Reg(vars), msc.List = list(BIC, AIC, PRESS), var.Frame =

# Time Series Example
modList <- make.Model.List.TS(c(1,0,1,0,0,1))

result2 <- TMC(num.Iter = 3,model.List = modList, msc.List = list(BIC, holdout.Mean,
AIC), test.Size = 10)
```

---

weight.Only.N   *Weight Only Models of Size N*

Description

Given a model list, this generates a weight vector which assigns positive (equal) weight to all models of size n (and ONLY models of exactly that size). Uses num.Terms to determine the size of each model.

Usage

```
weight.Only.N(model.List, n)
```

Arguments

model.List   A list of models.

n   The desired size to weight.

Author(s)

Andrew K. Smith

weightsGivenSize	*Generate weight vector based on prior on model size*

Description

Given a model list, and a numeric vector of prior weights on model size, generates a weight vector for the entire model list based on the prior.

Usage

```
weightsGivenSize(vec, modlist)
```

Arguments

vec	Vector of weights on model size. The first entry is the weight on the model with 0 terms, the second entry is the weight for models with 1 term, etc.
modlist	A list of models

Author(s)

Andrew K. Smith

Index