Package 'BALD'

October 22, 2018

Maintainer Can Wang <bald@frankschmid.com>

License GPL (>= 3)

Title Robust Loss Development Using MCMC

LazyLoad yes

Author Can Wang, Christopher W. Laws, Frank A. Schmid

SystemRequirements JAGS (>= 4.3.0), GNU make

Description Bayesian analysis of loss development on insurance triangles or 'BALD' is a Bayesian model of developing aggregate loss triangles in property casualty insurance. This actuarial model makes use of a heteroskedastic and skewed t-likelihood with endogenous degrees of freedom, employs model averaging by means of Reversible Jump MCMC, and accommodates a structural break in the path of the consumption of benefits. Further, the model is capable of incorporating expert information in the calendar year effect. In an accompanying vignette, this model is applied to two widely studied General Liability and Auto Bodily Injury Liability loss triangles. For a description of the methodology, see Frank A. Schmid (2010) <doi:10.2139/ssrn.1501706>.

Version 1.0.0-3

Repository CRAN

Date 2018-10-19

Imports stats, graphics

Suggests R.rsp

VignetteBuilder R.rsp

Depends R (>= 3.3.0), methods, rjags (>= 3-3), logspline, utils, lattice

Collate 'zzz.R' 'LossDevModelInput.R' 'AnnualAggLossDevModelInput.R' 'LossDevModelOutput.R' 'NodeOutput.R' 'AnnualAggLossDevModelOutput.R' 'BreakAnnualAggLossDevModelOutput.R' 'StandardAnnualAggLossDevModelOutput.R' 'AnnualAggLossDevModelOutputWithZeros.R' 'BALD.R' 'BreakAnnualAggLossDevModelInput.R' 'DocumentationOfData.R' 'StandardAnnualAggLossDevModelInput.R' 'util.R'

NeedsCompilation yes

Date/Publication 2018-10-22 12:00:07 UTC

R topics documented:

`accountForZeroPayments` *A function to take a triangle estimated without considering zero payments, and account for the possibility of zero payments.*

Description

A function to take a triangle estimated without considering zero payments, and account for the possibility of zero payments.

Usage

```
accountForZeroPayments(object, burnIn=1000, nAddapt=1000)
```

Arguments

`object`	The object containing the triangle estimated without accounting for zero payments.
`burnIn`	An integer to represent the number of initial MCMC iterations to be discarded. (The adaptive phase (`nAddapt`) is not considered part of `burnIn`.)
`nAddapt`	The length of the adaptive phase for the MCMC algorithm. (Default is `trunc(burnIn/4)+1`.)

Details

As incremental payments are modeled on the log scale, zero payments (and negative payments) are treated as missing values. So, without somehow accounting for zero payments, the estimated payments would be overstated. Zero payments are accounted for by weighting the predicted payment (given that the payment is greater than zero) with the probability that this payment is zero. (Negative payments are not (currently) accounted for.) Currently the trajectory for this probably follows a gompertz curve and is constant across exposure years. This is currently implemented as a function but may be switched to a method. See `vignette('BALD')`.

Examples

```
rm(list=ls())
library(BALD)
data(CumulativeAutoBodilyInjuryTriangle)
CumulativeAutoBodilyInjuryTriangle <- as.matrix(CumulativeAutoBodilyInjuryTriangle)
sample.col <- (dim(CumulativeAutoBodilyInjuryTriangle)[2] - 6:0)
print(decumulate(CumulativeAutoBodilyInjuryTriangle)[1:7, sample.col])
data(HPCE)
HPCE <- as.matrix(HPCE)[,1]
HPCE.rate <- HPCE[-1] / HPCE[-length(HPCE)] - 1
print(HPCE.rate[(-10):0 + length(HPCE.rate)])
HPCE.years <- as.integer(names(HPCE.rate))
max.exp.year <- max(as.integer(
dimnames(CumulativeAutoBodilyInjuryTriangle)[[1]]))
years.to.keep <- HPCE.years <=  max.exp.year + 3
```

```
HPCE.rate <- HPCE.rate[years.to.keep]
break.model.input <- makeBreakAnnualInput(
cumulative.payments = CumulativeAutoBodilyInjuryTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = HPCE.rate,
first.year.in.new.regime = c(1986, 1987),
prior.for.first.year.in.new.regime=c(2,1),
exp.year.type = 'ay',
extra.dev.years = 5,
use.skew.t = TRUE,
bound.for.skewness.parameter=5)
## Not run:
break.model.output <- runLossDevModel(
break.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
break.model.output.w.zeros <- accountForZeroPayments(break.model.output)

## End(Not run)
```

AnnualAggLossDevModelInput-class

The base class for all aggregate models.

Description

The base class for all aggregate models.

Details

`AnnualAggLossDevModelInput` is the base class for all aggregate model input objects. It is derived from `LossDevModelInput`.

See Also

`LossDevModelInput`, `StandardAnnualAggLossDevModelInput`

AnnualAggLossDevModelOutput-class

The base output class for all aggregate annual models.

Description

The base output class for all aggregate annual models.

Details

AnnualAggLossDevModelOutput is the base output class for all aggregate annual model objects. Derived classes should contain all output from a JAGS run of the input object in the slot "input". Currenly only the slot "input" is allowed to be a non-model node. All other nodes should be the exact name of some settable node in the model. This is because getModelOutputNodes currently looks at the slot names to determine what values to set; only slot "input" is known to be a slot other than a settable node. This class is derived from LossDevModelOutput

See Also

LossDevModelOutput

AnnualAggLossDevModelOutputWithZeros-class

The parent of StandardAnnualAggLossDevModelOutputWithZeros and BreakAnnualAggLossDevModelOutputWithZeros.

Description

The parent of StandardAnnualAggLossDevModelOutputWithZeros and BreakAnnualAggLossDevModelOutputWithZeros

Details

To avoid creating multiple inheritance (directly), this class is created using setClassUnion. The union consists of classes StandardAnnualAggLossDevModelOutputWithZeros and StandardAnnualAggLossDevModelOut

See Also

LossDevModelOutput BreakAnnualAggLossDevModelOutputWithZeros StandardAnnualAggLossDevModelOutputWithZ

---

autoregressiveParameter

A generic function to plot and/or return the posterior of the autoregressive parameter for models in BALD.

Description

A generic function to plot and/or return the posterior of the autoregressive parameter for models in **BALD**. See vignette('BALD').

Arguments

object	The object from which to plot and/or return the autoregressive parameter.
plotDensity	A logical value. If TRUE, the density is plotted. If plotTrace is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.
plotTrace	A logical value. If TRUE, the trace is plotted. If plotDensity is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.

Value

Mainly called for the side effect of plotting.

See Also

autoregressiveParameter("AnnualAggLossDevModelOutput") standardDeviationOfCalendarYearEffect calendarYearEffect calendarYearEffectErrors

Examples

```
rm(list=ls())
library(BALD)
data(CumulativeAutoBodilyInjuryTriangle)
CumulativeAutoBodilyInjuryTriangle <- as.matrix(CumulativeAutoBodilyInjuryTriangle)
sample.col <- (dim(CumulativeAutoBodilyInjuryTriangle)[2] - 6:0)
data(HPCE)
HPCE <- as.matrix(HPCE)[,1]
HPCE.rate <- HPCE[-1] / HPCE[-length(HPCE)] - 1
print(HPCE.rate[(-10):0 + length(HPCE.rate)])
HPCE.years <- as.integer(names(HPCE.rate))
max.exp.year <- max(as.integer(
dimnames(CumulativeAutoBodilyInjuryTriangle)[[1]]))
years.to.keep <- HPCE.years <=  max.exp.year + 3
HPCE.rate <- HPCE.rate[years.to.keep]
break.model.input.w.ar1 <- makeBreakAnnualInput(
cumulative.payments = CumulativeAutoBodilyInjuryTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = HPCE.rate,
first.year.in.new.regime = c(1986, 1987),
prior.for.first.year.in.new.regime=c(2,1),
exp.year.type = 'ay',
extra.dev.years = 5,
use.skew.t = TRUE,
bound.for.skewness.parameter=5,
use.ar1.in.calendar.year = TRUE)
## Not run:
break.model.output.w.ar1 <- runLossDevModel(
break.model.input.w.ar1,
burnIn=30.0E+3,
sampleSize=30.0E+3,
```

```
thin=10)
calendarYearEffectAutoregressiveParameter(break.model.output.w.ar1)

## End(Not run)
```

`autoregressiveParameter,AnnualAggLossDevModelOutput-method` *A method to plot and/or return the posterior of the autoregressive parameter for models in BALD.*

Description

A method to plot and/or return the posterior of the autoregressive parameter for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the autoregressive parameter.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with select quantiles of the posterior for the autoregressive parameter. Returned invisibly.

See Also

`autoregressiveParameter` `standardDeviationOfCalendarYearEffect` `calendarYearEffect` `calendarYearEffectErrors`

`BALD` *A package for robust stochastic loss development.*

Description

A package for robust stochastic loss development.

Details

BALD makes available a Bayesian time series model of loss development, estimated using MCMC. This package is intended used for Property and Casualty (PC) actuaries. PC actuaries aggregate historical loss experience in a triangle two dimensions form. The two dimensions are accident years and development years. Most of the time, the loss experience is paid loss and/or incurred loss. Incurred loss is the sum of paid loss and case reserve put up for the estimate of future payment of the claims.

Accident Years is on the row dimension. It is the year when the incident of the claims happened.

Development Year is on the column dimension. It is the time when the claims payments progresses.

Calendar Year is on the diagonal of the triangle form. It is by the calendar year view of loss experience and often times the insurance companies reporting balance sheet and income statement views.

The features of this package include skewed Student-t distribution with time-varying scale parameter for the likelihood of the loss distribution with given parameters. User can put expert prior for the calendar year effect, and structural break in the consumption path of services/development years. This is an update for the older package lossDev as it has been stopped supported

Please read the vignette for guidance on usage. And "Robust Loss Development Using MCMC" by Schmid, Frank A. for theory.

References

Schmid, Frank A., "Robust Loss Development Using MCMC,", 2009

`BreakAnnualAggLossDevModelInput-class`

The final input class for models with a break.

Description

The final input class for models with a break.

Details

`BreakAnnualAggLossDevModelInput` is the final input class for models with a break.

See Also

`LossDevModelInput` `StandardAnnualAggLossDevModelInput`

BreakAnnualAggLossDevModelOutput-class

The final output class for all standard aggregate annual models.

Description

The final output class for all standard aggregate annual models.

Details

`StandardAnnualAggLossDevModelOutput` is the final output class for all standard aggregate annual model objects. Currenly only the slot "input" is allowed to be a non-model node. All other nodes should be the exact name of some settable node in the model. This is because `getModelOutputNodes` currently looks the slot names to determine what values to set; only slot "input" is known to be a slot other than a settable node. This class is derived from `AnnualAggLossDevModelOutput`

See Also

`AnnualAggLossDevModelOutput`

BreakAnnualAggLossDevModelOutputWithZeros-class

The class to handle incremental payments of zero for the break model.

Description

The class to handle incremental payments of zero for the break model.

Details

`BreakAnnualAggLossDevModelOutputWithZeros` is a special class designed to be merged with aggregate annual model objects. It adds one extra node `prob.of.non.zero.payment` to the list of slots.

See Also

`LossDevModelOutput` `AnnualAggLossDevModelOutputWithZeros`

`calculateProbOfPayment` *A function to calculate an empirical vector of the probability of payment.*

Description

A function to calculate an empirical vector of the probability of payment. Intended for internal use only.

Usage

```
calculateProbOfPayment(x)
```

Arguments

`x`	The matrix of the form returned by `getPaymentNoPaymentMatrix`.

Value

A vector equal in length to the number of columns in x representing the empirical probably of payment.

`calendarYearEffect` *A generic function to plot and/or return the predicted and forecast calendar year effects for models in BALD.*

Description

A generic function to plot and/or return the predicted and forecast calendar year effects for models in **BALD**.

Arguments

`object`	The object from which to plot and/or return the calendar year effect.
`restrictedSize`	A logical value. If `TRUE`, the plotted calendar year effect is restricted to the square of dimension equal to the observed triangle with which the model was estimated.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

The calendar year effect is comprised of two components: 1) a prior expected value that may be unique to every cell (subject to weights and bounds) and 2) a diagonal-specific error term. This function plots and returns the factor resulting from the combined effect of these two, which includes an autoregressive component if the model is estimated with such a feature.

The first cell is NA. Values in the first column represent the rate of inflation/escalation to the corresponding cell from the cell in the same column but previous row. Values in the 2nd column and beyond represent the rate of inflation/escalation to the corresponding cell from the cell in the same row but previous column. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`calendarYearEffect("AnnualAggLossDevModelOutput")` `calendarYearEffectErrors` `autoregressiveParameter` `standardDeviationOfCalendarYearEffect` `calendarYearEffectErrorTracePlot`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
calendarYearEffect(standard.model.output)
```

```
## End(Not run)
```

`calendarYearEffect,AnnualAggLossDevModelOutput-method`
A method to plot and/or return predicted and forecast calendar year effects for models in BALD.

Description

A method to plot and/or return predicted and forecast calendar year effects for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the calendar year effect.
`restrictedSize`	A logical value. If `TRUE`, the plotted calendar year effect is restricted to the square of dimension equal to the observed triangle with which the model was estimated.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

The first cell is `NA`. Values in the first column represent the rate of inflation/escalation to the corresponding cell from the cell in the same column but previous row. Values in the 2nd column and beyond represent the rate of inflation/escalation to the corresponding cell from the cell in the same row but previous column.

Value

Mainly called for the side effect of plotting. Also returns a named array with the median predicted values (not on the log scale). Returned invisibly.

See Also

`calendarYearEffect` `calendarYearEffectErrors` `autoregressiveParameter` `standardDeviationOfCalendarYearEf`
`calendarYearEffectErrorTracePlot`

`calendarYearEffectAutoregressiveParameter`
A generic function to plot and/or return the posterior of the autoregressive parameter for the calendar year effect for models in BALD.

Description

A generic function to plot and/or return the posterior of the autoregressive parameter for the calendar year effect for models in **BALD**. See `vignette('BALD')`.

Arguments

`object`	The object from which to plot and/or return the autoregressive parameter which is associated with the calendar year effect.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Value

Mainly called for the side effect of plotting.

See Also

`calendarYearEffectAutoregressiveParameter("AnnualAggLossDevModelOutput")` `standardDeviationOfCalendar`
`calendarYearEffect` `calendarYearEffectErrors`

Examples

```
rm(list=ls())
library(BALD)
data(CumulativeAutoBodilyInjuryTriangle)
CumulativeAutoBodilyInjuryTriangle <- as.matrix(CumulativeAutoBodilyInjuryTriangle)
sample.col <- (dim(CumulativeAutoBodilyInjuryTriangle)[2] - 6:0)
print(decumulate(CumulativeAutoBodilyInjuryTriangle)[1:7, sample.col])
data(HPCE)
HPCE <- as.matrix(HPCE)[,1]
HPCE.rate <- HPCE[-1] / HPCE[-length(HPCE)] - 1
print(HPCE.rate[(-10):0 + length(HPCE.rate)])
HPCE.years <- as.integer(names(HPCE.rate))
max.exp.year <- max(as.integer(dimnames(CumulativeAutoBodilyInjuryTriangle)[[1]]))
years.to.keep <- HPCE.years <=  max.exp.year + 3
HPCE.rate <- HPCE.rate[years.to.keep]
break.model.input.w.ar1 <- makeBreakAnnualInput(
cumulative.payments = CumulativeAutoBodilyInjuryTriangle,
```

```
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = HPCE.rate,
first.year.in.new.regime = c(1986, 1987),
prior.for.first.year.in.new.regime=c(2,1),
exp.year.type = 'ay',
extra.dev.years = 5,
use.skew.t = TRUE,
bound.for.skewness.parameter=5,
use.ar1.in.calendar.year = TRUE)
## Not run:
break.model.output.w.ar1 <- runLossDevModel(
break.model.input.w.ar1,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
calendarYearEffectAutoregressiveParameter(break.model.output.w.ar1)

## End(Not run)
```

`calendarYearEffectAutoregressiveParameter,AnnualAggLossDevModelOutput-method`

A method to plot and/or return the posterior of the autoregressive parameter for the calendar year effect for models in BALD.

Description

A method to plot and/or return the posterior of the autoregressive parameter for the calendar year effect for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the autoregressive parameter which is associated with the calendar year effect.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with select quantiles of the posterior for the autoregressive parameter. Returned invisibly.

See Also

calendarYearEffectAutoregressiveParameter standardDeviationOfCalendarYearEffect calendarYearEffect calendarYearEffectErrors

---

`calendarYearEffectErrors`

A generic function to plot and/or return predicted and forecast calendar year effect errors for models in BALD.

Description

A generic function to plot and/or return predicted and forecast calendar year effect errors for models in **BALD**.

Arguments

`object`	The object from which to plot and/or return the calendar year effect errors.
`extraYears`	An integer expressing the (maximum) number of years to plot (beyond the final observed calendar year). Must be greater than or equal to zero. Default is 15.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

The calendar year effect is comprised of two components: 1) a prior expected value which may be unique to every cell (subject to weights and bounds) and 2) a diagonal-specific error term. This function only plots and returns the error term, which includes an autoregressive component if the model is estimated with such a feature. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

calendarYearEffectErrors("AnnualAggLossDevModelOutput") calendarYearEffect autoregressiveParameter standardDeviationOfCalendarYearEffect calendarYearEffectErrorTracePlot

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
```

```
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
calendarYearEffectErrors(standard.model.output)

## End(Not run)
```

`calendarYearEffectErrors,AnnualAggLossDevModelOutput-method`

A method to plot and/or return predicted and forecast calendar year effect errors for models in BALD.

