Package 'AdequacyModel'

May 20, 2016

Type Package

Title Adequacy of Probabilistic Models and General Purpose Optimization

Author Pedro Rafael Diniz Marinho [aut, cre], Marcelo Bourguignon [aut, ctb], Cicero Rafael Barros Dias [aut, ctb]

Version 2.0.0

Maintainer Pedro Rafael Diniz Marinho <pedro.rafael.marinho@gmail.com>

Description The main application concerns to a new robust optimization package with two major contributions. The first contribution refers to the assessment of the adequacy of probabilistic models through a combination of several statistics, which measure the relative quality of statistical models for a given data set. The second one provides a general purpose optimization method based on meta-heuristics functions for maximizing or minimizing an arbitrary objective function.

URL http://www.r-project.org

License GPL (>= 2)

NeedsCompilation no

Imports methods, stats, graphics

Repository CRAN

Date/Publication 2016-05-20 01:15:58

R topics documented:

AdequacyModel-package *Adequacy of probabilistic models and and general purpose optimization*

Description

The main application concerns to a new robust optimization package with two major contributions. The first contribution refers to the assessment of the adequacy of probabilistic models through a combination of several statistics, which measure the relative quality of statistical models for a given data set. The second one provides a general purpose optimization method based on meta-heuristics functions for maximizing or minimizing an arbitrary objective function.

Details

Package:	AdequacyModel
Type:	Package
Version:	1.0
License:	GPL-2
Depends:	R (>= 2.10.0)

References

Aarset, M. V. (1987). How to identify bathtub hazard rate. IEEE Transactions Reliability, 36, 106-108.

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) The New S Language. Wadsworth & Brooks/Cole.

Beni G, Wang J (1993). Swarm intelligence in cellular robotic systems. pp. 703-712.

Chen, G., Balakrishnan, N. (1995). A general purpose approximate goodness-of-fit test. Journal of Quality Technology 27, 154-16.

Eberhart RC, Kennedy J (1995). A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science, volume 1, pp. 39-43. New York, NY.

Kennedy J, Kennedy JF, Eberhart RC, Shi Y (2001). Swarm intelligence. Morgan Kaufmann.

Nichols, M.D, Padgett, W.J. (2006). A Bootstrap control chart for Weibull percentiles. Quality and Reliability Engineering International 22, 141-151.

Shi Y, Eberhart R (1998). A modified particle swarm optimizer. In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pp. 69-73. IEEE.

carbone *Breaking stress of carbon fibres*

Description

The first real data set corresponds to an uncensored data set from Nichols and Padgett (2006) on breaking stress of carbon fibres (in Gba).

Usage

```
data(carbone)
```

Format

The format is: num [1:100] 3.7 2.74 2.73 2.5 3.6 3.11 3.27 2.87 1.47 3.11 ...

References

Nichols, M.D, Padgett, W.J. (2006). A Bootstrap control chart for Weibull percentiles. Quality and Reliability Engineering International 22, 141-151.

Examples

```
data(carbone)
hist(carbone)
```

descriptive *descriptive - Calculation of descriptive statistics*

Description

The function `descriptive` calculates the main descriptive statistics of a vector of data.

Usage

```
descriptive(x)
```

Arguments

x	Data vector.

Author(s)

Pedro Rafael Diniz Marinho <pedro.rafael.marinho@gmail.com>

Marcelo Bourguignon <m.p.bourguignon@gmail.com>

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988). The New S Language. Wadsworth & Brooks/Cole.

Examples

```
data(carbone)
descriptive(carbone)
```

`goodness.fit` *Adequacy of models*

Description

This function provides some useful statistics to assess the quality of fit of probabilistic models, including the statistics Cramér-von Mises and Anderson-Darling. These statistics are often used to compare models not fitted. You can also calculate other goodness of fit such as AIC, CAIC, BIC, HQIC and Kolmogorov-Smirnov test.

Usage

```
goodness.fit(pdf, cdf, starts, data, method = "PSO", domain = c(0,Inf),
             mle = NULL,...)
```

Arguments

`pdf`	Probability density function;
`cdf`	Cumulative distribution function;
`starts`	Initial parameters to maximize the likelihood function;
`data`	Data vector;
`method`	Method used for minimization of the function `-log(likelihood)`. The methods supported are: `PSO` (default), `BFGS`, `Nelder-Mead`, `SANN`, `CG`. Can also be transmitted only the first letter of the methodology, i.e., `P`, `B`, `N`, `S` or `C` respectively;
`domain`	Domain of probability density function. By default the domain of probability density function is the open interval 0 to infinity.This option must be an vector with two values;
`mle`	Vector with the estimation maximum likelihood. This option should be used if you already have knowledge of the maximum likelihood estimates. The default is `NULL`, ie, the function will try to obtain the estimates of maximum likelihoods;
`...`	If `method = "PSO"` or `method = "P"`, inform the arguments of the `pso` function. Get details about these arguments into `pso`. Basically the arguments that should be provided are the vectors `lim_inf` and `lim_sup`. The other parameters of the `pso` function can be informed in the desired configuration. However, may be omitted if the default configuration is sufficient.

