Package ‘SEL’

November 25, 2023

Type Package

Title Semiparametric Elicitation

Version 1.0-4

Date 2023-11-25

Depends splines, quadprog, lattice

Author Bjoern Bornkamp

Maintainer Bjoern Bornkamp <bbnkmp@mail.de>

Description Implements a method for fitting a bounded probability distribution to quantiles (for example stated by an expert), see Bornkamp and Ickstadt (2009) for details. For this purpose B-splines are used, and the density is obtained by penalized least squares based on a Brier entropy penalty. The package provides methods for fitting the distribution as well as methods for evaluating the underlying density and cdf. In addition methods for plotting the distribution, drawing random numbers and calculating quantiles of the obtained distribution are provided.

License GPL

Repository CRAN

Date/Publication 2023-11-25 15:50:02 UTC

NeedsCompilation yes

R topics documented:

`SEL-package` *Semiparametric Elicitation of a bounded parameter.*

Description

This package implements a novel method for fitting a bounded probability distribution to quantiles stated for example by an expert (see Bornkamp and Ickstadt (2009)). For this purpose B-splines are used, and the density is obtained by penalized least squares based on a Brier entropy penalty. The package provides methods for fitting the distribution as well as methods for evaluating the underlying density and cdf. In addition methods for plotting the distribution, drawing random numbers and calculating quantiles of the obtained distribution are provided.

Details

Package:	SEL
Type:	Package
Version:	1.0-2
Date:	2010-05-21
License:	GPL

Author(s)

Bjoern Bornkamp

Maintainer: Bjoern Bornkamp <bornkamp@statistik.tu-dortmund.de>

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

O'Hagan A., Buck C. E., Daneshkhah, A., Eiser, R., Garthwaite, P., Jenkinson, D., Oakley, J. and Rakow, T. (2006), *Uncertain Judgements: Eliciting Expert Probabilities*, John Wiley and Sons Inc.

Garthwaite, P., Kadane, J. O'Hagan, A. (2005), Statistical Methods for Eliciting Probability Distributions, *Journal of the American Statistical Association*, **100**, 680–701

Dierckx, P. (1993), *Curve and Surface Fitting with Splines*, Clarendon Press.

Examples

```
## example from O'Hagan et al. (2006)
x <- c(177.5, 183.75, 190, 205, 220)
y <- c(0.175, 0.33, 0.5, 0.75, 0.95)

default   <- SEL(x, y, Delta = 0.05, bounds = c(165, 250))
bernst    <- SEL(x, y, d = 10, N = 0, Delta = 0.05, bounds = c(165, 250))
unifknots <- SEL(x, y, d = 3, N = 5, Delta = 0.05, bounds = c(165, 250))
lin       <- SEL(x, y, d = 1, inknts = x, Delta = 0.05, bounds = c(165, 250))
comparePlot(default, bernst, unifknots, lin, type = "cdf")
comparePlot(default, bernst, unifknots, lin, type = "density")

## compare summaries
summary(default)
summary(bernst)
summary(unifknots)
summary(lin)

## sample from SEL object and evaluate density
xxx <- rvSEL(50000, bernst)
hist(xxx, breaks=100, freq=FALSE)
curve(predict(bernst, newdata=x), add=TRUE)
```

---

comparePlot *Compare different elicitated densities.*

Description

Compare different elicitated distributions in a trellis display.

Usage

```
comparePlot(..., type = c("density", "cdf"), deriv,
             points = TRUE, superpose = FALSE, n = 101,
             xlab= "", ylab, addArgs = NULL)
```

Arguments

`...`	Fitted SEL objects separated by a comma.
`type`	Determines whether to plot densities or cdfs (ignored if `deriv` is not missing).
`deriv`	Specify derivative of the expert's density to be plotted.
`points`	Logical indicating whether elicited quantiles should be displayed, when displaying the cdf.
`superpose`	Logical indicate if plots should be superposed.
`n`	Integer, number of points used for plotting.

xlab, ylab — Labels for x-axis and y-axis. If ylab is missing the value of type is used.

addArgs — List specifying additional arguments for xyplot function, list elements should match the corresponding arguments of the xyplot function.