Description

A method to plot and/or return predicted and forecast calendar year effect errors for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the calendar year effect errors.
`extraYears`	An integer expressing the (maximum) number of years to plot (beyond the final observed calendar year). Must greater than or equal to zero.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with the median predicted errors (not on the log scale). Returned invisibly.

See Also

calendarYearEffectErrors calendarYearEffect autoregressiveParameter standardDeviationOfCalendarYearEf
calendarYearEffectErrorTracePlot

---

calendarYearEffectErrorTracePlot

A generic function to generate the trace plots for select calendar year effect errors.

Description

A generic function to generate the trace plots for select calendar year effect errors.

Arguments

object	The object from which to generate the trace plots.
elements	A numeric vector indicating the elements for which to plot the trace. Valid values are 2 through the total number of exposure years(observed and forecast). If NULL, values are selected automatically.

Details

The calendar year effect is comprised of two components: 1) a prior expected value that may be unique to every cell and 2) a diagonal-specific error term. This function generates trace plots for the diagonal specific error terms only. See vignette('BALD').

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

calendarYearEffectErrorTracePlot("AnnualAggLossDevModelOutput") calendarYearEffectErrors

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
```

```
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
calendarYearEffectErrorTracePlot(standard.model.output)

## End(Not run)
```

`calendarYearEffectErrorTracePlot,AnnualAggLossDevModelOutput-method`
A method to generate the trace plots for select calendar year effect errors.

Description

A method to generate the trace plots for select calendar year effect errors.

Arguments

`object`	The object of type `AnnualAggLossDevModelOutput` from which to generate the trace plots.
`elements`	A numeric vector indicating the elements for which to plot the trace. Valid values are 2 through the total number of exposure years. If NULL, values are selected automatically.

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

`calendarYearEffectErrorTracePlot` `calendarYearEffectErrors`

consumptionPath *A generic function to plot and/or return the estimated consumption path vs development year time.*

Description

A generic function to plot and/or return the estimated consumption path vs development year time.

Arguments

object	The object from which to plot and/or return the estimated consumption path.
plot	A logical value. If TRUE, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

At the heart of aggregate loss development models in **BALD** is the consumption path. The consumption path is (on a log scale) the trajectory of incremental payments absent calendar year effects and with exposure normalized to the first row. Note that the measurement error term is (possibly) a skewed t and as such (possibly) has a non zero mean. The consumption path is absent any such shifts due to skewness. This is a generic function that allows for the retrieval and illustration of this consumption path. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting. Also returns the plotted statistics. Returned invisibly.

See Also

consumptionPath("StandardAnnualAggLossDevModelOutput") consumptionPath("BreakAnnualAggLossDevModelOu
consumptionPathTracePlot

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
```

```
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
consumptionPath(standard.model.output)

## End(Not run)
```

`consumptionPath,BreakAnnualAggLossDevModelOutput-method`
A method to plot and/or return the estimated consumption path vs development year time for break models.

Description

A method to plot and/or return the estimated consumption path vs development year time for break models.

Arguments

`object`	The object from which to plot and/or return the estimated consumption paths.
`plot`	A logical value. If `TRUE`, then the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

At the heart of aggregate loss development models in **BALD** is the consumption path. The consumption path is (on a log scale) the trajectory of incremental payments absent any calendar year effects and normalized to the first row. Note that the measurement error term is (possibly) a skewed t and as such (possibly) has a non zero mean. The consumption path is absent any such shifts in due to skewness. This is a method to allow for the retrieval and illustration of this consumption path.

The break model has a two consumption paths: exposure years prior to the structural break follow one path and exposure years after the break follow another. The slope of the consumption path for exposure years prior to the break is used to extend the consuption path for exposure years post break to the end of the triangle.

Because the model is Bayesian, the estimated consumption paths come as distributions; only the medians are plotted and/or returned.

Value

Mainly called for the side effect of plotting. Also returns the plotted statistics. Returned invisibly.

See Also

consumptionPath consumptionPath("StandardAnnualAggLossDevModelOutput")

`consumptionPath,StandardAnnualAggLossDevModelOutput-method`

A method to plot and/or return the estimated consumption path vs development year time for standard models.

Description

A method to plot and/or return the estimated consumption path vs development year time for standard models.

Arguments

`object`	The object from which to plot and/or return the estimated consumption path.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

At the heart of aggregate loss development models in **BALD** is the consumption path. The consumption path is (on a log scale) the trajectory of incremental payments absent any calendar year effects and wisht exposure normalized to the first row. Note that the measurement error term is (possibly) a skewed t and as such (possibly) has a non zero mean. The consumption path is absent any such shifts due to skewness. This is a method that allows for the retrieval and illustration of this consumption path.

The standard model has a common consumption path for all exposure years.

Because the model is Bayesian, the estimated consumption path comes as a distribution; only the median is plotted and/or returned.

Value

Mainly called for the side effect of plotting. Also returns the plotted statistics. Returned invisibly.

See Also

consumptionPath consumptionPath("BreakAnnualAggLossDevModelOutput")

`consumptionPathTracePlot` *A generic function to generate the trace plots for select consumption path values.*

Description

A generic function to generate the trace plots for select consumption path values. See `vignette('BALD')`.

Arguments

`object`	The object from which to generate the trace plots.
`elements`	A numeric vector indicating the elements for which to plot the trace. Valid values are 1 through the number of development years (columns) in the observed triangle. If NULL, values are selected automatically.
`...`	Additional arguments used by methods.

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

`consumptionTracePlot("StandardAnnualAggLossDevModelOutput")` `consumptionPath`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
```

```
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
consumptionPathTracePlot(standard.model.output)

## End(Not run)
```

`consumptionPathTracePlot,BreakAnnualAggLossDevModelOutput-method`
A method to generate the trace plots for select consumption path values.

Description

A method to generate the trace plots for select consumption path values.

Arguments

`object`	The object of type `BreakAnnualAggLossDevModelOutput` from which to generate the trace plots.
`elements`	A numeric vector indicating the elements for which to plot the trace. Valid values are 1 through the number of development years (columns) in the observed triangle. If NULL, values are selected automatically.
`preBreak`	A logical value indicating whether to plot the trace for the pre-break consumption path or the post-break consumption path.
`...`	Additional arguments used by other methods. Not utilized by this method.

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

`consumptionPathTracePlot` `consumptionPath`

`consumptionPathTracePlot,StandardAnnualAggLossDevModelOutput-method`
A method to generate the trace plots for select consumption path values.

Description

A method to generate the trace plots for select consumption path values.

Arguments

`object`	The object of type `StandardAnnualAggLossDevModelOutput` from which to generate the trace plots.
`elements`	A numeric vector indicating the elements for which to plot the trace. Valid values are 1 through the number of development years (columns) in the observed triangle. If NULL, values are selected automatically.
`...`	Additional arguments used by other methods. Not utilized by this method.

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

`consumptionPathTracePlot` `consumptionPath`

`CPI` *Consumer price index (CPI-U) for the United States.*

Description

Consumer price index (CPI-U) for the United States.

Usage

```
CPI
```

References

bls.gov

Not Seasonally Adjusted

Series Id:	CUUR0000SA0
Area:	U.S. city average
Item:	All items
Base Period:	1982-84=100

`cumulate` *A function to cumulate a triangle.*

Description

A function to cumulate a triangle.

Usage

```
cumulate(triangle)
```

Arguments

`triangle`	A matrix of incremental payments. Or the incremental payment triangle. Each cell of the incremental triangle is the payment during that specific year.

Details

PC actuaries aggregate historical loss experience in a triangle two dimensions form. The two dimensions are accident years and development years. Most of the time, the loss experience is paid loss and/or incurred loss. Incurred loss is the sum of paid loss and case reserve put up for the estimate of future payment of the claims. For paid losses triangle, each cell could represent either net payment of the year or paid to date throughout the life of the claims. The purpose of this function is to transform the incremental payment triangle to cumulative triangle.

Value

A matrix resulting from cumulating the input triangle. This is the cumulative triangles. Each cell is the payment-to-date of the claims.

Examples

```
library(BALD)
#load Generial Liability
data(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
#cumulate the triangle
cumulate(IncrementalGeneralLiablityTriangle)
```

CumulativeAutoBodilyInjuryTriangle *An and example of a cumulative loss triangle.*

Description

An and example of a cumulative loss triangle.

Usage

```
CumulativeAutoBodilyInjuryTriangle
```

Details

Rows pertain to accident years; and columns pertain to development years. For this triangle accident years range from 1974 to 1991 and development years range from 1 to 18.

References

Hayne, Roger M. (2003), "Measurement of Reserve Variability," *Casualty Actuarial Society Forum*, Fall 2003, pp. 141-172, http://www.casact.org/pubs/forum/03fforum/03ff141.pdf.

decumulate *A function to decumulate a triangle.*

Description

A function to decumulate a triangle.

Usage

```
decumulate(triangle)
```

Arguments

triangle	A matrix of cumulative payments.

Value

A matrix resulting from decumulating the input triangle.

Examples

```
library(BALD)
#load Generial Liability
data(IncrementalGeneralLiablityTriangle)
#cumulate the incremental triangle
CumulateGeneralLiablityTriangle<-cumulate(IncrementalGeneralLiablityTriangle)
#decumulate the loss triangle
decumulate(CumulateGeneralLiablityTriangle)
```

`degreesOfFreedom` *A generic function to plot and/or return the posterior of the degrees of freedom for the Student-t in models in BALD.*

Description

A generic function to plot and/or return the posterior of the degrees of freedom for the Student-t in models in **BALD**.

Arguments

`object`	The object from which to plot and/or return the degrees of freedom.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Details

When there is zero skew, the degrees of freedom are the degrees of freedom for the non-skewed t. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

References

Kim, Y., and J. McCulloch (2007) "The Skew-Student Distribution with Application to U.S. Stock Market Returns and the Equity Premium," Department of Economics, Ohio State University, October 2007.

See Also

`degreesOfFreedom("AnnualAggLossDevModelOutput")`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
degreesOfFreedom(standard.model.output)

## End(Not run)
```

`degreesOfFreedom,AnnualAggLossDevModelOutput-method`

A method to plot and/or return the posterior of the degrees of freedom for the Student-t in models in BALD.

Description

A method to plot and/or return the posterior of the degrees of freedom for the Student-t in models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the degrees of freedom.

plotDensity	A logical value. If TRUE, the density is plotted. If plotTrace is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.
plotTrace	A logical value. If TRUE, the trace is plotted. If plotDensity is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with some select quantiles of the posterior for the degrees of freedom. Returned invisibly.

See Also

degreesOfFreedom

---

dot-onLoad-lossDev *Intialize the Namespace.*

Description

Intialize the Namespace. Intended for internal use only.

Usage

```
.onLoad(libname, pkgname)
```

Arguments

libname	The library where the R package is installed.
pkgname	The name of the R package.

Details

Currently only sets correct functions for myPkgName and myLibPath and loads the JAGS module.

See Also

.onLoad

`estimate.priors` *A function to estimate priors for the gompertz curve.*

Description

A function to estimate priors for the gompertz curve. Intended for internal use only.

Usage

```
estimate.priors(p)
```

Arguments

`p`	A vector of the form returned by `calculateProbOfPayment`. `NA`s are allowed.

Details

The function uses `nlm` to minimize the squared error.

Value

A vector equal in length to the number of columns in x representing the empirical probably of payment.

`exposureGrowth` *A generic function to plot and/or return the posterior predicted exposure growth (corresponding to eta in the model).*

Description

A generic function to plot and/or return the posterior predicted exposure growth (corresponding to *eta* in the model). See `vignette('BALD')`.

Arguments

`object`	The object from which to plot and/or return the posterior predicted exposure growth.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Value

Mainly called for the side effect of plotting the exposure growth. Also returns a named numeric vector for the median of the posterior for the exposure growth on the real (not log) scale. Returned invisibly.

See Also

exposureGrowth("AnnualAggLossDevModelOutput") exposureGrowthTracePlot

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(
as.integer(dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
exposureGrowthTracePlot(standard.model.output)

## End(Not run)
```

---

exposureGrowth,AnnualAggLossDevModelOutput-method

A method to plot and/or return the posterior predicted exposure growth (corresponding to eta in the model).

Description

A method to plot and/or return the posterior predicted exposure growth (corresponding to *eta* in the model).

Arguments

object	The object from which to plot and/or return the posterior predicted exposure growth.
plot	A logical value. If TRUE, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Value

Mainly called for the side effect of plotting the exposure growth. Also returns a named numeric vector for the median of the posterior for the exposure growth on the real (not log) scale. Returned invisibly.

See Also

exposureGrowth exposureGrowthTracePlot

---

exposureGrowthAutoregressiveParameter

A generic function to plot and/or return the posterior of the autoregressive parameter for the exposure growth for models in BALD.

Description

A generic function to plot and/or return the posterior of the autoregressive parameter for the exposure growth for models in **BALD**. See `vignette('BALD')`.

Arguments

object	The object from which to plot and/or return the autoregressive parameter which is associated with exposure growth.
plotDensity	A logical value. If TRUE, the density is plotted. If plotTrace is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.
plotTrace	A logical value. If TRUE, the trace is plotted. If plotDensity is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.

Value

Mainly called for the side effect of plotting.

See Also

exposureGrowthAutoregressiveParameter("AnnualAggLossDevModelOutput")

Examples

```
rm(list=ls())
library(BALD)
data(CumulativeAutoBodilyInjuryTriangle)
CumulativeAutoBodilyInjuryTriangle <- as.matrix(CumulativeAutoBodilyInjuryTriangle)
sample.col <- (dim(CumulativeAutoBodilyInjuryTriangle)[2] - 6:0)
print(decumulate(CumulativeAutoBodilyInjuryTriangle)[1:7, sample.col])
data(HPCE)
HPCE <- as.matrix(HPCE)[,1]
HPCE.rate <- HPCE[-1] / HPCE[-length(HPCE)] - 1
print(HPCE.rate[(-10):0 + length(HPCE.rate)])
HPCE.years <- as.integer(names(HPCE.rate))
max.exp.year <- max(as.integer(dimnames(CumulativeAutoBodilyInjuryTriangle)[[1]]))
years.to.keep <- HPCE.years <=  max.exp.year + 3
HPCE.rate <- HPCE.rate[years.to.keep]
break.model.input.w.ar1 <- makeBreakAnnualInput(
cumulative.payments = CumulativeAutoBodilyInjuryTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = HPCE.rate,
first.year.in.new.regime = c(1986, 1987),
prior.for.first.year.in.new.regime=c(2,1),
exp.year.type = 'ay',
extra.dev.years = 5,
use.skew.t = TRUE,
bound.for.skewness.parameter=5,
use.ar1.in.exposure.growth = TRUE)
## Not run:
break.model.output.w.ar1 <- runLossDevModel(
break.model.input.w.ar1,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
exposureGrowthAutoregressiveParameter(break.model.output.w.ar1)

## End(Not run)
```

`exposureGrowthAutoregressiveParameter,AnnualAggLossDevModelOutput-method`

A method to plot and/or return the posterior of the autoregressive parameter for the exposure growth for models in BALD.

Description

A method to plot and/or return the posterior of the autoregressive parameter for the exposure growth for models in **BALD**.

Arguments

object	The object of type AnnualAggLossDevModelOuput from which to plot and/or return the autoregressive parameter which is associated with exposure growth.
plotDensity	A logical value. If TRUE, the density is plotted. If plotTrace is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.
plotTrace	A logical value. If TRUE, the trace is plotted. If plotDensity is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with select quantiles of the posterior for the autoregressive parameter. Returned invisibly.

See Also

exposureGrowthAutoregressiveParameter

---

exposureGrowthTracePlot

A generic function to generate the trace plots for select exposure growth rates.

Description

A generic function to generate the trace plots for select exposure growth rates. See vignette('BALD').

Arguments

object	The object from which to generate the trace plots.
elements	A numeric vector indicating the elements for which to plot the trace. Valid values are 2 through the total number of exposure years. If NULL, values are selected automatically.

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

exposureGrowthTracePlot("AnnualAggLossDevModelOutput") exposureGrowth

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
exposureGrowthTracePlot(standard.model.output)

## End(Not run)
```

---

exposureGrowthTracePlot,AnnualAggLossDevModelOutput-method

A method to generate the trace plots for select exposure growth rates.

Description

A method to generate the trace plots for select exposure growth rates.

Arguments

object	The object of type AnnualAggLossDevModelOutput from which to generate the trace plots.
elements	A numeric vector indicating the elements for which to plot the trace. If NULL, values are selected automatically.

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

exposureGrowthTracePlot exposureGrowth

---

finalCumulativeDiff *A generic function to plot and/or return the difference between final actual and predicted cumulative payments.*

Description

A generic function to plot and/or return the difference between final actual and predicted cumulative payments. See `vignette('BALD')`.

Arguments

object	The object from which to plot and/or return the difference.
plot	A logical value. If TRUE, the plot is generated and the statistics are returned; otherwise only the statistics are returned.
expYearRange	Either a range of years (for example c(1995, 2006)) or one of the keywords "all" or "fullyObs".

Value

Mainly called for the side effect of plotting.