Details

The function goodness.fit returns statistics KS (Kolmogorov-Smirnov), A (Anderson-Darling), W (Cramér-von Misses). Are also calculated other measures of goodness of fit. These functions are: AIC (Akaike Information Criterion), CAIC (Consistent Akaikes Information Criterion), BIC (Bayesian Information Criterion) and HQIC (Hannan-Quinn information criterion).

The Kolmogorov-Smirnov test may return NA with a certain frequency. The return NA informs that the statistical KS is not reliable for the data set used. More details about this issue can be obtained from ks.test.

By default, the function calculates the maximum likelihood estimates. The errors of the estimates are also calculated. In cases that the function can not obtain the maximum likelihood estimates, the change of the values initial, in some cases, resolve the problem. You can also enter with the maximum likelihood estimation if there is already prior knowledge.

Value

W	Statistic Cramér-von Misses;
A	Statistic Anderson Darling;
KS	Kolmogorov Smirnov test;
mle	Maximum likelihood estimates;
AIC	Akaike Information Criterion;
CAIC	Consistent Akaikes Information Criterion;
BIC	Bayesian Information Criterion;
HQIC	Hannan-Quinn information criterion;
Erro	Standard errors of the maximum likelihood estimates;
Value	Minimum value of the function -log(likelihood);
Convergence	0 indicates successful completion and 1 indicates that the iteration limit maxit had been reached. More details at optim.

Note

It is not necessary to define the likelihood function or log-likelihood. You only need to define the probability density function and distribution function.

Author(s)

Pedro Rafael Diniz Marinho <pedro.rafael.marinho@gmail.com>

References

Chen, G., Balakrishnan, N. (1995). A general purpose approximate goodness-of-fit test. Journal of Quality Technology, 27, 154-161.

Hannan, E. J. and Quinn, B. G. (1979). The Determination of the Order of an Autoregression. Journal of the Royal Statistical Society, Series B, 41, 190-195.

Nocedal, J. and Wright, S. J. (1999) Numerical Optimization. Springer.

Sakamoto, Y., Ishiguro, M. and Kitagawa G. (1986). Akaike Information Criterion Statistics. D. Reidel Publishing Company.

See Also

For details about the optimization methodologies may view the functions pso and optim.

Examples

```
# Example 1:

data(carbone)

# Exponentiated Weibull - Probability density function.
pdf_expweibull <- function(par,x){
  beta = par[1]
  c = par[2]
  a = par[3]
  a * beta * c * exp(-(beta*x)^c) * (beta*x)^(c-1) * (1 - exp(-(beta*x)^c))^(a-1)
}

# Exponentiated Weibull - Cumulative distribution function.
cdf_expweibull <- function(par,x){
  beta = par[1]
  c = par[2]
  a = par[3]
  (1 - exp(-(beta*x)^c))^a
}

set.seed(0)
result_1 = goodness.fit(pdf = pdf_expweibull, cdf = cdf_expweibull,
                        starts = c(1,1,1), data = carbone, method = "PSO",
                        domain = c(0,Inf),mle = NULL, lim_inf = c(0,0,0),
                        lim_sup = c(2,2,2), S = 250, prop=0.1, N=50)

x = seq(0, 6, length.out = 500)
hist(carbone, probability = TRUE)
lines(x, pdf_expweibull(x, par = result_1$mle), col = "blue")

# Example 2:

pdf_weibull <- function(par,x){
  a = par[1]
  b = par[2]
  dweibull(x, shape = a, scale = b)
}

cdf_weibull <- function(par,x){
  a = par[1]
  b = par[2]
  pweibull(x, shape = a, scale = b)
}

set.seed(0)
random_data2 = rweibull(250,2,2)
```

```
result_2 = goodness.fit(pdf = pdf_weibull, cdf = cdf_weibull, starts = c(1,1), data = random_data2,
           method = "PSO", domain = c(0,Inf), mle = NULL, lim_inf = c(0,0), lim_sup = c(10,10),
              N = 100, S = 250)

x = seq(0,ceiling(max(random_data2)), length.out = 500)
hist(random_data2, probability = TRUE)
lines(x, pdf_weibull(par = result_2$mle, x), col = "blue")

# TO RUN THE CODE BELOW, UNCOMMENT THE CODES.

# Example 3:

# Kumaraswamy Beta - Probability density function.
#pdf_kwbeta <- function(par,x){
#  beta = par[1]
#  a = par[2]
#  alpha = par[3]
#  b = par[4]
#  (a*b*x^(alpha-1)*(1-x)^(beta-1)*(pbeta(x,alpha,beta))^(a-1)*
#  (1-pbeta(x,alpha,beta)^a)^(b-1))/beta(alpha,beta)
#}

# Kumaraswamy Beta - Cumulative distribution function.
#cdf_kwbeta <- function(par,x){
#  beta = par[1]
#  a = par[2]
#  alpha = par[3]
#  b = par[4]
#  1 - (1 - pbeta(x,alpha,beta)^a)^b
#}

#set.seed(0)
#random_data3 = rbeta(150,2,2.2)

#system.time(goodness.fit(pdf = pdf_kwbeta, cdf = cdf_kwbeta, starts = c(1,1,1,1),
#             data = random_data3, method = "PSO", domain = c(0,1), lim_inf = c(0,0,0,0),
#              lim_sup = c(10,10,10,10), S = 200, prop = 0.1, N = 40))
```