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

See Also

SEL, plot.SEL

Examples

```
# example from O'Hagan et al. (2006)
x <- c(177.5, 183.75, 190, 205, 220)
y <- c(0.175, 0.33, 0.5, 0.75, 0.95)

default   <- SEL(x, y, Delta = 0.05, bounds = c(165, 250))
bernst    <- SEL(x, y, d = 10, N = 0, Delta = 0.05, bounds = c(165, 250))
unifknots <- SEL(x, y, d = 3, N = 5, Delta = 0.05, bounds = c(165, 250))
lin       <- SEL(x, y, d = 1, inknts = x, Delta = 0.05, bounds = c(165, 250))
comparePlot(default, bernst, unifknots, lin, type = "cdf")
comparePlot(default, bernst, unifknots, lin, type = "density")
```

---

fit.SEL — *Fit an SEL object*

Description

Function that performs the actual fitting of the expert's distribution via quadratic programming (using the solve.QP function from the quadprog package). This function is mainly for internal use.

Usage

```
fit.SEL(N, alpha, P, A, b, gamma, dplus = 0)
```

Arguments

`N`	Matrix containing the B-spline basis functions evaluated at the elicited quantiles.
`alpha`	Vector giving the levels of the elicited quantiles.
`P`	Penalty matrix (called Omega in the reference below).
`A,b`	Matrix and vector containing the specifying the constraints A'b >= 0.
`gamma`	Gamma parameter trading of goodness of fit and Brier entropy/Brier divergence.
`dplus`	Additional offset for dvec in `solve.QP`.

Value

Returns solution of the quadratic programming problem.

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

Goldfarb, D., and Idnani, A. (1982), Dual and Primal-Dual Methods for Solving Strictly Convex Quadratic Programs, in *Numerical Analysis*, (eds.) J. Hennart, Springer Verlag, Berlin, pp. 226–239.

Goldfarb, D., and Idnani, A. (1983), A Numerically Stable Dual Method for Solving Strictly Convex Quadratic Programs", *Mathematical Programming*, 27, 1–33.

See Also

`SEL`, `solve.QP`

`getbinPost` *Calculate posterior distribution for binomial model*

Description

Calculate posterior distribution for the simple beta-binomial model

Usage

```
getbinPost(x, object, k, n, type = c("density", "cdf"),
           rel.tol = .Machine$double.eps^0.5)
```

Arguments

x	Vector specifying, where posterior distribution should be evaluated.
object	An SEL object.
k	Number of observed successes in the binomial model.
n	Number of total trials.
type	Character specifying, whether posterior density or cdf should be evaluated.
rel.tol	`rel.tol` argument of the integrate subroutine, which numerically calculates the normalization constant, when a non-conjugate prior is used (ie a B-spline basis not leading to a Bernstein mixture).

Value

Numeric containing the function values corresponding to x values

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

Diaconis, P., and Ylvisaker, D. (1985), Quantifying Prior Opinion, *Bayesian Statistics 2*, (eds.) J.M. Bernardo, M.H. DeGroot, D.V. Lindley and A.F.M. Smith, Elsevier Science Publishers B.V., Amsterdam, pp. 133–156.

See Also

SEL

Examples

```
## Diaconis, Ylvisaker spun coin example (see references)
## simulate elicitations
x <- seq(0, 1, length=9)[2:8]
ymu <- 0.5*pbeta(x, 10, 20)+0.5*pbeta(x, 20, 10)
sig <- 0.05
set.seed(4)
y <- ymu+rnorm(7, 0, sqrt(ymu*(1-ymu))*sig)

## perform fitting with different selections
A <- SEL(x, y, d = 1, Delta = 0.001, inknts = x)
foo1 <- function(x) dbeta(x, 0.5, 0.5)
B <- SEL(x, y, d = 19, N=0, Delta = 0.02, pistar = foo1)
C <- SEL(x, y, d = 19, N=0, Delta = 0.05, pistar = foo1)
comparePlot(A, B, C, addArgs = list(layout = c(3,1)))

## posterior
sq <- seq(0,1,length=201)
```

```
res1 <- getbinPost(sq, A, 3, 10, type = "density")
res2 <- getbinPost(sq, B, 3, 10, type = "density")
res3 <- getbinPost(sq, C, 3, 10, type = "density")

## parametric posterior (corresponding to B(0.5,0.5) prior)
plot(sq, dbeta(sq, 3.5, 7.5), type="l", xlab = "",
     ylab = "Posterior density", lty = 4,
     ylim=c(0,max(c(res1, res2, res3))))
## "semiparametric" posteriors
lines(sq, res1, lty = 1)
lines(sq, res2, lty = 2)
lines(sq, res3, lty = 3)
legend(0.65,4, legend=c("Scenario A", "Scenario B", "Scenario C",
       "B(0.5,0.5) prior"), lty = 1:4)
```

---

knotave — *Calculate the knot averages of a B-spline basis.*

Description

Calculates the knot averages of a B-spline basis.