See Also

finalCumulativeDiff("AnnualAggLossDevModelOutput")

Examples

```
rm(list=ls())
library(BALD)
options(device.ask.default=FALSE)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
```

```
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
finalCumulativeDiff(standard.model.output)

## End(Not run)
```

`finalCumulativeDiff,AnnualAggLossDevModelOutput-method`

A method to plot and/or return the difference between final actual and predicted cumulative payments.

Description

A method to plot and/or return the difference between final actual and predicted cumulative payments.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the difference between final actual and predicted cumulative payments.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.
`expYearRange`	Either a range of years (for example c(1995, 2006)) or one of the keywords "all" or "fullyObs".

Details

The relative difference (x/y - 1) between the final observed cumulative payment and the corresponding predicted cumulative payment is plotted for each exposure year. The horizontal lines of each box represent (starting from the top) the 90th, 75th, 50th, 20th, and 10th percentiles. Exposure years in which all cumulative payments are `NA` are omitted.

If `expYearRange` is "fullyObs", then only exposure years with a non missing value in the first column will be plotted.

Value

Mainly called for the side effect of plotting the difference between final actual and predicted cumulative payments by exposure year. Also returns a named array for the percentiles in the plot. Returned invisibly.

See Also

finalCumulativeDiff

`finalCumulativeDiff,AnnualAggLossDevModelOutputWithZeros-method`

A method to plot and/or return the difference between final actual and predicted cumulative payments.

Description

A method to plot and/or return the difference between final actual and predicted cumulative payments.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuputWithZeros` from which to plot and/or return the difference between final actual and predicted cumulative payments.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.
`expYearRange`	Either a range of years (for example c(1995, 2006)) or one of the keywords "all" or "fullyObs".

Details

This method accounts for zero payments. By weighting estimated predicted payments by the probably that the payment is greater than zero.

The relative difference (x/y - 1) between the final observed cumulative payment and the corresponding predicted cumulative payment is plotted for each exposure year. The horizontal lines of each box represent (starting from the top) the 90th, 75th, 50th, 20th, and 10th percentiles. Exposure years in which all cumulative payments are `NA` are omitted.

Value

Mainly called for the side effect of plotting the difference between final actual and predicted cumulative payments by exposure year. Also returns a named array for the percentiles in the plot. Returned invisibly.

See Also

accountForZeroPayments finalCumulativeDiff finalCumulativeDiff,AnnualAggLossDevModelOutput-method

`firstYearInNewRegime` *A generic function to plot and/or return the posterior change point.*

Description

A generic function to plot and/or return the posterior change point.

Arguments

`object`	The object from which to plot and/or return the posterior change point estimate.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

When incorporating a structural break, the user has the option of specifying either 1) the first year in which the new regime applies or 2) a (inclusive) range in which the first year in the new regime applies. If the user specifies a range, the actual year is estimated as a model parameter. This function allows for the retrieval/illustration of the posterior for this estimate. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`firstYearInNewRegime("BreakAnnualAggLossDevModelOutput")` `firstYearInNewRegimeTracePlot`

Examples

```
rm(list=ls())
library(BALD)
data(CumulativeAutoBodilyInjuryTriangle)
CumulativeAutoBodilyInjuryTriangle <- as.matrix(CumulativeAutoBodilyInjuryTriangle)
sample.col <- (dim(CumulativeAutoBodilyInjuryTriangle)[2] - 6:0)
print(decumulate(CumulativeAutoBodilyInjuryTriangle)[1:7, sample.col])
data(HPCE)
HPCE <- as.matrix(HPCE)[,1]
HPCE.rate <- HPCE[-1] / HPCE[-length(HPCE)] - 1
print(HPCE.rate[(-10):0 + length(HPCE.rate)])
HPCE.years <- as.integer(names(HPCE.rate))
max.exp.year <- max(as.integer(
dimnames(CumulativeAutoBodilyInjuryTriangle)[[1]]))
years.to.keep <- HPCE.years <=  max.exp.year + 3
HPCE.rate <- HPCE.rate[years.to.keep]
break.model.input <- makeBreakAnnualInput(
cumulative.payments = CumulativeAutoBodilyInjuryTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
```

```
stoch.inflation.rate = HPCE.rate,
first.year.in.new.regime = c(1986, 1987),
prior.for.first.year.in.new.regime=c(2,1),
exp.year.type = 'ay',
extra.dev.years = 5,
use.skew.t = TRUE,
bound.for.skewness.parameter=5)
## Not run:
break.model.output <- runLossDevModel(
break.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
firstYearInNewRegime(break.model.output)

## End(Not run)
```

`firstYearInNewRegime,BreakAnnualAggLossDevModelOutput-method`
A method to plot and/or return the posterior change point.

Description

A method to plot and/or return the posterior change point.

Arguments

`object`	The object from which to plot and/or return the posterior change point estimate.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named numeric vector with names corresponding to the first year in the new regime and with values corresponding to the density. Returned invisibly.

See Also

`firstYearInNewRegime` `firstYearInNewRegimeTracePlot`

`firstYearInNewRegimeTracePlot` *A generic function to generate the trace plot for the posterior change point.*

Description

A generic function to generate the trace plot for the posterior change point. See `vignette('BALD')`.

Arguments

`object`	The object from which to generate the trace plot for the change point estimate.

Value

Only called for the side effect of plotting.

See Also

`firstYearInNewRegimeTracePlot("BreakAnnualAggLossDevModelOutput")` `firstYearInNewRegimeTracePlot`

Examples

```
rm(list=ls())
library(BALD)
data(CumulativeAutoBodilyInjuryTriangle)
CumulativeAutoBodilyInjuryTriangle <- as.matrix(CumulativeAutoBodilyInjuryTriangle)
sample.col <- (dim(CumulativeAutoBodilyInjuryTriangle)[2] - 6:0)
print(decumulate(CumulativeAutoBodilyInjuryTriangle)[1:7, sample.col])
data(HPCE)
HPCE <- as.matrix(HPCE)[,1]
HPCE.rate <- HPCE[-1] / HPCE[-length(HPCE)] - 1
print(HPCE.rate[(-10):0 + length(HPCE.rate)])
HPCE.years <- as.integer(names(HPCE.rate))
max.exp.year <- max(as.integer(
dimnames(CumulativeAutoBodilyInjuryTriangle)[[1]]))
years.to.keep <- HPCE.years <=  max.exp.year + 3
HPCE.rate <- HPCE.rate[years.to.keep]
break.model.input <- makeBreakAnnualInput(
cumulative.payments = CumulativeAutoBodilyInjuryTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = HPCE.rate,
first.year.in.new.regime = c(1986, 1987),
prior.for.first.year.in.new.regime=c(2,1),
exp.year.type = 'ay',
extra.dev.years = 5,
use.skew.t = TRUE,
bound.for.skewness.parameter=5)
```

```
## Not run:
break.model.output <- runLossDevModel(
break.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
firstYearInNewRegimeTracePlot(break.model.output)

## End(Not run)
```

`firstYearInNewRegimeTracePlot,BreakAnnualAggLossDevModelOutput-method` *A method to generate the trace plot for the posterior change point.*

Description

A method to generate the trace plot for the posterior change point.

Arguments

`object`	The object from which to generate the trace plot for the change point estimate.

Value

Only called for the side effect of plotting.

See Also

`firstYearInNewRegimeTracePlot` `firstYearInNewRegime`

`get.color` *A function to get color values.*

Description

A function to get color values. Intended for internal use only.

Usage

```
get.color(i)
```

Arguments

`i`	An integer value of 1 or greater.

Details

Currenly, the main feature is that it never returns yellow. May, in the future, be expanded.

Value

A color value suitable for ploting commands.

---

getExposureYearLabel *A function to return a "nice" character string for the exposure year label in charts.*

Description

A function to return a "nice" character string for the exposure year label in charts.

Usage

```
getExposureYearLabel(object)
```

Arguments

`object`	An object of type `AnnualAggLossDevModelInput`.

Value

A character.

---

getJagsData *A method to retrieve data for a JAGS model.*

Description

A method to retrieve data for a JAGS model. Intended for internal use only.

Arguments

`object`	The object from which to retrieve initial values.

Details

Any classes derived from class `LossDevModelInput` must provide a method to override this generic function. This overriding method should return a named list of data values.

Value

This method returns a named list of data values.

getJagsData,AnnualLossDevModelInput-method

A method to create all the needed JAGS input common to both the standard model and the break model.

Description

A method to create all the needed JAGS input common to both the standard model and the break model. Intended for internal use only.

Arguments

object	An object of type AnnualAggLossDevModelInput from which to collect the needed model input.

Details

There are currenly two types of AnnualAggLossDevModels (break and standard). These models have many data elements in common and the code to create these common elements is placed in this method. The derived classes StandardAnnualAggLossDevModelInput and BreakAnnualAggLossDevModelInput should call this method via NextMethod() and then should append any needed data.

The returned value is a list containing the following elements with the following meanings:

log.inc A square matrix with only the upper right containing non-missing values of the log of the incremental payments.

log.inc.index A 2-column matrix. Each row corresponds to a non-missing value in log.inc. The first column gives the row of the non-missing value, the second the column.

log.inc.index.length The number of rows in log.inc.index.

smooth.tau.h.2.log A 2-valued vector giving the smoothing parameters for the changing (in development time) variance and skewness.

h.same.as.last A vector of 1's and 0's equal in length to the columns in log.inc. A value of one means the variance and skewness in the corresponding column should be the same as in the previous column.

K Dimension of log.inc.

H Number of additional rows to estimate.

L.vec Vector equal in length to the number of total rows (observed plus forecast). Each value is the column number of the final non-zero incremental payment.

delta.p Vector equal in length to the number of total rows (observed plus forecast). First column is (on the log scale) the mean of the final decay rate. Second is standard deviation.

v Matrix with number of rows equal to total rows (observed plus forecast) and number of columns equal to total development years (observed plus forecast) minus the number of observed development years.

ar1 Zero if calendar year effect does not have an ar1 component; 1 if it does.

`rho.prior` The parameters for the beta prior of the autoregressive parameter in calendar year effect. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`ar1.eta` Zero if exposure growth does not have an ar1 component; 1 if it does.

`rho.eta.prior` The parameters for the beta prior of the autoregressive parameter in exposure growth. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`allow.for.skew` Zero tells the model to assume zero skewness, one to use the skewed t.

`precision.for.skewness` The prior precision for the skewness parameter. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`df.for.skewness` The prior df for the skewness parameter. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`bounds.for.skewness` The the skewness parameter is bounted to prevent chains from getting stuck. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`precision.for.eta.mu` The prior precision for `eta.mu`. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`df.k` The parameter for the chi-sqr prior on the degrees of freedom. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`df.bounds` The lower and upper bounds on the degrees of freedom. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`a.ou.prior` The parameters for the beta prior of the autoregressive parameter for the stochastic inflation. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`precision.for.b.ou` The prior precision for the intercept in the stochastic inflation ar1 process. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`sigma.eta.bounds` The lower and upper bounds on the standard deviation of the exposure year growth rate. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`sigma.kappa.bounds` The lower and upper bounds on the standard deviation of the calendar year growth rate. Stored in R rather than in the model so that modifications are accurately reflected in charts for the prior.

`stoch.log.inf` A vector giving the log inflation rate; missing future values are coded as NA.

`w.stoch.pct.inf.` Matrix with number of rows equal to total (observed and forecast) exposure years and number of columns equal to total development years. The portion of dollars inflating.

`non.stoch.log.inf` Matrix with number of rows equal to total (observed and forecast) exposure years and number of columns equal total development years. Non-stochastic inflation rate. Cannot be `NA`.

`w.non.stoch.pct.inf` Matrix with number of rows equal to total (observed and forecast) exposure years and number of columns equal to total development years. The percent of dollars inflating.

`P` Single value. The position in `stoch.log.inf.c` corresponding to the final observed diagonal in `log.inc`.

`stoch.log.inf.lower.bound` Matrix with number of rows equal to total (observed and forecast) exposure years and number of columns equal total development years. Floor for inflation rate.

`stoch.log.inf.upper.bound` Matrix with number of rows equal to total (observed and forecast) exposure years and number of columns equal total development years. Ceiling for inflation rate.

`estimate.a.ou` Single value (zero or one). Should the auto correlation coefficient for the Ornstein-Uhlenbeck process be estimated or should `fixed.a.ou` be used?

`fixed.a.ou` Single value. Possible non-stochastic auto-correlation coefficient for the Ornstein-Uhlenbeck process.

`estimate.b.ou` Single value (zero or one). Should the stochastic rate of inflation have an estimated constant term or should `fixed.b.ou` be used?

`fixed.b.ou` Single value. Possible non-stochastic constant term for stochastic inflation rate.

`stoch.log.inf.known.mu` Single value. Added to the log stochastic inflation rate after the ar1 estimation process.

Value

A named list of the specific model elements. See Details for more information.

`getJagsData,BreakAnnualLossDevModelInput-method`

A method to collect all the needed model input specific to the break model.

Description

A method to collect all the needed model input specific to the break model.

Arguments

`object`	An object of type `BreakAnnualAggLossDevModelInput` from which to collect the needed model input.

Details

There are currently two types of `AnnualAggLossDevModels` (break and standard). These models have many data elements in common. This method appends only the elements specific to the break model onto the list created by a call to `NextMethod()`.

The following elements are appended:

`x.0` Two values. The lower bound for the number of knots. First is for first spline.

`x.r` Two values. The upper bound for the number of knots. First is for first spline.

`break.row` Two integer values given the (inclusive) range of rows in which the first year in the new regime could occur.

`break.row.priors` The parameters for the beta distribution which serves as the prior for the location of the structural break.

`K.trim` The maximum number of columns in the post-break triangle.

`beta.prior` A matrix giving the prior for the location of knots. First column is for the pre-break spline. Second is for the post-break spline.

`mu.number.of.knots.prior` A matrix giving the prior for the mean of the number of knots. First column is for the pre-break spline. Second is for the post-break spline

`number.of.knots.ubound` A vector giving the upper bound for the number of knots. First is for the pre-break spline. Second is for the post-break spline

Value

A named list of the specific model elements. See details for more info.

`getJagsData,StandardAnnualLossDevModelInput-method`

A method to collect all the needed model input specific to the standard model.

Description

A method to collect all the needed model input specific to the standard model. Intended for internal use only.

Arguments

`object`	An object of type `StandardAnnualAggLossDevModelInput` from which to collect the needed model input.

Details

There are currently two types of `AnnualAggLossDevModels` (break and standard). These models have many data elements in common. This method appends only the elements specific to the standard model onto the list created by a call to `NextMethod()`.

The following elements are appended:

`x.0` A single value. The lower bound for the location of knots.

`x.r` A single value. The upper bound for the location of knots.

`beta.prior` A vector giving the prior for the location of knots.

`mu.number.of.knots.prior` The priors for the mean of the number of knots.

`number.of.knots.ubound` The upper bound for the number of knots.

Value

A named list of the specific model elements. See details for more info.

`getJagsInits` *A method to retrieve initial values for a JAGS model.*

Description

A method to retrieve initial values for a JAGS model. Intended for internal use only.

Arguments

`object`	The object from which to retrieve initial values.

Details

All classes derived from class `LossDevModelInput` should override this method. This the overriding method should return a parameterless function that, when evaluated, returns a named list of initial values.

Value

A function. Overriding methods should return a parameterless function that, when evaluated, returns a named list of initial values.

`getJagsInits,AnnualLossDevModelInput-method`
A method to collect all the needed initial values common to both the standard model and the break model.

Description

A method to collect all the needed initial values common to both the standard model and the break model. Intended for internal use only.

Arguments

`object`	An object of type `AnnualAggLossDevModelInput` from which to collect the needed initial values for the model.

Details

There are currenly two types of `AnnualAggLossDevModels` (break and standard). These models have many features in common, and code to create initial values for these common features is placed in this method. The derived classes `StandardAnnualAggLossDevModelInput` and `BreakAnnualAggLossDevModelInput` call this method via `NextMethod()` and then return a new function.

Value

A named list of the specific model elements. See details for more information.

`getJagsInits,BreakAnnualLossDevModelInput-method`

A method to collect all the needed model initial values unique to the break model.

Description

A method to collect all the needed model initial values unique to the break model.

Arguments

`object`	An object of type `BreakAnnualAggLossDevModelInput` from which to collect the needed model initial values.

Details

There are currently two types of `AnnualAggLossDevModels` (break and standard). Code needed to create initial values specific the break model is placed in this method. This method returns a parameterless function which when called first calls the function returned by `NextMethod()` and then to the list returned by that function appends the following initial values.

`R.` The initial values for the spline node. Needed because dspline cannot create initial values.

Value

A named list of the specific model elements. See details for more info.

See Also

`getJagsInits`

`getJagsInits,StandardAnnualLossDevModelInput-method`

A method to collect all the needed initial values unique to the standard model.

Description

A method to collect all the needed initial values unique to the standard model. Intended for internal use only.

Arguments

`object`	An object of type `StandardAnnualAggLossDevModelInput` from which to collect the needed initial values for the model.

Details

There are currently two types of `AnnualAggLossDevModels` (break and standard). Code needed to create initial values specific the standard model is placed in this method. This method returns a parameterless function which, when called, first calls the function returned by `NextMethod()` and then appends the following initial values onto the list returned by that function:

`S.` The initial values for the spline node. Needed because *dspline* cannot create initial values.

Value

A named list of the specific model elements. See details for more information.

See Also

`getJagsInits`

`getModelOutputNodes` *A method to determine which JAGS nodes the output object is expecting.*

Description

A method to determine which JAGS nodes the output object is expecting. Intended for internal use only.

Arguments

`object`	The object from which to get the expected nodes.