---

pso *Adequacy of models*

Description

In computer science, the PSO is a computational method for optimization of parametric and multiparametric functions. The PSO algorithm is a meta-heuristic method, which has been providing good solutions for problems of global optimization functions with box-constrained. As in most heuristic methods that are inspired by biological phenomena, the PSO method is inspired by the

behavior of flying birds. The philosophical idea of the PSO algorithm is based on the collective behavior of birds (particle) in search of food (point of global optimal).

The pso function is an efficient function for global minimization, wherein it is not necessary to provide Initial kicks. This is the function for general purpose optimization.

Usage

```
pso(func, S = 350, lim_inf, lim_sup, e = 0.0001, data = NULL, N = 500, prop = 0.2)
```

Arguments

func	Objective function, i.e, function to be minimized;
S	Particle number considered. By default, S = 350;
lim_inf	Vector with the lower limit of the search for the parameters of the objective function that will be minimized;
lim_sup	Vector with the upper limits of search for the parameters of the objective function that will be minimized;
e	Stop value of the algorithm, i.e., if the variance of the last 20 minimum values is less than or equal to e, the algorithm will converge to the global minimum. By default, e = 0.0001;
data	Vector of data provided in the event of function to be minimized (passed as an argument for func) involve some data set. An example of a function that you should inform a data set is when we want to minimize the log-likelihood function multiplied by -1 (-log-likelihood). By defatul, data = NULL;
N	Minimum number of iterations. By default, N = 500;
prop	Proportion of last minimum value that is calculated variance used as a stopping criterion. That is, if the number of iterations is greater or equal to the minimum number of iterations N, calculate the variance of the last values of minimum obtained, wherein 0 <= prop <= 1.

Details

The PSO optimizes a problem by having a population of candidate solutions and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. The movement of the particles in the search space is randomized. Each iteration of the PSO algorithm, there is a leader particle, which is the particle that minimizes the objective function in the corresponding iteration. The remaining particles arranged in the search region will follow the leader particle randomly and sweep the area around this leading particle. In this local search process, another particle may become the new leader particle and the other particles will follow the new leader randomly. Each particle arranged in the search region has a velocity vector and position vector and its movement in the search region is given by changes in these vectors.

As a stopping criterion is considered the variance of the last 20 minimum values estimated by the algorithm. If this variance is less or equal the e the algorithm will stop providing the global minimum value. This is a conditional criterion, which will only be evaluated if the number of iterations is greater than or equal to the minimum number of iterations set to N.

The amount of minimum values considered in the calculation of the variance is given by the proportion of minimum values established by the argument prop which by default is prop = 0.2. That is, if the last 20% (prop = 0.2) of the minimum values has less variance than or equal to e, the algorithm will stop global search, indicating convergence according to the established criteria. This indicates that there was no significant improvements in this proportion of last iterations.

Value

par_pso	Global minimum point;
f_pso	Global minimum value.

Note

In other versions of the package, the paper with more details that complement the documentation of this function will be provided in the above references and this note will be undone.

Author(s)

Pedro Rafael Diniz Marinho <pedro.rafael.marinho@gmail.com>

References

Beni G, Wang J (1993). Swarm intelligence in cellular robotic systems. pp. 703-712.

Eberhart RC, Kennedy J (1995). A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science, volume 1, pp. 39-43. New York, NY.

Kennedy J, Kennedy JF, Eberhart RC, Shi Y (2001). Swarm intelligence. Morgan Kaufmann.

Shi Y, Eberhart R (1998). A modified particle swarm optimizer. In Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pp. 69-73. IEEE.