Usage

```
knotave(knots, d)
```

Arguments

knots	Knot Vector (with d+1 coincident knots on the boundaries).
d	Degree of the B-spline basis.

Value

Numeric containing knot averages

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

Dierckx, P. (1993), *Curve and Surface Fitting with Splines*, Clarendon Press

See Also

SEL

Examples

```
## Example for calculation of a control polygon
knts <- c(rep(0, 4), rep(1, 4))
cf <- c(-1, -1, 1/2, 0)
sq <- seq(0, 1, length = 101)
N <- splineDesign(sq, knots = knts, ord = 4)
res <- colSums(t(N)*cf)
plot(sq, res, type = "l", ylim = c(-1, 0.6))
kntAv <- knotave(knts, 3)
lines(kntAv, cf, col = "red") # add control polygon
```

---

min_abs *Find gamma given Delta*

Description

Find gamma so that squareroot of average squared distance of fitted distribution and statements is equal to `Delta` (using the `uniroot` function). This function is not meant to be called by the user, unless one is familiar with the source code.

Usage

```
min_abs(Delta, N, alpha, P, A, b, lb, ub, dplus = 0)
```

Arguments

`Delta`	Average root of squared error to be achieved with fitted distribution.
`N`	Matrix containing the B-spline basis functions evaluated at the elicited quantiles.
`alpha`	Vector giving the levels of the elicited quantiles.
`P`	Penalty matrix (called Omega in reference below).
`A,b`	Matrix and vector containing the specifying the constraints A'b >= 0.
`lb, ub`	lower and upper bound to search for gamma.
`dplus`	offset integral needed when divergence instead of entropy is used.

Value

Returns gamma

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

See Also

uniroot, solve.QP

---

plot.SEL *Plotting an SEL object*

Description

This function displays a fitted SEL object and displays the expert's cdf or pdf (depending on the deriv argument)

Usage

```
## S3 method for class 'SEL'
plot(x, ..., type = c("density", "cdf"), deriv,
      points = TRUE, n = 101, xlab="", ylab="", ylim)
```

Arguments

x	An SEL object.
...	Additional arguments to plot function.
type	Determines whether to plot the expert's density or cdf (only if deriv is missing).
deriv	Determines which derivative of the expert's distribution should be plotted. If specified the type argument is ignored.
points	Logical indicating whether elicited quantiles should be displayed, when displaying the cdf.
n	Integer, number of points used for plotting.
xlab, ylab	Label of x-axis and y-axis.
ylim	Limits of y axis.

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

See Also

comparePlot, SEL

Examples

```
# example from O'Hagan et al. (2006)
x <- c(177.5, 183.75, 190, 205, 220)
y <- c(0.175, 0.33, 0.5, 0.75, 0.95)

fit <- SEL(x, y, Delta = 0.05, bounds = c(165, 250))
plot(fit)
plot(fit, type = "cdf")
plot(fit, deriv = 1)
```

---

predict.SEL *Evaluate the expert's density (or cdf)*

Description

Evaluate the density or cdf of an SEL object.

Usage

```
## S3 method for class 'SEL'
predict(object, newdata = seq(object$bounds[1],
      object$bounds[2], length = 101), type = c("density", "cdf"), deriv, ...)
```

Arguments

`object`	An SEL object.
`newdata`	Where to evaluate the distribution.
`type`	Determines whether to evaluate the expert's density or cdf (only if `deriv` is missing).
`deriv`	Determines which derivative of the expert's distribution should be evaluated.
`...`	...

Value

A numeric vector

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

See Also

SEL

Examples

```
# example from O'Hagan et al. (2006)
x <- c(177.5, 183.75, 190, 205, 220)
y <- c(0.175, 0.33, 0.5, 0.75, 0.95)

default   <- SEL(x, y, Delta = 0.05, bounds = c(165, 250))
predict(default, newdata = c(200, 205))
```

---

quantSEL — *Calculate quantiles of an SEL object.*

Description

Returns the quantiles of the fitted distribution.