Details

This method is used to query the nodes a particular `LossDevModelOutput` is expecting.

Value

A character vector with each element corresponding to the name of an expected node.

See Also

`getModelOutputNodes("LossDevModelOutput")`

`getModelOutputNodes,LossDevModelOutput-method`
A method to determine which JAGS nodes the output object is expecting.

Description

A method to determine which JAGS nodes the output object is expecting. Intended for internal use only.

Arguments

`object`	The object of type `LossDevModelOutput` from which to get the expected nodes.

Details

This method works by examining all the slot names in `object` and returning all of them except for a slot named "input."

Value

A character vector with each element corresponding to the name of an expected node.

See Also

`getModelOutputNodes`

`getPaymentNoPaymentMatrix`
A function to turn a matrix of incremental payments into zero or ones depending upon whether a payment is positive.

Description

A function to turn a matrix of incremental payments into zero or ones depending upon whether a payment is positive. Intended for internal use only.

Usage

```
getPaymentNoPaymentMatrix(object)
```

Arguments

`object`	The matrix of incremental payments.

Details

The conversion rule is as follows. If NA, then NA. Else if greater than zero, then 1. Else if equal to zero, then zero. Else NA.

Value

A matrix of zero or one (or NA) matching the structure of in input matrix.

`getTriDim` *A generic function to get the dimension of the observed triangle.*

Description

A generic function to get the dimension of the observed triangle. Intended for internal use only.

Arguments

`object`	The object from which to get the dim of the observed triangle.

Details

The dimension of a triangle is the number of rows and number of columns.

Value

An integer vector of the dimension of the observed triangle.

`getTriDim,AnnualAggLossDevModelInput-method`
A method to get the dimension of the observed triangle.

Description

A method to get the dimension of the observed triangle. Intended for internal use only.

Arguments

`object`	An object of type `AnnualLossDevModelInput`.

Value

An integer vector of the dimension of the observed triangle.

See Also

`getTriDim`

`gompertz` *The gompertz function.*

Description

The gompertz function. Intended for internal use only.

Usage

```
gompertz(x, scale, fifty.fifty)
```

Arguments

`x`	The value(s) at which to evaluate the gompertz function.
`scale`	The scale parameter should always (or at least for how it is used in BALD) be positive as this indicates an increasing probably of a zero payment.
`fifty.fifty`	The value at which the gompertz function returns 0.5.

Details

This function is used as the probably of observing a zero payment. (Or one minus the probably of observing a positive payment.) (Note that negative payments are assumed to be missing values.)

Value

An object of type `AnnualAggLossDevModelOutputWithZeros` and `AnnualAggLossDevModelOutput`.

`gompertzParameters` *A generic function to plot and/or return the posterior of the parameters for the gompertz curve which describes the probability of payment.*

Description

A generic function to plot and/or return the posterior of the parameters for the gompertz curve which describes the probability of payment.

Arguments

`object`	The object from which to plot and/or return the parameters.
`parameter`	A character describing which parameter to plot. "scale" for the scale parameter. "fifty.fifty" for the point at which the gompertz give a probably of fifty percent.
`plotDensity`	A logical value. If `TRUE`, then the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, then the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Details

The scale parameter describes how steep the curve is. Larger values are steeper. Positive values indicate that the probability of a positive payment should decrease with development time. (The scale is restricted to be positive.)

The fifty.fifty parameter gives the point (in development time) when the gompertz curve gives a probability of fifty percent. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

gompertzParameters,AnnualAggLossDevModelOutputWithZeros-method

Examples

```
rm(list=ls())
library(BALD)
data(CumulativeAutoBodilyInjuryTriangle)
CumulativeAutoBodilyInjuryTriangle <- as.matrix(CumulativeAutoBodilyInjuryTriangle)
sample.col <- (dim(CumulativeAutoBodilyInjuryTriangle)[2] - 6:0)
print(decumulate(CumulativeAutoBodilyInjuryTriangle)[1:7, sample.col])
data(HPCE)
HPCE <- as.matrix(HPCE)[,1]
HPCE.rate <- HPCE[-1] / HPCE[-length(HPCE)] - 1
print(HPCE.rate[(-10):0 + length(HPCE.rate)])
HPCE.years <- as.integer(names(HPCE.rate))
max.exp.year <- max(as.integer(
dimnames(CumulativeAutoBodilyInjuryTriangle)[[1]]))
years.to.keep <- HPCE.years <=  max.exp.year + 3
HPCE.rate <- HPCE.rate[years.to.keep]
break.model.input <- makeBreakAnnualInput(
cumulative.payments = CumulativeAutoBodilyInjuryTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = HPCE.rate,
first.year.in.new.regime = c(1986, 1987),
prior.for.first.year.in.new.regime=c(2,1),
exp.year.type = 'ay',
extra.dev.years = 5,
use.skew.t = TRUE,
bound.for.skewness.parameter=5)
## Not run:
break.model.output <- runLossDevModel(
break.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
break.model.output.w.zeros <- accountForZeroPayments(break.model.output)
gompertzParameters(break.model.output.w.zeros, parameter='scale')
```

```
## End(Not run)
```

`gompertzParameters,AnnualAggLossDevModelOutputWithZeros-method`
A method to plot and/or return the posterior of the parameters for the gompertz curve which describes the probability of payment.

Description

A method to plot and/or return the posterior of the parameters for the gompertz curve which describes the probability of payment.

Arguments

`object`	The object from which to plot and/or return the parameters.
`parameter`	A character describing which parameter to plot. "scale" for the scale parameter. "fifty.fifty" for the point at which the gompertz give a probably of fifty percent.
`plotDensity`	A logical value. If `TRUE`, then the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, then the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Details

The scale parameter describes how steep the curve is. Larger values are steeper. Positive values indicate that the probability of a positive payment should decrease with development time. (The scale is restricted to be positive.)

The fifty.fifty parameter gives the point (in development time) when the gompertz curve gives a probability of fifty percent.

Value

Mainly called for the side effect of plotting.

See Also

`gompertzParameters`

HPCE *Health Care price index for the United States.*

Description

Health Care price index for the United States.

Usage

```
HPCE
```

References

www.bea.gov

Table 2.5.4

Area:	U.S.
Item:	Health
Base Period:	2012=100

IncrementalGeneralLiablityTriangle *An and example of an incremental incurred loss triangle.*

Description

An and example of an incremental incurred loss triangle.

Usage

```
IncrementalGeneralLiablityTriangle
```

Details

Rows pertain to accident years; and columns pertain to development years. For this triangle accident years range from 1981 to 1990 and development years range from 1 to 10.

The triangle contains incurred incrementals of Automatic Facultative business in General Liability (excluding Asbestos & Environmental).

References

Historical Loss Development Study, 1991 edition, published by the Reinsurance Association of America (RAA), page 96. Quoted from, Thomas Mack (1995), "Which Stochastic Model is Underlying the Chain Ladder Method," *Casualty Actuarial Society Forum*, Fall 1995, pp. 229-240, http://www.casact.org/pubs/forum/95fforum/95ff229.pdf.

`lossDevelopmentFactors`

A generic function to plot and/or return a table of predicted age-to-age loss development factors (or link ratios).

Description

A generic function to plot and/or return a table of predicted age-to-age loss development factors (or link ratios).

Arguments

`object`	The object from which to plot and/or return loss development factors.
`cex.text`	The `cex` value supplied to `text`. Adjusts the relative size of text.
`linespace`	Adjusts the spacing between observed and predicted values.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

While the model estimates ultimate losses directly, comparisons of predicted to observed development factors can give the user a better feel for the model's adequacy. Since the model is Bayesian, each development factor comes as a distribution. Only the median, as a point estimate, are plotted/returned.

The age-to-age factors are the ratios of the cumulative paid values at one period to the previous period. Note that the median of products is not the product of medians, and thus it is possible (or rather likely) that age-to-age factors will not line up with age-to-ultimate factors (see `tailFactor`). See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting. Also returns a numeric matrix of plotted statistics.

See Also

`lossDevelopmentFactors("AnnualAggLossDevModelOutput")` `tailFactor`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
lossDevelopmentFactors(standard.model.output)

## End(Not run)
```

---

lossDevelopmentFactors,AnnualAggLossDevModelOutput-method

A method to plot and/or return a table of predicted age-to-age loss development factors (or link ratios).

Description

A method to plot and/or return a table of predicted age-to-age loss development factors (or link ratios).

Arguments

object	The object of type /codeAnnualAggLossDevModelOutput from which to plot and/or return the factors.
cex.text	The cex value supplied to text. Adjusts the relative size of text.

linespace	Adjusts the spacing between observed and predicted values.
plot	A logical value. If TRUE, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Value

Mainly called for the side effect of plotting, but also returns a numeric matrix of plotted statistics.

See Also

lossDevelopmentFactors tailFactor

LossDevModelInput-class

The base input class for all models in BALD.

Description

The base input class for all models in **BALD**.

Details

LossDevModelInput is the base input class for all model objects. Derived classes must contain all needed data to construct input and initials for the JAGS model. These are accessed via the S4 methods "getJagsData" and "getJagsInits."

See Also

LossDevModelOutput

LossDevModelOutput-class

The base output class for all models in BALD.

Description

The base output class for all models in **BALD**.

Details

LossDevModelOutput is the base class for all output model objects. Derived classes should contain all needed output from a JAGS run and the input object associated with that run in the slot "input." Currenly, only the slot "input" is allowed to be a non-model node. All other nodes must be the exact name of some settable node in the model. This is because getModelOutputNodes currently looks at the slot names to determine what values to set; only slot "input" is known to be a slot other than a settable node. Any class adding extra non-model node slots must also override the method getModelOutputNodes.

See Also

AnnualAggLossDevModelInput

---

lossDevOptions *Options for BALD.*

Description

Options for **BALD**.

Usage

```
lossDevOptions(...)
```

Arguments

...	named values to set. If empty, only the current list of option settings is returned.

Details

Currently the only options are keepCodaOnDisk and logsplinePenaltyFunction.

logsplinePenaltyFunction When drawing kernal density plots using the **logspline**, it maybe desirable to specify a penalty to smooth the density (See ?logspline). This value must be a function which takes one paramter (a vector of the sampled data points) and returns one value – the penalty. The default returns the the log of the number of draws.

Value

The current (or altered) list of option settings is returned.

Examples

```
library(BALD)
#define the log of sample size function
logsamplesize <- function(x) {
log(length(x))
}
#assign the log of sample size function as penalty function
lossDevOptions(logsplinePenaltyFunction = logsamplesize)
```

makeBreakAnnualInput *Create an Object of class BreakAnnualAggLossDevModelInput.*

Description

Create an Object of class `BreakAnnualAggLossDevModelInput`.

Usage

```
makeBreakAnnualInput(
incremental.payments=decumulate(cumulative.payments),
first.year.in.new.regime=trunc(median(as.integer(dimnames(incremental.payments)[[1]]))),
prior.for.first.year.in.new.regime=c(2,2),
extra.dev.years=1,
extra.exp.years=1,
non.stoch.inflation.rate=0,
non.stoch.inflation.weight=1,
stoch.inflation.rate=0,
stoch.inflation.weight=1-non.stoch.inflation.weight,
stoch.inflation.lower.bound=-1,
stoch.inflation.upper.bound=Inf,
known.stoch.inflation.mean=NA,
known.stoch.inflation.persistence=NA,
total.dev.years=extra.dev.years+dim(incremental.payments)[2],
total.exp.years=extra.exp.years+dim(incremental.payments)[1],
cumulative.payments=cumulate(incremental.payments),
exp.year.type=c('ambiguous', 'py', 'ay'),
prior.for.knot.locations.pre.break=NA,
prior.for.number.of.knots.pre.break=c(3, 1/7),
prior.for.knot.locations.post.break=NA,
prior.for.number.of.knots.post.break=c(3, 1/7),
use.skew.t=FALSE,
bound.for.skewness.parameter=10,
last.column.with.scale.innovation=dim(incremental.payments)[2],
use.ar1.in.calendar.year=FALSE,
use.ar1.in.exposure.growth=TRUE,
projected.rate.of.decay=NA)
```

Arguments

`incremental.payments`
: A square matrix of incremental payments. Row names should correspond to the exposure year. Only upper-left (including the diagonal) of Triangle may have non-missing values. Lower-right must be `NA`.

`first.year.in.new.regime`
: May be one of two types. 1) A single numeric value. 2) A numeric vector of length 2. See Details.

`prior.for.first.year.in.new.regime`
: A numeric vector of length 2. See Details.

`extra.exp.years`
: A single integer value (`total.exp.years` overrides) greater than or equal to 1 (default is 1) specifying the number of additional exposure years (or rows in the triangle) to project.

`extra.dev.years`
: A single integer value (`total.dev.years` overrides) greater than or equal to 1 (default is 1) specifying the additional number of development years (or columns in the triangle) to project.

`non.stoch.inflation.rate`
: May be one of three types (See *Inflation Rate* in Details). 1) A single numeric value. 2) A vector of numerics (of specific length). 3) A matrix of numerics (of specific dim).

`non.stoch.inflation.weight`
: May be one of three types (See *Inflation Rate* in Details). 1) A single numeric value. 2) A vector of numerics (of specific length). 3) A matrix of numerics (of specific dim).

`stoch.inflation.rate`
: May be one of two types (See *Inflation Rate* in Details). 1) A single numeric value of *zero*. 2) A vector of numerics (of specific length).

`stoch.inflation.weight`
: May be one of three types (See *Inflation Rate* in Details). 1) A single numerical value. 2) A vector of numerics (of specific length). 3) A matrix of numerics (of specific dim).

`stoch.inflation.lower.bound`
: May be one of three types (See *Inflation Rate* in Details). 1) A single numeric value. 2) A vector of numerics (of specific length). 3) A matrix of numerics (of specific dim).

`stoch.inflation.upper.bound`
: May be one of three types (See *Inflation Rate* in Details). 1) A single numeric value. 2) A vector of numerics (of specific length). 3) A matrix of numerics (of specific dim).

`known.stoch.inflation.mean`
: May be one of two types (See *Inflation Rate* in Details). 1) A single numeric value. 2) `NA`.

`known.stoch.inflation.persistence`
: May be one of two types (See *Inflation Rate* in Details). 1) A single numeric value. 2) `NA`.

`total.exp.years`
: A single integer value (overrides `extra.exp.years`) specifying the last exposure year to project. Must be at least the number of rows in `incremental.payments` + 1.

`total.dev.years`
: A single integer value (overrides `extra.dev.years`) specifying the last development year to project. Must be at least the number of columns in `incremental.payments` + 1 .

`cumulative.payments`
: A numeric matrix with the same dim and dimnames as `incremental.payments`. Must be a possible cumulative payment triangle of `incremental.payments`. (See *Cumulative Payments* Section).

`exp.year.type`
: A single character value indicating the type of exposure years: 'ambiguous', 'py', and 'ay' mean 'Exposure Year', 'Policy Year', and 'Accident Year'; respectively.

`prior.for.knot.locations.pre.break`
: A single numeric value of at least 1. The prior for the location of knots is a scaled beta with parameters `c(1,prior.for.knot.locations.pre.break)`.

`prior.for.number.of.knots.pre.break`
: A two element vector giving the paramters for the prior number of knots.

`prior.for.knot.locations.post.break`
: See `prior.for.knot.locations.pre.break`. Large values produce stable consumption paths at high development years.

`prior.for.number.of.knots.post.break`
: A two element vector giving the paramters for the prior number of knots.

`use.skew.t`
: A logical value. If `TRUE` the model assumes the observed and estimated log incremental payments are realizations from a skewed t distribution; if `FALSE` it will assume zero skewness. (See Reference.)

`bound.for.skewness.parameter`
: A positive numerical value representing the symetric boundaries for the skewness parameter. In most cases, the default should be sufficient. Ignored if `use.skew.t=FALSE`.

`last.column.with.scale.innovation`
: A single integer must be at least 1 and at most the number of columns in `incremental.payments`. See *Measurment Error-Second Order Random Walk* in Details.

`use.ar1.in.calendar.year`
: A logical value. The calendar year effect errors may (at the users digression) include an autoregressive process of order 1. `TRUE` turns on the ar1 process, `FALSE` turns it off.

`use.ar1.in.exposure.growth`
: A logical value. The exposure growth errors may (at the users discretion) include an autoregressive process of order 1. `TRUE` (the Default) turns on the ar1 process, `FALSE` turns it off.

`projected.rate.of.decay`
: May be one of three types (See *Projected Rate of Decay* in Details). 1) `NA`. 2) A matrix of numerics (of specific dim). 3) A named list.

Details

The loss development models require a lot of input. Much of the input is directly dependent on the values of other input. As such, this function facilitates much of the work of setting model parameters and determining which output to collect. See `vignette('BALD')`.

The function creates an object of class `BreakAnnualAggLossDevModelInput`.

Many arguments and much functionality is described in `makeStandardAnnualInput`.

`first.year.in.new.regime` The break model allows for a structural break along the exposure year axis in the consumption path. The slope of the pre-break consumption path is used to extend the post-break consumption path. The time when the break occurs can either be specified with 100 To specify *the* first exposure year in which a new consumption path applies, the user should supply a single value to the parameter `first.year.in.new.regime`. To have the model estimate the point in time when the break occurs, the user should supply a range (min and max) for the possible values to `first.year.in.new.regime`. Note also that the number of rows above and below the break must be at least 4 and as such a triangle of size 8 is the smallest triangle which can be estimated with a break.

`prior.for.first.year.in.new.regime` The prior for the `first.year.in.new.regime` is a (scaled and discretized) beta distribtion. A value of `c(1,1)` would indicate a uniform distribution. See `firstYearInNewRegime`.

`prior.for.knot.locations.pre.break` **and** `prior.for.knot.locations.post.break` If these values are `NA` (the default), then `prior.for.knot.locations.pre.break` will be assigned a value of 2. And `prior.for.knot.locations.post.break` wil be assigned a value of `1 + (num.years.in.post.break.period + 0.5 * num.years.in.break.period)/num.years.in.triangle.`. These values must either both be `NA` or both be set to numbers.

Value

An object of class `AggModelInput`. This the model specified by the returned object must then be estimated using the function `runLossDevModel`.

References

Kim, Y., and J. McCulloch (2007) "The Skew-Student Distribution with Application to U.S. Stock Market Returns and the Equity Premium," Department of Economics, Ohio State University, October 2007

Examples