Examples

```
# The objective functions below are rather difficult to be optimized.
# However, the function pso has great results.

# Example 1 (Easom function):

easom_function <- function(par,x){
  x1 = par[1]
  x2 = par[2]
  -cos(x1)*cos(x2)*exp(-((x1-pi)^2 + (x2-pi)^2))
}

set.seed(0)
result_1 = pso(func = easom_function, S = 500, lim_inf = c(-10,-10), lim_sup = c(10,10),
               e = 0.00001)
result_1$par
```

```
# Example 2 (Holder table function):

holder <- function(par,x){
  x1 = par[1]
  x2 = par[2]
  -abs(sin(x1)*cos(x2) * exp(abs(1 - sqrt(x1^2+x2^2)/pi)))
}

set.seed(0)
result_2 = pso(func = holder, S = 700, lim_inf = c(-10,-10), lim_sup = c(10,10),
               e = 0.00001, N=500)
result_2$par

# Example 3:

f_pso <- function(par,x){
  theta = par[1]
  -(6 + theta^2 * sin(14*theta))
}

set.seed(0)
result_3 <- pso(func = f_pso, S = 500, lim_inf = c(-2.5), lim_sup = c(2.5), e = 0.0001)
result_3$par

# TO RUN THE CODE BELOW, UNCOMMENT THE CODES.

# Example 4 (maximizing a function of the log-likelihood function):

# pdf_exp <- function(par,x){
#  lambda = par[1]
#  lambda*exp(-lambda*x)
#}

# -log-likelihood function of the exponential distribution.
#likelihood <- function(par,x){
#  lambda = par[1]
#  -sum(log(pdf_exp(par,x)))
#}

#set.seed(0)
#random_data1 = rexp(500,1)
#result_1 = pso(func = likelihood, S = 250, lim_inf = c(0), lim_sup = c(100), e = 0.0001,
#    data = random_data1, N = 50, prop = 0.2)

#x = seq(0,ceiling(max(random_data1)), length.out = 500)
#hist(random_data1, probability = TRUE)
#lines(x, pdf_exp(par = result_1$par, x), col = "blue")

# Example 5 (maximizing a function of the log-likelihood function):

# Probability density function (Weibull)
#pdf_weibull <- function(par,x){
#  a = par[1]
```

```
#  b = par[2]
#  dweibull(x,shape=a,scale=b)
#}

# -log-likelihood function of the Weibull distribution.
#likelihood <- function(par,x){
#  -sum(log(pdf_weibull(par,x)))
#}

#set.seed(0)
#random_data2 = rweibull(250,2,2)
#result_2 = pso(func = likelihood, S = 250, lim_inf = c(0,0), lim_sup = c(10,10), e = 0.0001,
#               data = random_data2, N = 50, prop = 0.2)

#x = seq(0,ceiling(max(random_data2)), length.out = 500)
#hist(random_data2, probability = TRUE, ylim = c(0,0.5))
#lines(x, pdf_weibull(par = result_2$par, x), col = "blue")
```

TTT *TTT function*

Description

There are several behaviors that the failure rate function of a random variable T can take. In this context, the graph of total test time (TTT curve) proposed by Aarset (1987) may be used for obtaining empirical behavior of the function failure rate.

Usage

```
TTT(x, lwd = 2, lty = 2, col = "black", grid = TRUE,...)
```

Arguments

x	Data vector;
lwd	Thickness of the TTT curve. The argument lwd must be a nonnegative real number;
lty	The argument lty modifies the style of the diagonal line chart TTT. Possible values are: 0 [blank], 1 [solid (default)], 2 [dashed], three [dotted], 4 [dotdash], 5 [longdash], 6 [twodash];
col	Color used in the TTT curve;
grid	If grid = FALSE graphic appears without the grid;
...	Other arguments passed by the user and available for the function plot. More details in par.

Note

The graphic TTT may have various forms. Aarset (1987) showed that if the curve approaches a straight diagonal function constant failure rate is adequate. When the curve is convex or concave the failure rate function is monotonically increasing or decescente respectively is adequate. If the failure rate function is convex and concave, the failure rate function in format U is adequate, otherwise the failure rate function unimodal is more appropriate.

The TTT curve is constructed by values r/n and $G(r/n)$, wherein

$$G(r/n) = \frac{[\sum_{i=1}^{r} T_{i:n} + (n-r)T_{r:n}]}{\sum_{i=1}^{n} T_{i:n}}, r = 1, \ldots, n, T_{1:n} = 1, \ldots, n.$$

Author(s)

Pedro Rafael Diniz Marinho (pedro.rafael.marinho@gmail.com);

Marcelo Bourguignon (m.p.bourguignon@gmail.com).

References

Aarset, M. V. (1987). How to identify bathtub hazard rate. IEEE Transactions Reliability, 36, 106-108.

Examples

```
data(carbone)
TTT(carbone, col = "red", lwd = 2.5, grid = TRUE, lty = 2)
```

Index