Usage

```
quantSEL(q, object, nPoints = 1000)
```

Arguments

`q`	A vector of quantiles.
`object`	An SEL object.
`nPoints`	Number of evaluations for the brute force inversion of the expert cdf (see details).

Details

The inverse of the distribution function is formed by evaluating the distribution function at nPoints points and interchanging the role of dependent and independent variable when building a function interpolating the data (using `splinefun` with `"monoH.FC"` option). Note that there are also direct ways of inverting a B-spline function, which however turned out to be less efficient for our purposes.

Value

A vector of quantiles.

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

See Also

SEL, splinefun

Examples

```
## example from O'Hagan et al. (2006)
x <- c(177.5, 183.75, 190, 205, 220)
y <- c(0.175, 0.33, 0.5, 0.75, 0.95)

default   <- SEL(x, y, Delta = 0.05, bounds = c(165, 250))
bernst    <- SEL(x, y, d = 10, N = 0, Delta = 0.05, bounds = c(165, 250))
unifknots <- SEL(x, y, d = 3, N = 5, Delta = 0.05, bounds = c(165, 250))
lin       <- SEL(x, y, d = 1, inknts = x, Delta = 0.05, bounds = c(165,250))
quantSEL(c(0.25, 0.5, 0.75), default)
quantSEL(c(0.25, 0.5, 0.75), bernst)
quantSEL(c(0.25, 0.5, 0.75), unifknots)
quantSEL(c(0.25, 0.5, 0.75), lin)
```

---

rvSEL *Simulate from the expert's distribution*

Description

Simulate random variables from an SEL object.

Usage

```
rvSEL(n, object, nPoints = 1000)
```

Arguments

`n`	Number of simulated values.
`object`	An SEL object.
`nPoints`	Number of evaluations for the brute force inversion of the expert cdf.

Details

The inverse of the distribution function is formed by evaluating the distribution function at nPoints points and interchanging the role of dependent and independent variable when building an function interpolating the data (using `splinefun` with `"monoH.FC"` option). Note that there are also direct ways of inverting a B-spline function, which however turned out to be less efficient for our purposes.

Value

A numeric vector containing pseudo-random variates from the expert's density.

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

See Also

SEL, splinefun

Examples

```
## bimodal example
x2 <- c(0.1, 0.2, 0.5, 0.8, 0.9)
y2 <- c(0.2, 0.4, 0.45, 0.85, 0.99)

fit1 <- SEL(x2, y2, Delta=0.05, d = 4, inknts = x2)
fit2 <- SEL(x2, y2, Delta=0.05, d = 15, N = 0)
comparePlot(fit1, fit2, superpose = TRUE)

## sample from SEL object
xxx <- rvSEL(50000, fit1)
hist(xxx, breaks=100, freq=FALSE)
curve(predict(fit1, newdata=x), add=TRUE)
```

SEL *Semiparametric Elicitation of a bounded parameter*

Description

Fits a distribution to quantiles (stated for example by an expert) using B-splines with a Brier entropy penalty.

There exists a `print` and a `summary` method to display details of the fitted SEL object. The fitted density (or cdf) can be displayed with the `plot` method. Different SEL objects can be displayed in one plot with the `comparePlot` function. The fitted density (or cdf) can be evaluated with the `predict` method. The `coef` method extracts the coefficients of the fitted B-spline basis. The `quantSEL` function calculates quantiles of the fitted distribution and the `rvSEL` function generates random variables from a fitted SEL object.