```
rm(list=ls())
library(BALD)
data(CumulativeAutoBodilyInjuryTriangle)
CumulativeAutoBodilyInjuryTriangle <- as.matrix(CumulativeAutoBodilyInjuryTriangle)
sample.col <- (dim(CumulativeAutoBodilyInjuryTriangle)[2] - 6:0)
print(decumulate(CumulativeAutoBodilyInjuryTriangle)[1:7, sample.col])
data(HPCE)
HPCE <- as.matrix(HPCE)[,1]
HPCE.rate <- HPCE[-1] / HPCE[-length(HPCE)] - 1
print(HPCE.rate[(-10):0 + length(HPCE.rate)])
HPCE.years <- as.integer(names(HPCE.rate))
max.exp.year <- max(as.integer(
dimnames(CumulativeAutoBodilyInjuryTriangle)[[1]]))
years.to.keep <- HPCE.years <=  max.exp.year + 3
HPCE.rate <- HPCE.rate[years.to.keep]
break.model.input <- makeBreakAnnualInput(
cumulative.payments = CumulativeAutoBodilyInjuryTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
```

```
stoch.inflation.rate = HPCE.rate,
first.year.in.new.regime = c(1986, 1987),
prior.for.first.year.in.new.regime=c(2,1),
exp.year.type = 'ay',
extra.dev.years = 5,
use.skew.t = TRUE,
bound.for.skewness.parameter=5)
```

`makeStandardAnnualInput` *Create an Object of class StandardAnnualAggLossDevModelInput.*

Description

Create an Object of class `StandardAnnualAggLossDevModelInput`.

Usage

```
makeStandardAnnualInput(
incremental.payments=decumulate(cumulative.payments),
extra.dev.years=1,
extra.exp.years=1,
non.stoch.inflation.rate=0,
non.stoch.inflation.weight=1,
stoch.inflation.rate=0,
stoch.inflation.weight=1-non.stoch.inflation.weight,
stoch.inflation.lower.bound=-1,
stoch.inflation.upper.bound=Inf,
known.stoch.inflation.mean=NA,
known.stoch.inflation.persistence=NA,
total.dev.years=extra.dev.years+dim(incremental.payments)[2],
total.exp.years=extra.exp.years+dim(incremental.payments)[1],
cumulative.payments=cumulate(incremental.payments),
exp.year.type=c('ambiguous', 'py', 'ay'),
prior.for.knot.locations=2,
prior.for.number.of.knots=c(3, 1/7),
use.skew.t=FALSE,
bound.for.skewness.parameter=10,
last.column.with.scale.innovation=dim(incremental.payments)[2],
use.ar1.in.calendar.year=FALSE,
use.ar1.in.exposure.growth=TRUE,
projected.rate.of.decay=NA)
```

Arguments

`incremental.payments`

A square matrix of incremental payments. Row names must correspond to the exposure year. Only the upper-left (including the diagonal) of this matrix may have non-missing values. Lower-right must be `NA`.

`extra.exp.years`
: A single integer value (`total.exp.years` overrides) greater than or equal to 1 (default is 1) specifying the number of additional exposure years (or rows in the triangle) to project forward.

`extra.dev.years`
: A single integer value (`total.dev.years` overrides) greater than or equal to 1 (default is 1) specifying the additional number of development years (or columns in the triangle) to project forward.

`non.stoch.inflation.rate`
: May be one of three types (See *Inflation Rate* in Details): 1) A single numeric value; 2) a vector of numerics (of specific length); 3) a matrix of numerics (of specific dim).

`non.stoch.inflation.weight`
: May be one of three types (See *Inflation Rate* in Details): 1) A single numeric value; 2) a vector of numerics (of specific length); 3) a matrix of numerics (of specific dim).

`stoch.inflation.rate`
: May be one of two types (See *Inflation Rate* in Details): 1) A single numeric value of *zero*; 2) a vector of numerics (of specific length).

`stoch.inflation.weight`
: May be one of three types (See *Inflation Rate* in Details): 1) A single numerical value; 2) a vector of numerics (of specific length); 3) a matrix of numerics (of specific dim).

`stoch.inflation.lower.bound`
: May be one of three types (See *Inflation Rate* in Details): 1) A single numeric value; 2) a vector of numerics (of specific length); 3) a matrix of numerics (of specific dim).

`stoch.inflation.upper.bound`
: May be one of three types (See *Inflation Rate* in Details): 1) A single numeric value; 2) a vector of numerics (of specific length); 3) a matrix of numerics (of specific dim).

`known.stoch.inflation.mean`
: May be one of two types (See *Inflation Rate* in Details): 1) A single numeric value; 2) `NA`.

`known.stoch.inflation.persistence`
: May be one of two types (See *Inflation Rate* in Details): 1) A single numeric value; 2) `NA`.

`total.exp.years`
: A single integer value (overrides `extra.exp.years`) specifying the last exposure year to project forward. Must be at least the number of rows in `incremental.payments` + 1.

`total.dev.years`
: A single integer value (overrides `extra.dev.years`) specifying the last development year to project forward. Must be at least the number of columns in `incremental.payments` + 1 .

`cumulative.payments`
: A numeric matrix with the same dim and dimnames as `incremental.payments`. Must be a possible cumulative payment triangle of `incremental.payments`. (See *Cumulative Payments* Section.)

`exp.year.type`
: A single character value indicating the type of exposure years: 'ambiguous', 'py', and 'ay' mean 'Exposure Year', 'Policy Year', and 'Accident Year'; respectively.

`prior.for.knot.locations`
: A single numeric value of at least 1. The prior for the location of knots is a scaled beta with parameters `c(1,prior.for.knot.locations)`. Large values produce stable consumption paths at high development years.

`prior.for.number.of.knots`
: A two element vector giving the paramters for the prior number of knots. (See *Number of Knots* in Details)

`use.skew.t`
: A logical value. If `TRUE`, the model assumes that the observed and estimated log incremental payments are realizations from a skewed t distribution; if `FALSE` it assumes zero skewness. (See Reference.)

`bound.for.skewness.parameter`
: A positive numerical value representing the symetric boundaries for the skewness parameter. In most cases, the default should be sufficient. Ignored if `use.skew.t=FALSE`.

`last.column.with.scale.innovation`
: A single integer. Must be at least 1 and at most the number of columns in `incremental.payments`. See *Measurment Error-Second Order Random Walk* in Details.

`use.ar1.in.calendar.year`
: A logical value. The calendar year effect errors may (at the users discretion) include an autoregressive process of order 1. `TRUE` turns on the ar1 process, `FALSE` (the Default) turns it off.

`use.ar1.in.exposure.growth`
: A logical value. The exposure growth errors may (at the users discretion) include an autoregressive process of order 1. `TRUE` (the Default) turns on the ar1 process, `FALSE` turns it off.

`projected.rate.of.decay`
: May be one of three types (See *Projected Rate of Decay* in Details): 1) `NA`; 2) a matrix of numerics (of specific dim); 3) a named list.

Details

The loss development models require extensive input. Much of the input is directly dependent on the values of other input. As such, this function facilitates much of the work of setting model parameters and determining which output to collect.

The function creates an object of class `StandardAnnualAggLossDevModelInput`.

Inflation Rate Workers Compensation indemnity statutes can be extremely complex. In order to provide a sufficient level of flexibility, the model allows for a combination for 3 inflation/escalation rates: one non-stochastic rate, one stochastic rate, and one zero rate. All inflation rates are on the real scale, that is, a 5 percent increase is entered as "0.05." Inflation rates

are only used to create an expected inflation rate. Actual calendar year effects deviate from this prior along the diagional.

Non-Stochastic Rate of Inflation/Escalation The user must specify a non-stochastic rate of inflation for all observed and future time periods. Assuming a non-stochastic inflation rate of zero turns this feature off.

`non.stoch.inflation.rate` The non-stochastic inflation rate is specified by the parameter `non.stoch.inflation.rate`. The default value is set to "0." There are three possible ways to specify `non.stoch.inflation.rate`:

If specified as a single value The inflation in every cell is assumed to be this value. For instance, choosing zero results in no non-stochastic inflation rate.

If specified as a vector Its length must be two less than `total.exp.years + total.dev.years` and must have `names` starting at `dimnames(incremental.payments) [[1]][2]` and increasing by one. The first value corresponds to the inflation rate from the first to second diagonal (or cell 1,1 to cells 1,2 and 2,1). The second value corresponds to the inflation rate from the second to third diagonal. etc.

If specified as a matrix It must have number of rows equal to `total.exp.years` and number of columns equal to `total.dev.years`. The row names must be equal to the row names of `incremental.payments`. Each cell represents the inflation to that cell. Cells in the first column represent inflation down rows. Cells in other columns represent inflation from left to right. Cell 1,1 is ignored.

`non.stoch.inflation.weight` The user has the option of allowing the percentage of dollars that inflate/escalate at the non-stochastic rate to vary. This is done by properly setting the value of `non.stoch.inflation.weight`. The default value is set to "1." `non.stoch.inflation.weight` must be at least zero and at most one. Also `non.stoch.inflation.weight + stoch.inflation.weight` must be at most one. There are three possible ways to specify `non.stoch.inflation.weight`:

If specified as a single value The proportion of dollars that inflate at the rate of `non.stoch.inflatoin.rate` in every cell is assumed to be this value. For instance, choosing zero results in the model assuming that no non-stochastic inflation applies.

If specified as a vector Its length must be `total.dev.years`. This vector is repeated to produce a matrix such that each row equals this vector. This matrix is then assumed to be the supplied for this parameter (See below). See next item.

If specified as a matrix It must have number of rows equal to `total.exp.years` and number of columns equal to `total.dev.years`. The row names must be equal to the row names of `incremental.payments`. Each cell represents the proportion of dollars in the previous cell inflating at the `non.stoch.inflation.rate` rate to that cell. Cells in the first column represent inflation down rows. Cells in other columns represent inflation from left to right. Cell 1,1 is ignored.

Stochastic Rate of Inflation/Escalation The stochastic inflation rate is modeled as an AR1 process (on the log scale). The user must supply a stochastic rate of inflation for at least all observed diagonals and has the option of supplying values for diagonals beyond the final diagonal in the observed triangle. As such, the stochasticity of the stochastic rate of inflation only applies to future rates of inflation. Values not supplied for future diagonals are predicted using the estimated parameters for the supplied inflation series. If the user does not wish to supply/assume a stochastic rate of inflation, he may set the stochastic inflation rate to zero for all periods (`stoch.inflation.rate=0`).

`stoch.inflation.rate` The stochastic inflation rate is specified by the parameter `stoch.inflation.rate`. The default value is set to "0." There are two possible ways to specify `stoch.inflation.rate`:

If specified as a single 0 (zero) It is assumed that no stochastic inflation applies.

If specified as a vector Its length must be at least one less than the number of rows in `incremental.payments`. Future values are simulated as needed. (No element in `stoch.inflation.rate` may be `NA`; only supply values for known rates of inflation.) The vector must be named and the names must be consecutive integers. The names must contain the second and last values of `dimnames(incremental.payments)[[1]]`. The names represent the calendar year of inflation.

`stoch.inflation.weight` The default value is set to "`1 - non.stoch.inflation.weight`". See `non.stoch.inflation.weight`.

`stoch.inflation.lower.bound` The user has the option of putting a lower bound on the stochastic rate of inflation/escalation. `stoch.inflation.rate` should not account for `stoch.inflation.lower.bound` as this would result in incorrectly simulated future inflation rates. The model properly applies the `stoch.inflation.lower.bound` to both observed and future rates of inflation prior to "inflating" dollars in the triangle. The user accounts for lower bounds by properly setting the value of `stoch.inflation.lower.bound`. `stoch.inflation.lower.bound` should be on the real scale (`x[i] / x[i-1] - 1`). The default value is set to `-1`, which is no bound. Bounds are applied prior to weights. There are three possible ways to specify `stoch.inflation.weight`:

If specified as a single value All stochastic rates of inflation in the triangle are bound by this value.

If specified as a vector Its length must be `total.dev.years`. This vector is repeated to produce a matrix with each rows equal to it. It will then be as if this matrix was supplied for this parameter. See next item.

If specified as a matrix It must have number of rows equal to `total.exp.years` and number of columns equal to `total.dev.years`. The row names should be equal to the row names of `incremental.payments`. Each cell represents the bound on the stochastic rate of inflation from the previous cell to that cell. Cells in the first column represent inflation down rows. Cells in other columns represent inflation from left to right. Cell 1,1 is ignored.

`stoch.inflation.upper.bound` Default value is `Inf`, which is no bound. See `stoch.inflation.lower.bound`.

`known.stoch.inflation.mean` The AR1 process used to simulate future stochastic rates of inflation has a non-zero mean which is (at least approximately) equal to the historical mean. If for some reason (i.e., the series of inflation rates is too short) the user believes this historical mean poorly represents future rates of inflation, the user can override the estimation process. If `known.stoch.inflation.mean` is set to `NA`, the mean is estimated. Otherwise the mean is assumed (with certainty) to be the specified value. (Note that since the estimation takes place on the log scale, the logarithm of `known.stoch.inflation.mean` plus 1 is used as the mean on the log scale; this results in `known.stoch.inflation.mean` being the geometric mean.

`known.stoch.inflation.persistence` The user has the option of overriding the estimation of the rho coefficient in the AR1 process used to simulate future stochastic rates of inflation. If `known.stoch.inflation.persistence` is set to `NA`, the rho coefficient is estimated; otherwise it is taken as this value.

Projected Rate of Decay The model estimates an inflation-free rate of decay up to the size of the triangle (`dim(incremental.payments)[2]`). Since no data is available to the model beyond

this point, the model must be supplied with an assumption for future values. This assumption is supplied via the parameter `projected.rate.of.decay` in one of two ways:

A single value of `NA` The inflation-free rate of decay for the final development year in the triangle (`dim(incremental.payments)[2]`) is used for all future development years through development year `total.dev.years`.

A named list This is currently a rather low level interface allowing for much flexibility at the expense of some ease of usability. (In the (possibly near) future, it is likely that the burden placed on the user will be eased.) The named list must contain three elements (with an optional fourth element):

`last.non.zero.payment` The named element `last.non.zero.payment` refers to the last development year in which a non-zero payment may occur. It is assumed that all subsequent incremental payments are exactly zero. Must be at most `total.dev.years` and at least `dim(incrementals)[2]+2`. `last.non.zero.payment` must be specified as a named numeric vector of length `total.exp.years`. Element one then refers to the first exposure year; and element two to the second. The names must start with `dimnames(incrementals)[[1]][1]` and increment by 1. `last.non.zero.payment` can be omitted. If omitted, then the `last.non.zero.payment` for each exposure year is `total.dev.years`.

`final.rate.of.decay.mean` The parameter `final.rate.of.decay.mean` refers to the rate of decay in `last.non.zero.payment`. As the rate of decay is estimated on the log scale, `final.rate.of.decay.mean` is first increased by 1 and then the log of the resulting value is taken; the result is then treated as the mean on the log scale. This results in `final.rate.of.decay.mean` being the geometric mean. (`final.rate.of.decay.mean=0.05` means that the mean of the final rate of decay (on the log scale) is `log(1.05)`.) This may be supplied as a single numeric value, in which case all exposure years are assumed to have the same `final.rate.of.decay.mean`. Or this may be supplied as a named numeric vector of length `total.exp.years`. Element one then refers to the first exposure year, and element two refers to the second, etc. The names of the vector must start with `dimnames(incrementals)[[1]][1]` and increment by `1`. (A Technical Note: `last.non.zero.payment` actually defines the final rate of decay since the rate of decay from it to the next development year is `-1.00`. Thus a better name for `final.rate.of.decay.mean` does not refer to the *final* rate of decay but rather the *penultimate* rate of decay.)

`final.rate.of.decay.sd` The parameter `final.rate.of.decay.sd` refers to the standard deviation around `final.rate.of.decay.mean`. This may be supplied as a single numeric value, in which case all exposure years are assumed to have the same `final.rate.of.decay.sd`. Or this may be supplied as a named numeric vector of length `total.exp.years`. Element one then refers to the first exposure year, and element two refers to the second, etc. The names of the vector must start with `dimnames(incrementals)[[1]][1]` and increment by `1`.

`final.rate.of.decay.weight` The basic concept behind a user-supplied rate of decay is that of a weighted average between the final estimated rate of decay (the one associated with `dim(incrementals)[2]`) and the final supplied value `final.rate.of.decay.mean`. Thus the user must supply a vector of weights to describe the path from the estimated value to the projected value. The "weight" is the weight (on the log scale) of the user-supplied `final.rate.of.decay`. So, a value of `1` means that the projected rate of decay should take on `final.rate.of.decay` and a value of `0` means that the projected rate of decay should equal the final estimated rate of decay. Note

that while these weights are allowed to be outside the interval [0,1], convention should dictate they remain within this interval. These weights are supplied by specifying a value for final.rate.of.decay.weight, which may be specified in two ways. It may be specified as a numeric matrix with number of columns equal to total.dev.years - dim(incremental.payments)[2] and number of rows equal to total.exp.years, in which case the first row corresponds to the first exposure year in incremental.payments and the first column corresponds to the development year immediately following the last one in incremental.payments. It may be specified as a numeric vector of length total.dev.years - dim(incremental.payments)[2], in which case it is assumed to be the same for all exposure years.

Cumulative Triangle As a safeguard, the cumulative and incremental triangles must match up. They are tested by taking the cumulative triangle and then decumulating it with the function decumulate. The decumulated triangle is then compared to the incremental triangle. Except for NA's in either triangle, the values in the triangles must match. (Errors due to numerical precision are accounted for.)

Measurment Error Second-Order Random Walk The model allows for changes in the scale of the measurement error with development time. Changes in the scale parameter (see scaleParameter) are assumed to follow a second-order random walk on the log scale. The model allows for imposed stationarity in the scale parameter by setting of the value last.column.with.scale.innovation. last.column.with.scale.innovation must be coercible to an integer, be least 1, and at most the number of columns in incremental.payments. A value of 1 constrains the scale parameter in each column to the same value. A value of dim(incremental.payments)[2] allows for all columns to have their own scale parameter (but smoothed by the second-order random walk). The effective value used by the model for this argument is the minimum of the supplied value and the last column in incremental.payments with a non-missing value (on the log scale). Note that since the scale parameter is assumed to follow a second-order random walk, a value of 2 results in an (effectively) unconstrained estimation for the scale parameter in the first column.

Number of Knots The consumption path is modeled (on the log scale) as a linear spline. The number of knots in the spline is endogenous to the model estimation. This allows the model to adapt to loss triangles of varying complexity. In order to allow the user to give more credibility to models of a lower complexity (and possibly avoid overfitting), a truncated negative binomial prior is placed on the number of knots. The truncation point is adjusted with the size of the triangle. The paramters for this negative binomial can be specified with the argument prior.for.number.of.knots, which should be a vector of two elements. The first element should be any realy number greater than zero. If this number is an integer, it can be interpreted as the number of failures until the experiment is stopped. The second parameter should be a real number greater than 0 but less than 1 and represents the success probability. The mean of the (un-truncated) negative binomial is then prior.for.number.of.knots[1] * prior.for.number.of.knots[2]/
And the variance is prior.for.number.of.knots[1] * prior.for.number.of.knots[2]/ (1 - prior.for.numb

Value

An object of class AggModelInput. The model specified by the returned object must then be estimated using the function runLossDevModel.

References

Kim, Y., and J. McCulloch (2007) "The Skew-Student Distribution with Application to U.S. Stock Market Returns and the Equity Premium," Department of Economics, Ohio State University, October 2007

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
print(PCE.rate[(-10):0 + PCE.rate.length])
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
```