Usage

```
SEL(x, alpha, bounds = c(0, 1), d = 4, inknts = x, N, gamma, Delta,
    fitbnds = c(1e-8, 10)*diff(bounds), pistar = NULL,
    constr = c("none", "unimodal", "decreasing", "increasing"),
    mode)
```

Arguments

`x`	Numeric vector of quantiles.
`alpha`	Numeric vector determining the levels of the quantiles specified in `x`.
`bounds`	Vector containing the bounds for the expert's density.
`N`	Number of equally spaced inner knots (ignored if `inknts` is specified).
`d`	Degree of the spline (the order of the spline is d+1), values larger than 20 are to be avoided; they can lead to numerical instabilities.
`inknts`	Vector specifying the inner knots. If equal to `NULL`, the B-spline basis reduces to the Bernstein polynomial basis. Per default the inner knots are located on the values specified via `x`.
`gamma`	Parameter controlling the weight of the negative entropy in the objective function to be optimized (see Bornkamp and Ickstadt (2009)).
`Delta`	The code calculates the `gamma` parameter achieving a certain overall L2 distance between the specified quantiles and the fitted distribution function (only if `gamma` is not specified).
`fitbnds`	Numeric of length 2 for the bisection search algorithm searching for gamma giving the error Delta (only relevant if gamma is missing).
`pistar`	Prior density function used in Brier divergence (see Bornkamp and Ickstadt (2009). If `NULL` Brier entropy is used.
`constr`	Character vector specifying, which shape constraint should be used, should be one of "none", "unimodal", "decreasing", "increasing".
`mode`	Numerical value needed when `constraint = "unimodal"`. The code selects the knot average closest to `mode` as the location of the mode for the finite dimensional constraints on the coefficients of the B-spline in the optimization (note that the final density will only have a mode close to the value specified via `mode`).

Value

An SEL object containing the following entries

`constr`	Character determining the shape constraint used.
`inknts`	Inner knots.
`nord`	Order of the spline.
`bounds`	Bounds for the elicited quantity.
`xalpha`	List containing the specified quantiles.
`Omega`	Penalty matrix used for calculating Brier entropy.
`coefs`	Fitted coefficients of the B-spline basis.
`pistar`	Function handed over to `SEL` via the `pistar` argument (if any).
`dplus`	The d vector, only necessary when pistar is specified (see Bornkamp and Ickstadt (2009).)

Author(s)

Bjoern Bornkamp

References

Bornkamp, B. and Ickstadt, K. (2009). A Note on B-Splines for Semiparametric Elicitation. *The American Statistician*, **63**, 373–377

O'Hagan A., Buck C. E., Daneshkhah, A., Eiser, R., Garthwaite, P., Jenkinson, D., Oakley, J. and Rakow, T. (2006), *Uncertain Judgements: Eliciting Expert Probabilities*, John Wiley and Sons Inc.