---

mcmcACF *A generic function to plot autocorrelations found in the MCMC samples for select parameters.*

Description

A generic function to plot autocorrelations found in the MCMC samples for select parameters.

Arguments

object	The object from which to plot autocorrelations.

Details

Chains with high autocorrelation require a longer burnin and more samples to fully explore the parameter space. See `vignette('BALD')`.

Value

Called for the side effect of plotting.

See Also

mcmcACF("StandardAnnualAggLossDevModelOutput") mcmcACF("BreakAnnualAggLossDevModelOutput")

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
mcmcACF(standard.model.output)

## End(Not run)
```

---

mcmcACF,BreakAnnualAggLossDevModelOutput-method

A method to plot autocorrelations found in the MCMC samples for select parameters.

Description

A method to plot autocorrelations found in the MCMC samples for select parameters.

Arguments

`object`	The object from which to plot autocorrelations.

Details

Chains with high autocorrelation require a longer burnin and more samples to fully explore the parameter space.

Value

Called for the side effect of plotting.

TODO

Add option to plot other values. Currently only plots "First Rate Of Decay" (for the pre and post break), "First Calendar Year Effect Error," and "First Exposure Year Growth,"

See Also

`mcmcACF` `mcmcACF("StandardAnnualAggLossDevModelOutput")`

`mcmcACF,StandardAnnualAggLossDevModelOutput-method`

A method to plot autocorrelations found in the MCMC samples for select parameters.

Description

A method to plot autocorrelations found in the MCMC samples for select parameters.

Arguments

`object`	The object from which to plot autocorrelations.

Details

Chains with high autocorrelation require a longer burnin and more samples to fully explore the parameter space.

Value

Called for the side effect of plotting.

TODO

Add option to plot other values. Currently only plots "First Rate Of Decay," "First Calendar Year Effect Error," and "First Exposure Year Growth."

See Also

mcmcACF mcmcACF("BreakAnnualAggLossDevModelOutput")

MCPI *Medical Care price index for the United States.*

Description

Medical Care price index for the United States.

Usage

```
MCPI
```

References

bls.gov

Not Seasonally Adjusted

Series Id:	CUUR0000SAM
Area:	U.S. city average
Item:	Medical care
Base Period:	1982-84=100

---

meanExposureGrowth *A generic function to plot and/or return the posterior of the mean exposure growth for models in BALD.*

Description

A generic function to plot and/or return the posterior of the mean exposure growth for models in **BALD**.

Arguments

object	The object from which to plot and/or return the mean exposure growth.
plotDensity	A logical value. If TRUE, the density is plotted. If plotTrace is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.
plotTrace	A logical value. If TRUE, the trace is plotted. If plotDensity is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.

Details

(Optionally) exposure growth is modeled as an ar1 process. This inherently assumes that periods of high exposure growth are (or at least have the possibility of being) followed by continued high periods. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`meanExposureGrowth("AnnualAggLossDevModelOutput")`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
meanExposureGrowth(standard.model.output)

## End(Not run)
```

meanExposureGrowth,AnnualAggLossDevModelOutput-method

A method to plot and/or return the posterior of the mean exposure growth for models in BALD.

Description

A method to plot and/or return the posterior of the mean exposure growth for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the mean exposure growth.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with select quantiles of the posterior for the mean exposure growth. Returned invisibly.

See Also

`meanExposureGrowth`

myLibPath

Installation Library of the Package.

Description

Installation Library of the Package. Intended for internal use only.

Value

The installation library path. Set by `.onLoad`.

`myPkgName` *Current Name of the Package.*

Description

Current Name of the Package. Intended for internal use only.

Details

Set by `.onLoad`.

Value

The current name of the package including version number if the package was installed as such. (i.e. 'BALD')

`newNodeOutput` *A method to construct new object of type NodeOutput.*

Description

A method to construct new object of type `NodeOutput`. Intended for internal use only.

Usage

```
newNodeOutput(mcarray)
```

Arguments

`mcarray`	An S3 object of type `mcarray`.

Details

This method will return a valid NodeOutput object.

Value

An object of class `NodeOutput`.

See Also

`NodeOutput`

NodeOutput-class *A class to hold JAGS output.*

Description

A class to hold JAGS output. This class is only used internally. No user-level function should return a class of this type.

Details

`NodeOutput` is a wrapper class for `mcarray`. It is used to provide easy access to summary statistics. Current slots are:

`get.value.end` An environment containing a parameterless function called `get.value` which when called will return the mcarray for the node. It also contains `value.name` which is the name of the key (or file on the disk) if the value is stored on the disk.

`mean` An array that is the marginalized mean of the value returned by calling `get.value`.

`median` An array that is the marginalized median of the value returned by calling `get.value`.

`sd` An array that is the marginalized standard deviation of the value returned by calling `get.value`.

See Also

`newNodeOutput`.

numberOfKnots *A generic function to plot and/or return the posterior number of knots.*

Description

A generic function to plot and/or return the posterior number of knots.

Arguments

`object`	The object from which to plot the number of knots.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

The consumption path (or calendar year effect and exposure growth adjusted log incremental payments) is modeled as a linear spline. The number of knots (or places where the spline changes slope) in this spline is endogenous to the model and estimated by way of Reversible Jump Markov Chain Monte Carlo simulation. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting. Also returns statics on the number of knots. Returned invisibly.

See Also

consumptionPath numberOfKnots("StandardAnnualAggLossDevModelOutput") numberOfKnots("BreakAnnualAggLos

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
numberOfKnots(standard.model.output,10)

## End(Not run)
```

---

numberOfKnots,BreakAnnualAggLossDevModelOutput-method

A method to plot and/or return the posterior number of knots.

Description

A method to plot and/or return the posterior number of knots.

Arguments

object	The object from which to plot the number of knots.
plot	A logical value. If TRUE, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

The break model has to consumption paths. This method will plot the number of knots for each path.

Value

Mainly called for the side effect of plotting. Also returns a list of length 2 with each element containing a named vector with names equal to the number of knots and values equal to the density. Returned invisibly.

See Also

consumptionPath numberOfKnots numberOfKnots("StandardAnnualAggLossDevModelOutput")

---

numberOfKnots,StandardAnnualAggLossDevModelOutput-method

A method to plot and/or return the posterior number of knots.

Description

A method to plot and/or return the posterior number of knots.

Arguments

object	The object from which to plot the number of knots.
plot	A logical value. If TRUE, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named vector with names equal to the number of knots and values equal to the density. Returned invisibly.

See Also

consumptionPath numberOfKnots numberOfKnots("BreakAnnualAggLossDevModelOutput")

PCE *Personal Consumption Expenditure (PCE) for the United States.*

Description

Personal Consumption Expenditure (PCE) for the United States.

Usage

```
PCE
```

References

www.bea.gov

table 2.5.4

Area:	U.S.
Item:	All items
Base Period:	2012=100

plot.density.and.or.trace
A rather generic function to plot diagnostics for a single node (a one-dimensional node or a single slot from a multi-dimensional node).

Description

A rather generic function to plot diagnostics for a single node (a one-dimensional node or a single slot from a multi-dimensional node). Intended for internal use only.

Usage

```
plot.density.and.or.trace( coda,
plotDensity,
plotTrace,
d.prior,
nice.parameter.name,
zero.line=FALSE,
lower.bound=NA,
upper.bound=NA,
draw.prior=TRUE)
```

Arguments

coda	The code for the node. Rows are iterations. Columns are chains.
plotDensity	A logical value. If TRUE, the density is plotted. If plotTrace is also TRUE, then two plots are generated. If they are both FALSE, only the statistics are returned.
plotTrace	A logical value. If TRUE, the trace is plotted. If plotDensity is also TRUE, then two plots are generated. If they are both FALSE, only the statistics are returned.
d.prior	A function that takes an array of values and returns the prior density evaluated at those values.
nice.parameter.name	A character value to use as labels in plots.
zero.line	A logical value. Should a verical zero line be drawn on the density plot?
lower.bound	Can be missing, used by density: (from).
upper.bound	Can be missing, used by density: (to).
draw.prior	Should the prior be drawn?

Value

Mainly called for the side effect of plotting. Also returns a vector of quantiles.

---

plot.top.bottom *A function to plot a top and bottom graph on the same chart.*

Description

A function to plot a top and bottom graph on the same chart. Intended for internal use only.

Usage

```
plot.top.bottom(f.top, f.bottom, top.scale=0.95, bottom.scale=0.1)
```

Arguments

f.top	The function to call for ploting the top graph.
f.bottom	The function to call for ploting the bottom graph.
top.scale	A number between zero and 1 indicating the bottom of the top graph.
bottom.scale	A number between zero and 1 indicating the top of the bottom graph.

Details

Main use is to plot the legend of a graph in the "bottom."

Value

NULL invisibly. This function is called for its side effects.

`plot.top.middle.bottom` *A function to plot a top, middle, and bottom graph on the same chart.*

Description

A function to plot a top, middle, and bottom graph on the same chart. Intended for internal use only.

Usage

```
plot.top.middle.bottom(f.top, f.middle, f.bottom, top.scale=0.525, middle.top=0.525,
    middle.scale=0.1, bottom.scale=0.525)
```

Arguments

`f.top`	The function to call for plotting the top graph.
`f.middle`	The function to call for plotting the middle graph.
`f.bottom`	The function to call for plotting the bottom graph.
`top.scale`	A number between zero and 1 indicating the bottom of the top graph.
`middle.top`	A number between zero and 1 indicating the top of the middle graph.
`middle.scale`	A numer between zero and 1 indicating the size of the middle graph.
`bottom.scale`	A number between zero and 1 indicating the top of the bottom graph.

Value

NULL invisibly. This function is called for its side effects.

`plot.trace.plots` *A rather generic function to plot (multiple) trace plots in one call on one graph.*

Description

A rather generic function to plot (multiple) trace plots in one call on one graph. Intended for internal use only.

Usage

```
plot.trace.plots(coda, names)
```

Arguments

coda	The coda for the node(s): first dimension indicates the node; second is iterations; third is chains.
names	A character vector equal in length to the first dim of coda representing the names of the nodes (these are used to label the trace plots).

Details

Plots a trace plot for each of the first dimensions in coda.

Value

NULL invisibly. Only called for the side effect of plotting.

---

predictedPayments *A generic function to plot predicted vs actual payments for models from the BALD package.*

Description

A generic function to plot predicted vs actual payments for models from the **BALD** package.

Arguments

object	The object from which to plot predicted vs actual payments and from which to return predicted payments.
type	A single character value specifying whether to plot/return the predicted incremental or cumulative payments. Valid values are "incremental" or "cumulative." See details as to why these may not match up.
logScale	A logical value. If TRUE, then values are plotted on a log scale.
mergePredictedWithObserved	A logical value. See details.
plotObservedValues	A logical value. If FALSE, then only the predicted values are plotted.
plotPredictedOnlyWhereObserved	A logical value. If TRUE, then only the predicted incremental payments with valid corresponding observed (log) incremental payment are plotted. Ignored for type="cumulative".
quantiles	A vector of quantiles for the predicted payments to return. Useful for constructing credible intervals.
plot	A logical value. If TRUE, then the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

Because the model is Bayesian, each estimated payment comes as a distribution. The median of this distribution is used as a point estimate when plotting and/or returning values. Note: One cannot calculate the estimated incremental payments from the estimated cumulative payments (and vice versa) since the median of sums need not be equal to the sum of medians.

If `mergePredictedWithObserved=TRUE` and `type="incremental"`, then any observed incremental payment will be used in place of its corresponding incremental payment. If `mergePredictedWithObserved=TRUE` and `type="cumulative"`, then only predicted incremental payments (by row) to the right of the last observed cumulative value will enter the calculation. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`predictedPayments("AnnualAggLossDevModelOutput")`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
predictedPayments(standard.model.output)
```

```
## End(Not run)
```

---

predictedPayments,AnnualAggLossDevModelOutput-method
A method to plot predicted vs actual payments for models from the BALD package.

Description

A method to plot predicted vs actual payments for models from the **BALD** package.

Arguments

object	The object of type AnnualAggLossDevModelOutput from which to plot predicted vs actual payments and return predicted payments.
type	A singe character value specifying whether to plot/return the predicted incremental or cumulative payments. Valid values are "incremental" or "cumulative." See details as to why these may not match up.
logScale	A logical value. If TRUE, then values are plotted on a log scale.
mergePredictedWithObserved	A logical value. If TRUE, then the returned values treat observed incremental payments at "face value"; otherwise predicted values are used in place of observed values.
plotObservedValues	A logical value. If FALSE, then only the predicted values are plotted.
plotPredictedOnlyWhereObserved	A logical value. See details.
quantiles	A vector of quantiles for the predicted payments to return. Usefull for constructing credible intervals.
plot	A logical value. If TRUE, then the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

Because the model is Bayesian, each estimated payment comes as a distribution. The median of this distribution is used as a point estimate when plotting and/or returning values. Note: One cannot calculate the estimated incremental payments from the estimated cumulative payments (and vice versa) since the median of sums need not be equal to the sum of medians.

If mergePredictedWithObserved=TRUE and type="incremental", then any observed incremental payment will be used in place of its corresponding incremental payment. If mergePredictedWithObserved=TRUE and type="cumulative", then only predicted incremental payments (by row) to the right of the last observed cumulative value will enter the calculation.

Value

Mainly called for the side effect of plotting. Also returns a named array (with the same structure as the input triangle) containing the predicted log incremental payments. Returned invisibly.

See Also

predictedPayments

`predictedPayments,AnnualAggLossDevModelOutputWithZeros-method`

A method to plot predicted vs actual payments for models from the BALD package.

Description

A method to plot predicted vs actual payments for models from the **BALD** package.

Arguments

`object`	The object of type `AnnualAggLossDevModelOutputWithZeros` from which to plot predicted vs actual payments and return predicted payments.
`type`	A singe character value specifying whether to plot/return the predicted incremental or cumulative payments. Valid values are "incremental" or "cumulative." See details as to why these may not match up.
`logScale`	A logical value. If `TRUE`, then values are plotted on a log scale.
`mergePredictedWithObserved`	A logical value. If `TRUE`, then the returned values treat observed incremental payments at "face value"; otherwise predicted values are used in place of observed values.
`plotObservedValues`	A logical value. If `FALSE`, then only the predicted values are plotted.
`plotPredictedOnlyWhereObserved`	A logical value. If `TRUE`, then only the predicted incremental payments with valid corresponding observed (log) incremental payment are plotted. Ignored for `type="cumulative"`.
`quantiles`	A vector of quantiles for the predicted payments to return. Usefull for constructing credible intervals.
`plot`	A logical value. If `TRUE`, then the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

This method accounts for zero payments. By weighting estimated predicted payments by the probably that the payment is greater than zero.

Because the model is Bayesian, each estimated payment comes as a distribution. The median of this distribution is used as a point estimate when plotting and/or returning values. Note: One cannot calculate the estimated incremental payments from the estimated cumulative payments (and vice versa) since the median of sums need not be equal to the sum of medians.

Value

Mainly called for the side effect of plotting. Also returns a named array (with the same structure as the input triangle) containing the predicted log incremental payments. Returned invisibly.

See Also

`accountForZeroPayments` `predictedPayments`

---

probablityOfPayment *A generic function to plot the probability of a payment.*

Description

A generic function to plot the probability of a payment.

Arguments

`object`	The object from which to plot the probability of a payment.
`plot`	A logical value. If `TRUE`, then the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

Because the model is Bayesian, each estimated payment comes as a distribution. The median of this distribution is used as a point estimate when plotting and/or returning values. Note: Negative payments are treated as missing and are not accounted for. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting. Also returns a matrix containing the (median) probably of payment. Returned invisibly.

See Also

`accountForZeroPayments`

Examples

```
rm(list=ls())
library(BALD)
data(CumulativeAutoBodilyInjuryTriangle)
CumulativeAutoBodilyInjuryTriangle <- as.matrix(CumulativeAutoBodilyInjuryTriangle)
sample.col <- (dim(CumulativeAutoBodilyInjuryTriangle)[2] - 6:0)
print(decumulate(CumulativeAutoBodilyInjuryTriangle)[1:7, sample.col])
data(HPCE)
HPCE <- as.matrix(HPCE)[,1]
HPCE.rate <- HPCE[-1] / HPCE[-length(HPCE)] - 1
print(HPCE.rate[(-10):0 + length(HPCE.rate)])
HPCE.years <- as.integer(names(HPCE.rate))
max.exp.year <- max(as.integer(
dimnames(CumulativeAutoBodilyInjuryTriangle)[[1]]))
years.to.keep <- HPCE.years <=  max.exp.year + 3
HPCE.rate <- HPCE.rate[years.to.keep]
break.model.input <- makeBreakAnnualInput(
cumulative.payments = CumulativeAutoBodilyInjuryTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = HPCE.rate,
first.year.in.new.regime = c(1986, 1987),
prior.for.first.year.in.new.regime=c(2,1),
exp.year.type = 'ay',
extra.dev.years = 5,
use.skew.t = TRUE,
bound.for.skewness.parameter=5)
## Not run:
break.model.output <- runLossDevModel(
break.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
break.model.output.w.zeros <- accountForZeroPayments(break.model.output)
probablityOfPayment(break.model.output.w.zeros)

## End(Not run)
```

`probablityOfPayment,AnnualAggLossDevModelOutputWithZeros-method`
A method to plot the probability of a payment.

Description

A method to plot the probability of a payment.

Arguments

`object`	The object from which to plot the probability of a payment.

plot — A logical value. If TRUE, then the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

Because the model is Bayesian, each estimated payment comes as a distribution. The median of this distribution is used as a point estimate when plotting and/or returning values. Note: Negative payments are treated as missing and are not accounted for.

Value

Mainly called for the side effect of plotting. Also returns a matrix containing the (median) probably of payment. Returned invisibly.

See Also

accountForZeroPayments

QQPlot *A generic function to plot a Q-Q plot for models in the BALD package.*

Description

A generic function to plot a Q-Q plot for models in the **BALD** package.

Arguments

object — The object from which to plot the values.

Details

This function plots sorted observed log incremental payments vs sorted predicted log incremental payments. Credible intervals are also plotted. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

QQPlot("AnnualAggLossDevModelOutput") triResi

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
QQPlot(standard.model.output)

## End(Not run)
```

QQPlot,AnnualAggLossDevModelOutput-method

A method to plot a Q-Q plot for models in the BALD package.

Description

A method to plot a Q-Q plot for models in the **BALD** package.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot the values.

Details

This function plots sorted observed log incremental payments vs sorted predicted log incremental payments. Credible intervals are also plotted.

Value

NULL. Called for the side effect of plotting.

See Also

QQPlot triResi

`rateOfDecay` *A generic function to plot and/or return the esimtated rate of decay vs development year time.*

Description

A generic function to plot and/or return the esimtated rate of decay vs development year time.

Arguments

`object`	The object from which to plot and/or return the estimated rate of decay.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

The simplest definition of the rate of decay is the exponentiated first difference of the consumption path. This is a generic function to allow for the retrieval and illustration of the rate of decay. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting. Also returns the plotted statistics. Returned invisibly.

See Also

`rateOfDecay("StandardAnnualAggLossDevModelOutput")` `rateOfDecay("BreakAnnualAggLossDevModelOutput")` `consumptionPath` `rateOfDecayTracePlot`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
rateOfDecay(standard.model.output)

## End(Not run)
```

`rateOfDecay,BreakAnnualAggLossDevModelOutput-method`

A method to plot and/or return the esimtated rate of decay vs development year time for break models.

Description

A method to plot and/or return the esimtated rate of decay vs development year time for break models.

Arguments

`object`	The object from which to plot and/or return the estimated rate of decay.
`plot`	A logical value. If `TRUE`, then the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

The simplest definition of the rate of decay is the exponentiated first difference of the consumption path. The break model has two rates of decay. One which applies to exposure years prior to a structural break. And another which applies after the break. This is a method to allow for the retrieval and illustration of these rates of decay.

Because the model is Bayesian, the estimated rates of decay come as distributions; only the medians are plotted and/or returned.

Value

Mainly called for the side effect of plotting. Also returns the plotted statistics. Returned invisibly.

See Also

`rateOfDecay` `rateOfDecay("StandardAnnualAggLossDevModelOutput")` `consumptionPath`

`rateOfDecay,StandardAnnualAggLossDevModelOutput-method`

A method to plot and/or return the esimtated rate of decay vs development year time for standard models.

Description

A method to plot and/or return the esimtated rate of decay vs development year time for standard models.

Arguments

`object`	The object from which to plot and/or return the estimated rate of decay.
`plot`	A logical value. If `TRUE`, then the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

The simplest definition of the rate of decay is the exponentiated first difference of the consumption path. The standard model has a common rate of decay for all exposure years. This is a method to allow for the retrieval and illustration of the rate of decay.

Because the model is Bayesian, the estimated rate of decay comes as a distribution; only the median is plotted and/or returned.

Value

Mainly called for the side effect of plotting. Also returns the plotted statistics. Returned invisibly.

See Also

`rateOfDecay` `rateOfDecay("BreakAnnualAggLossDevModelOutput")` `consumptionPath`

rateOfDecayTracePlot *A generic function to plot the trace plots for select rate of decay values.*

Description

A generic function to plot the trace plots for select rate of decay values. See `vignette('BALD')`.

Arguments

`object`	The object from which to generate the trace plots.
`elements`	A numeric vector indicating for which elements to plot the trace. Valid values are 2 through the number of columns in the observed triangle. If NULL, values are selected automatically.
`...`	Additional arguments used by methods.

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

`rateOfDecayTracePlot("StandardAnnualAggLossDevModelOutput")` `rateOfDecay`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
```

```
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
rateOfDecayTracePlot(standard.model.output)

## End(Not run)
```

`rateOfDecayTracePlot,BreakAnnualAggLossDevModelOutput-method`
A method to generate the trace plots for select rate of decay values.

Description

A method to generate the trace plots for select rate of decay values.

Arguments

`object`	The object of type `BreakAnnualAggLossDevModelOutput` from which to generate the trace plots.
`elements`	A numeric vector indicating for which elements to plot the trace. Valid values are 2 through the number of columns in the observed triangle. If NULL, values are selected automatically.
`preBreak`	A logical value indicating whether to plot the trace for the pre-break rate(s) of decay or the post-break rate(s) of decay.
`...`	Additional arguments used by other methods. Not utilized by this method.

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

`rateOfDecayTracePlot` `rateOfDecay`

rateOfDecayTracePlot,StandardAnnualAggLossDevModelOutput-method

A method to generate the trace plots for select rate of decay values.

Description

A method to generate the trace plots for select rate of decay values.

Arguments

object	The object of type StandardAnnualAggLossDevModelOutput from which to generate the trace plots.
elements	A numeric vector indicating for which elements to plot the trace. Valid values are 2 through the number of columns in the observed triangle. If NULL, values are selected automatically.
...	Additional arguments used by other methods. Not utilized by this method.

Value

NULL invisibly. Only called for the side effect of plotting.

See Also

rateOfDecayTracePlot rateOfDecay

runLossDevModel

A generic function to run models in BALD.

Description

A generic function to run models in **BALD**. See vignette('BALD'). Overriding methods must return a valid output object.

Arguments

object	The object containing the model to estimate.

Value

object of class LossDevModelOutput.

See Also

runLossDevModel("LossDevModelInput")

Examples

```
rm(list=ls())
library(BALD)
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
print(PCE.rate[(-10):0 + PCE.rate.length])
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
print(PCE.rate[(-10):0 + PCE.rate.length])
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)

## End(Not run)
```

---

runLossDevModel,LossDevModelInput-method

A method to run models in BALD.

Description

A method to run models in **BALD**.

Arguments

object	The object of type LossDevModelInput containing the model to estimate.
burnIn	An integer to represent the number of initial MCMC iterations to be discarded. (The adaptive phase (nAddapt) is not considered part of burnIn.)

sampleSize	An integer to represent the number of MCMC iterations to execute following the burnIn. (Actual number of iterations kept approximately sampleSize / thin.)
thin	Keep every thin'th value of sampleSize.
nChains	The number of MCMC chains to run.
nAddapt	The length of the adaptive phase for the MCMC algorithm. (Default is trunc(burnIn/4)+1.)

Details

This method returns a valid output object or flags an error. This method is suitable for classes properly derived from class LossDevModelInput that properly override "getJagsData" and "getJagsInits" and whose output type has a valid getModelOutputNodes method.

BALD sets the seed in each chain from a random number generated inside of R. So to make a run reproducible, all one must do is set the seed (using set.seed) in R prior to the execution of this method.

Value

An object of class LossDevModelOutput.

See Also

runLossDevModel set.seed

---

scaleParameter *A generic function to plot and/or return the posterior of the scale parameter for the Student-t measurement equation for models in BALD.*

Description

A generic function to plot and/or return the posterior of the scale parameter for the Student-t measurement equation for models in **BALD**.

Arguments

object	The object from which to plot and/or return the scale parameter.
column	The scale parameter is allowed to vary with development time. Setting column results in the plotting and returning of the scale parameter corresponding to that column. Default value is 1.
plotDensity	A logical value. If TRUE, then the density is plotted. If plotTrace is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.
plotTrace	A logical value. If TRUE, then the trace is plotted. If plotDensity is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.

Details

As the degrees of freedom of the t goes to infinity, the scale parameter is the standard deviation of the resulting normal distribution (assuming zero skew). See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`scaleParameter("AnnualAggLossDevModelOutput")`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
scaleParameter(standard.model.output)

## End(Not run)
```

`scaleParameter,AnnualAggLossDevModelOutput-method` *A method to plot and/or return the posterior of the scale parameter for the Student-t measurement equation for models in BALD.*

Description

A method to plot and/or return the posterior of the scale parameter for the Student-t measurement equation for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the scale parameter.
`column`	The scale parameter is allowed to vary with development time. Setting `column` results in the plotting and returning of the scale parameter corresponding to that column. Default value is `1`.
`plotDensity`	A logical value. If `TRUE`, then the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, then the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with select quantiles of the posterior for the scale parameter. Returned invisibly.

See Also

`scaleParameter`

`setGenericVerif` *A Safe Version of setGeneric.*

Description

A Safe Version of `setGeneric`. Intended for internal use only.

Usage

```
setGenericVerif(name, ...)
```

Arguments

name	The character string name of the generic function.
...	Additional arguments to pass to setGeneric.

Details

setGeneric will overwrite existing generic functions. This will result in the loss of all methods already associated with that generic. setGenericVerif only sets the generic if it is not already a generic. If a generic by the name of name already exists, a warning is issued and NULL is returned. Otherwise setGeneric is called and its value returned.

Value

setGenericVerif really exists for its side effect; but returns the value returned by setGeneric or NULL.

See Also

setGeneric

---

skewnessParameter *A generic function to plot and/or return the posterior of the skewness parameter for models in BALD.*

Description

A generic function to plot and/or return the posterior of the skewness parameter for models in **BALD**.

Arguments

object	The object from which to plot and/or return the skewness parameter.
plotDensity	A logical value. If TRUE, then the density is plotted. If plotTrace is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.
plotTrace	A logical value. If TRUE, then the trace is plotted. If plotDensity is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.

Details

The skewness parameter does not directly correspond to the degree of skewness. However, all else being equal, a larger (in magnitude) skewness parameter indicates a higher degree of skewness, and a skewness parameter of zero equates to zero skew. See vignette('BALD').

Value

Mainly called for the side effect of plotting.

References

Kim, Y., and J. McCulloch (2007) "The Skew-Student Distribution with Application to U.S. Stock Market Returns and the Equity Premium," Department of Economics, Ohio State University, October 2007

See Also

skewnessParameter("AnnualAggLossDevModelOutput")

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
skewnessParameter(standard.model.output)

## End(Not run)
```

`skewnessParameter,AnnualAggLossDevModelOutput-method`
A method to plot and/or return the posterior of the skewness parameter for models in BALD.

Description

A method to plot and/or return the posterior of the skewness parameter for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the skewness parameter.
`plotDensity`	A logical value. If `TRUE`, then the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, then the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Details

The skewness parameter does not directly correspond to the degree of skewness. However, all else being equal, a larger (in magnitude) skewness parameter indicates a higher degree of skewness, and a skewness parameter of zero equates to zero skew.

Value

Mainly called for the side effect of plotting. But also returns a named array with some select quantiles of the posterior for the skewness parameter. Returned invisibly.

References

Kim, Y., and J. McCulloch (2007) "The Skew-Student Distribution with Application to U.S. Stock Market Returns and the Equity Premium," Department of Economics, Ohio State University, October 2007

See Also

`skewnessParameter`

---

slot,NodeOutput,character-method

A method to override the behavoir of the function slot.

Description

A method to override the behavoir of the function `slot`.

Arguments

`object`	The object of type `NodeOutput` with the slot to look up.
`name`	A character value giving the name of the slot to look up.

Details

In order to enhance the memory management, coda files are optionally stored on the harddrive in temporary files and loaded on an as needed basis. By overriding this function, we are able to make this seamless. Overriding the function `slot` is a slight abuse and, as such, may in the future be replaced by an accessor function.

Value

Only if name is exactly "value" will the method return the `mcarray` containing the coda. Otherwise, it returns the result of `callNextMethod()`.

See Also

`slot`

StandardAnnualAggLossDevModelInput-class

The final input class for models without a break.

Description

The final input class for models without a break.

Details

`StandardAnnualAggLossDevModelInput` is the final input class for models without a break.

See Also

`LossDevModelInput` `StandardAnnualAggLossDevModelInput`

StandardAnnualAggLossDevModelOutput-class

The final output class for all standard aggregate annual models.

Description

The final output class for all standard aggregate annual models.

Details

`StandardAnnualAggLossDevModelOutput` is the final output class for all standard aggregate annual model objects. Currenly, only the slot "input" is allowed to be a non-model node. All other nodes should be the exact name of some settable node in the model. This is because `getModelOutputNodes` currently looks at the slot names to determine what values to set; only slot "input" is known to be a slot other than a settable node. This class is derived from `AnnualAggLossDevModelOutput`.

See Also

`AnnualAggLossDevModelOutput`

StandardAnnualAggLossDevModelOutputWithZeros-class

The class to handle incremental payments of zero for the standard model.

Description

The class to handle incremental payments of zero for the standard model.

Details

`StandardAnnualAggLossDevModelOutputWithZeros` is a special class designed to be merged with aggregate annual model objects. It adds one extra node `prob.of.non.zero.payment` to the list of slots.

See Also

`LossDevModelOutput` `AnnualAggLossDevModelOutputWithZeros`

`standardDeviationForScaleInnovation`

A generic function to plot and/or return the posterior of the standard deviation for the innovation in the scale parameter for models in BALD.

Description

A generic function to plot and/or return the posterior of the standard deviation for the innovation in the scale parameter for models in **BALD**.

Arguments

`object`	The object from which to plot and/or return the standard deviation for the innovation in the log of the scale parameter.
`plotDensity`	A logical value. If `TRUE`, then the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, then the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.

Details

Changes in the scale parameter (see `scaleParameter`) are assumed to follow a second-order random walk on the log scale. This function plots the posterior standard deviation for this random walk. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`standardDeviationForScaleInnovation("AnnualAggLossDevModelOutput")` `standardDeviationVsDevelopmentTim`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
```

```
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
standardDeviationForScaleInnovation(standard.model.output)

## End(Not run)
```

`standardDeviationForScaleInnovation,AnnualAggLossDevModelOutput-method`

A method to plot and/or return the posterior of the standard deviation for the innovation in the scale parameter for models in BALD.

Description

A method to plot and/or return the posterior of the standard deviation for the innovation in the scale parameter for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the standard deviation for the innovation in the log of the scale parameter.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with some select quantiles of the posterior for the standard deviation in question. Returned invisibly.