Dierckx, P. (1993), *Curve and Surface Fitting with Splines*, Clarendon Press

See Also

`plot.SEL`, `comparePlot`, `quantSEL`, `rvSEL`

Examples

```
### example from O'Hagan et al. (2006)
### 1st example in Bornkamp and Ickstadt (2009)
x <- c(177.5, 183.75, 190, 205, 220)
y <- c(0.175, 0.33, 0.5, 0.75, 0.95)
I <- SEL(x, y, d = 1, Delta = 0.015, bounds = c(165, 250), inknts = x)
II <- SEL(x, y, d = 14, N = 0, Delta = 0.015, bounds = c(165, 250))
III <- SEL(x, y, d = 4, Delta = 0.015, bounds = c(165, 250), inknts = x)
IV <- SEL(x, y, d = 4, Delta = 0.015, bounds = c(165, 250), inknts = x,
      constr = "u", mode = 185)
comparePlot(I, II, III, IV, type = "cdf")
comparePlot(I, II, III, IV, type = "density")



### bimodal example
x2 <- c(0.1, 0.2, 0.5, 0.8, 0.9)
y2 <- c(0.2, 0.4, 0.45, 0.85, 0.99)
fit1 <- SEL(x2, y2, Delta=0.05, d = 4, inknts = x2)
fit2 <- SEL(x2, y2, Delta=0.05, d = 15, N = 0)
comparePlot(fit1, fit2, superpose = TRUE)



### sample from SEL object and evaluate density
xxx <- rvSEL(50000, fit1)
hist(xxx, breaks=100, freq=FALSE)
curve(predict(fit1, newdata=x), add=TRUE)



### illustrate shrinkage against uniform dist.
gma01 <- SEL(x2, y2, gamma = 0.1)
gma02 <- SEL(x2, y2, gamma = 0.2)
gma04 <- SEL(x2, y2, gamma = 0.4)
```

```
gma10 <- SEL(x2, y2, gamma = 1.0)
comparePlot(gma01, gma02, gma04, gma10, superpose = TRUE)



### including shape constraints
x <- c(177.5, 183.75, 190, 205, 220)
y <- c(0.175, 0.33, 0.5, 0.75, 0.95)
unconstr1 <- SEL(x, y, Delta = 0.05, bounds = c(165, 250))
unconstr2 <- SEL(x, y, d = 10, N = 0, Delta = 0.05, bounds = c(165,250))
unimod1 <- SEL(x, y, Delta = 0.05, bounds = c(165, 250),
            constr = "unimodal", mode = 185)
unimod2 <- SEL(x, y, d = 10, N = 0, Delta = 0.05, bounds = c(165, 250),
           constr =  "unimodal", mode = 185)
comparePlot(unconstr1, unconstr2, unimod1, unimod2)



### shrinkage against another distribution
pr <- function(x) dbeta(x, 2, 2)
pr01  <- SEL(x2, y2, gamma = 0.1, d = 3, pistar = pr)
pr03  <- SEL(x2, y2, gamma = 0.3, d = 3, pistar = pr)
pr12 <- SEL(x2, y2, gamma = 1.2, pistar = pr)
comparePlot(pr01, pr03, pr12, superpose = TRUE)



### 2nd example from Bornkamp and Ickstadt (2009)
# theta
# "true" density
pmixbeta <- function(x, a1, b1, a2, b2){
  0.3*pbeta(x/20,a1,b1)+0.7*pbeta(x/20,a2,b2)
}
dmixbeta <- function(x, a1, b1, a2, b2){
  out <- 0.3*dbeta(x/20,a1,b1)+0.7*dbeta(x/20,a2,b2)
  out/20
}
x <- c(2,10,15)
a1 <- 1;a2 <- 10;b1 <- 15;b2 <- 5
mu <- pmixbeta(x, a1, b1, a2, b2)
set.seed(1) # simulate experts statements
y <- rnorm(length(mu), mu, sqrt(mu*(1-mu)*0.05^2))
thet <- SEL(x, y, d = 4, Delta = 0.03, bounds = c(0, 20), inknts = x)
plot(thet, ylim = c(0,0.25))
curve(dmixbeta(x, a1, b1, a2, b2), add=TRUE, col="red")
abline(h=0.05, lty = 2)
legend("topright", c("true density", "elicited density", "uniform density"),
       col=c(2,1,1), lty=c(1,1,2))

# sigma
# "true" density
dtriang <- function(x,m,a,b){
  inds <- x < m;res <- numeric(length(x))
```

```
  res[inds] <- 2*(x[inds]-a)/((b-a)*(m-a))
  res[!inds] <- 2*(b-x[!inds])/((b-a)*(b-m))
  res
}
ptriang <- function(x,m,a,b){
  inds <- x < m;res <- numeric(length(x))
  res[inds] <- (x[inds]-a)^2/((b-a)*(m-a))
  res[!inds] <- 1-(b-x[!inds])^2/((b-a)*(b-m))
  res
}
x <- c(1,2,4)
mu <- ptriang(x, 1, 0, 5)
set.seed(1) # simulate experts statements
y <- rnorm(length(mu), mu, sqrt(mu*(1-mu)*0.05^2))
sig <- SEL(x, y, d = 4, Delta = 0.03, bounds = c(0, 5),
       inknts = x, mode = 1, constr="unimodal")
plot(sig, ylim=c(0,0.4))
curve(dtriang(x, 1, 0, 5), add=TRUE, col="red")
abline(h=0.2, lty = 2)
legend("topright", c("true density", "elicited density", "uniform density"),
       col=c(2,1,1), lty=c(1,1,2))


## Not run:
### generate some random elicitations
numb <- max(rnbinom(1, mu = 4, size = 1), 1)
x0 <- runif(1, -1000, 1000)
x1 <- x0+runif(1, 0, 1000)
xs <- sort(runif(numb, x0, x1))
y <- runif(numb+1)
ys <- (cumsum(y)/sum(y))[1:numb]
fit1 <- SEL(xs, ys, bounds = c(x0, x1))
fit2 <- SEL(xs, ys, d = 1, inknts = xs, bounds = c(x0, x1))
fit3 <- SEL(xs, ys, d = 15, N = 0, bounds = c(x0, x1))
comparePlot(fit1, fit2, fit3, type="cdf")

## End(Not run)
```

Index