See Also

`standardDeviationForScaleInnovation` `scaleParameter` `standardDeviationVsDevelopmentTime`

`standardDeviationOfCalendarYearEffect` *A generic function to plot and/or return the posterior of the standard deviation of the calendar year effect for models in BALD.*

Description

A generic function to plot and/or return the posterior of the standard deviation of the calendar year effect for models in **BALD**. See `vignette('BALD')`.

Arguments

`object`	The object from which to plot and/or return the standard deviation of the calendar year effect.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.

Value

Mainly called for the side effect of plotting.

See Also

`standardDeviationOfCalendarYearEffect("AnnualAggLossDevModelOutput")` `calendarYearEffect` `calendarYearEffectErrors` `autoregressiveParameter`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
```

```
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
standardDeviationOfCalendarYearEffect(standard.model.output)

## End(Not run)
```

`standardDeviationOfCalendarYearEffect,AnnualAggLossDevModelOutput-method`
A method to plot and/or return the posterior of the standard deviation of the calendar year effect for models in BALD.

Description

A method to plot and/or return the posterior of the standard deviation of the calendar year effect for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the standard deviation of the calendar year effect.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with select quantiles of the posterior for the standard deviation of the calendar year effect. Returned invisibly.

See Also

`standardDeviationOfCalendarYearEffect` `calendarYearEffect` `calendarYearEffectErrors` `autoregressiveParameter`

`standardDeviationOfExposureGrowth`

A generic function to plot and/or return the posterior of the standard deviation of the exposure growth rate for models in BALD.

Description

A generic function to plot and/or return the posterior of the standard deviation of the exposure growth rate for models in **BALD**. See `vignette('BALD')`.

Arguments

`object`	The object from which to plot and/or return the standard deviation of the exposure growth rate.
`plotDensity`	A logical value. If `TRUE`, then the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, then the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Value

Mainly called for the side effect of plotting.

See Also

`standardDeviationOfExposureGrowth("AnnualAggLossDevModelOutput")`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
```

```
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
standardDeviationOfExposureGrowth(standard.model.output)

## End(Not run)
```

`standardDeviationOfExposureGrowth,AnnualAggLossDevModelOutput-method`

A method to plot and/or return the posterior of the standard deviation of rhe exposure growth rate for models in BALD.

Description

A method to plot and/or return the posterior of the standard deviation of rhe exposure growth rate for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the standard deviation of the exposure growth rate.
`plotDensity`	A logical value. If `TRUE`, then the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, then the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with select quantiles of the posterior for the standard deviation of exposure growth. Returned invisibly.

See Also

`standardDeviationOfExposureGrowth` `exposureGrowth` `meanExposureGrowth`

`standardDeviationVsDevelopmentTime` *A generic function to plot and/or return the posterior estimated standard deviation by development year.*

Description

A generic function to plot and/or return the posterior estimated standard deviation by development year.

Arguments

`object`	The object from which to plot and/or return the estimated standard deviation by development year.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

Aggregate loss development models in **BALD** allow for changes (by development year) in the measurement error around the log incremental payments. This is a generic function that allows for the retrieval and illustration of this standard deviation. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`standardDeviationVsDevelopmentTime("AnnualAggLossDevModelOutput")`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
```

```
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
standardDeviationVsDevelopmentTime(standard.model.output)

## End(Not run)
```

`standardDeviationVsDevelopmentTime,AnnualAggLossDevModelOutput-method`
A method to plot and/or return the posterior estimated standard deviation by development year.

Description

A method to plot and/or return the posterior estimated standard deviation by development year.

Arguments

`object`	The object of type `AnnualAggLossDevModelOutput` from which to plot and/or return the estimated standard deviation by development year.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

Aggregate loss development models in **BALD** allow for changes (by development year) in the measurement error around the log incremental payments. This is a method that allows for the retrieval and illustration of this standard deviation.

Value

Mainly called for the side effect of plotting. Also returns a numeric vector of the plotted statistics. Returned invisibly.

See Also

`standardDeviationVsDevelopmentTime`

`stochasticInflation` *A generic function to plot and/or return predicted and forecast stochastic inflation rates for models in BALD.*

Description

A generic function to plot and/or return predicted and forecast stochastic inflation rates for models in **BALD**.

Arguments

`object`	The object from which to plot and/or return the stochastic inflation rates.
`extraYears`	An integer expressing the (maximum) number of years to plot (beyond the final observed year). Must be at least zero. Default is 15.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

If the model incorporates a stochastic rate of inflation, then that rate is assumed to follow (on the log scale) an autoregressive process of order 1. (The autoregressive process of order 1 is the discrete equivalent to an Ornstein-Uhlenbeck process.) This function plots the median of the posterior predictive distribution for stochastic inflation (not on the log scale) rates by year. Values are returned prior to the application of any limits or weights. Note that for years where observed values are supplied, the model takes those values at face value. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`stochasticInflation("AnnualAggLossDevModelOutput")` `stochasticInflationRhoParameter` `stochasticInflationStationaryMean`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
```

```
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
stochasticInflation(standard.model.output)

## End(Not run)
```

`stochasticInflation,AnnualAggLossDevModelOutput-method`

A method to plot and/or return predicted and forecast stochastic inflation rates for models in BALD.

Description

A method to plot and/or return predicted and forecast stochastic inflation rates for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return predicted and forecast stochastic inflation rates.
`extraYears`	An integer expressing the (maximum) number of years to plot (beyond the final observed year). Must be at least zero.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array of the median predicted inflation rate (not on the log scale). Returned invisibly.

See Also

`stochasticInflation` `stochasticInflationRhoParameter` `stochasticInflationStationaryMean`

`stochasticInflationRhoParameter` *A generic function to plot and/or return the posterior of the stochastic inflation rho parameter for models in BALD.*

Description

A generic function to plot and/or return the posterior of the stochastic inflation rho parameter for models in **BALD**.

Arguments

`object`	The object from which to plot and/or return the stochastic inflation rho parameter.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.

Details

If the model incorporates a stochastic rate of inflation, then that rate is assumed to follow (on the log scale) an autoregressive process of order 1. (The autoregressive process of order 1 is the discrete equivalent to an Ornstein-Uhlenbeck process.) This function plots the posterior for the rho parameter, assuming one was estimated. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`stochasticInflationRhoParameter("AnnualAggLossDevModelOutput")` `stochasticInflationStationaryMean` `stochasticInflation`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
```

```
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
stochasticInflationRhoParameter(standard.model.output)

## End(Not run)
```

`stochasticInflationRhoParameter,AnnualAggLossDevModelOutput-method`

A method to plot and/or return the posterior of the stochastic inflation rho parameter for models in BALD.

Description

A method to plot and/or return the posterior of the stochastic inflation rho parameter for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the stochastic inflation rho parameter.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with select quantiles of the rho parameter. Returned invisibly.

See Also

`stochasticInflationRhoParameter` `stochasticInflationStationaryMean` `stochasticInflation`

stochasticInflationStationaryMean *A generic function to plot and/or return the posterior of the stochastic inflation stationary mean for models in BALD.*

Description

A generic function to plot and/or return the posterior of the stochastic inflation stationary mean for models in **BALD**.

Arguments

object	The object from which to plot and/or return the stochastic inflation stationary mean.
plotDensity	A logical value. If TRUE, the density is plotted. If plotTrace is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.
plotTrace	A logical value. If TRUE, the trace is plotted. If plotDensity is also TRUE, then two plots are generated. If they are both FALSE, then only the statistics are returned.

Details

If the model incorporates a stochastic rate of inflation, then that rate is assumed to follow (on the log scale) an autoregressive process of order 1. (The autoregressive process of order 1 is the discrete equivalent to an Ornstein-Uhlenbeck process.) This function plots the posterior for the stationary mean (on the log scale), assuming such a mean was estimated. See `vignette('BALD')`.

Value

Mainly called for the side effect of plotting.

See Also

`stochasticInflationStationaryMean("AnnualAggLossDevModelOutput")` `stochasticInflationRhoParameter` `stochasticInflation`

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
```

```
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
stochasticInflationStationaryMean(standard.model.output)

## End(Not run)
```

`stochasticInflationStationaryMean,AnnualAggLossDevModelOutput-method`
A method to plot and/or return the posterior of the stochastic inflation stationary mean for models in BALD.

Description

A method to plot and/or return the posterior of the stochastic inflation stationary mean for models in **BALD**.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the stochastic inflation stationary mean.
`plotDensity`	A logical value. If `TRUE`, the density is plotted. If `plotTrace` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.
`plotTrace`	A logical value. If `TRUE`, the trace is plotted. If `plotDensity` is also `TRUE`, then two plots are generated. If they are both `FALSE`, then only the statistics are returned.

Value

Mainly called for the side effect of plotting. Also returns a named array with select quantiles of the stochastic inflation stationary mean. Returned invisibly.

See Also

stochasticInflationStationaryMean stochasticInflationRhoParameter stochasticInflation

---

`tailFactor` *A generic function to plot and/or return the predicted tail factors for a specific attachment point.*

Description

A generic function to plot and/or return the predicted tail factors for a specific attachment point.

Arguments

`object`	The object from which to plot the predicted tail factors and return tail factors for *all* attachment points.
`attachment`	An integer value specifying the attachment point for the tail. Must be at least 1. See Details for more information.
`useObservedValues`	A logical value. If `TRUE`, observed values are substituted for predicted values whenever possible in the calculation. If `FALSE`, only predicted values are used.
`firstIsHalfReport`	A logical value or `NA`. See Details for more info.
`finalAttachment`	An integer value must be at least 1. Default value is `attachment`. A call to `tailFactor` will return (invisibly) a matrix of tail factors through this value.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.
`expYearRange`	Either a range of years (for example c(1995, 2006)) or one of the keywords "all" or "fullyObs".

Details

The tail factor is the ratio of the estimated ultimate loss to cumulative loss at some point in development time. This is a generic function to allow for the retrieval and illustration the tail factor by exposure year.

Note on `firstIsHalfReport` and `attachment`: `firstIsHalfReport` refers to the first column of the triangle. For policy year triangles, the first column is often referred to as a "half-report", the second column is called "first-report", the third column is called "second-report", etc. If `firstIsHalfReport=TRUE`, then `tailFactor` will assume the triangle is arranged in such a way that the first column is the "half-report" and `attachment=1` indicates that the charted tail factor attaches at the cumulative loss through the second column. If `firstIsHalfReport=FALSE`, then `attachment=1` indicates that the charted tail factor attaches at the cumulative loss through the first column. Since `attachment` must be coercible to an integer, it is impossible to plot half-to-ultimate tail factors; however, they are the first column in the returned matrix.

firstIsHalfReport can be NA (the default) if the exposure year type was specified to be one of "policy year" or "accident year" at the time the input object was constructed (see makeStandardAnnualInput or makeBreakAnnualInput). An exposure year type of "policy year" corresponds to firstIsHalfReport=TRUE, and an exposure year type of "accident year" corresponds to firstIsHalfReport=FALSE. Setting firstIsHalfReport to a non-missing value will override this default.

If expYearRange is "fullyObs", then only exposure years with a non missing value in the first column will be plotted. See vignette('BALD').

Value

Mainly called for the side effect of plotting.

See Also

tailFactor("StandardAnnualAggLossDevModelOutput") tailFactor("BreakAnnualAggLossDevModelOutput")

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
tailFactor(standard.model.output,10)

## End(Not run)
```

`tailFactor,BreakAnnualAggLossDevModelOutput-method`
A method to plot and/or return the predicted tail factors for a specific attachment point.

Description

A method to plot and/or return the predicted tail factors for a specific attachment point.

Arguments

`object`	The object from which to plot the predicted tail factors and return tail factors for *all* attachment points.
`attachment`	An integer value specifying the attachment point for the tail. Must be at least 1. See Details for more info.
`useObservedValues`	A logical value. If `TRUE`, observed values are substituted for predicted values whenever possible in the calculation. If `FALSE`, only predicted values are used.
`firstIsHalfReport`	A logical value or `NA`. See Details for more information.
`finalAttachment`	An integer value must be at least 1 default value is `attachment`. A call to `tailFactor` returns (invisibly) a matrix of tail factors through this value.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.
`expYearRange`	Either a range of years (for example c(1995, 2006)) or one of the keywords "all" or "fullyObs".

Details

The tail factor is the ratio of the estimated ultimate loss to cumulative loss at some point in development time. This is a method to allow for the retrieval and illustration of the tail factor by exposure year.

Because the model is Bayesian, each tail factor comes as a distribution. To ease graphical interpretation, only the median for each factor is plotted/returned. See for more details `tailFactor`.

For comparison purposes, the function returns three separated tail factors for three scenarios. Theses three tail factors are returned as a list with the following names and meanings:

"Actual" These are the tail factors estimated when taking the break into consideration.

"AsIfPostBreak" These are the tail factors estimated when assuming all years where in the post-break regime.

"AsIfPreBreak" These are the tail factors estimated when assuming all years where in the pre-break regime.

Value

Mainly called for the side effect of plotting. Also returns tail factors for *all* attachment points through `finalAttachment`. See Details. Returned invisibly.

See Also

`tailFactor` `tailFactor("StandardAnnualAggLossDevModelOutput")`

`tailFactor,BreakAnnualAggLossDevModelOutputWithZeros-method`

A method to plot and/or return the predicted tail factors for a specific attachment point.

Description

A method to plot and/or return the predicted tail factors for a specific attachment point.

Arguments

`object`	The object from which to plot the predicted tail factors and return tail factors for *all* attachment points.
`attachment`	An integer value specifying the attachment point for the tail. Must be at least 1. See Details for more info.
`useObservedValues`	A logical value. If `TRUE`, observed values are substituted for predicted values whenever possible in the calculation. If `FALSE`, only predicted values are used.
`firstIsHalfReport`	A logical value or `NA`. See Details for more information.
`finalAttachment`	An integer value must be at least 1 default value is `attachment`. A call to `tailFactor` returns (invisibly) a matrix of tail factors through this value.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.
`expYearRange`	Either a range of years (for example c(1995, 2006)) or one of the keywords "all" or "fullyObs".

Details

This method accounts for zero payments. By weighting estimated predicted payments by the probably that the payment is greater than zero.

The tail factor is the ratio of the estimated ultimate loss to cumulative loss at some point in development time. This is a method to allow for the retrieval and illustration of the tail factor by exposure year.

Because the model is Bayesian, each tail factor comes as a distribution. To ease graphical interpretation, only the median for each factor is plotted/returned. See for more details `tailFactor`.

For comparison purposes, the function returns three separated tail factors for three scenarios. Theses three tail factors are returned as a list with the following names and meanings:

"Actual" These are the tail factors estimated when taking the break into consideration.

"AsIfPostBreak" These are the tail factors estimated when assuming all years where in the post-break regime.

"AsIfPreBreak" These are the tail factors estimated when assuming all years where in the pre-break regime.

Value

Mainly called for the side effect of plotting. Also returns tail factors for *all* attachment points through `finalAttachment`. See Details. Returned invisibly.

See Also

`accountForZeroPayments` `tailFactor` `tailFactor("BreakAnnualAggLossDevModelOutput")` `tailFactor("StandardAnnualAggLossDevModelOutputWithZeros")` `tailFactor("StandardAnnualAggLossDevModel`

---

tailFactor,StandardAnnualAggLossDevModelOutput-method

A method to plot and/or return the predicted tail factors for a specific attachment point.

Description

A method to plot and/or return the predicted tail factors for a specific attachment point.

Arguments

`object`	The object from which to plot the predicted tail factors and return tail factors for *all* attachment points.
`attachment`	An integer value specifying the attachment point for the tail. Must be at least 1. See Details for more info.
`useObservedValues`	A logical value. If `TRUE`, observed values are substituted for predicted values whenever possible in the calculation. If `FALSE`, only predicted values are used.
`firstIsHalfReport`	A logical value or `NA`. See Details for more information.
`finalAttachment`	An integer value must be at least 1 default value is `attachment`. A call to `tailFactor` returns (invisibly) a matrix of tail factors through this value.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.
`expYearRange`	Either a range of years (for example c(1995, 2006)) or one of the keywords "all" or "fullyObs".

Details

The tail factor is the ratio of the estimated ultimate loss to cumulative loss at some point in development time. This is a method to allow for the retrieval and illustration of the tail factor by exposure year.

Because the model is Bayesian, each tail factor comes as a distribution. To ease graphical interpretation, only the median for each factor is plotted/returned. See for more details `tailFactor`.

Value

Mainly called for the side effect of plotting. Also returns tail factors for *all* attachment points through `finalAttachment`. Returned invisibly.

See Also

`tailFactor` `tailFactor("BreakAnnualAggLossDevModelOutput")`

`tailFactor,StandardAnnualAggLossDevModelOutputWithZeros-method`

A method to plot and/or return the predicted tail factors for a specific attachment point.

Description

A method to plot and/or return the predicted tail factors for a specific attachment point.

Arguments

`object`	The object from which to plot the predicted tail factors and return tail factors for *all* attachment points.
`attachment`	An integer value specifying the attachment point for the tail. Must be at least 1. See Details for more info.
`useObservedValues`	A logical value. If `TRUE`, observed values are substituted for predicted values whenever possible in the calculation. If `FALSE`, only predicted values are used.
`firstIsHalfReport`	A logical value or `NA`. See Details for more information.
`finalAttachment`	An integer value must be at least 1 default value is `attachment`. A call to `tailFactor` returns (invisibly) a matrix of tail factors through this value.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.
`expYearRange`	Either a range of years (for example c(1995, 2006)) or one of the keywords "all" or "fullyObs".

Details

This method accounts for zero payments. By weighting estimated predicted payments by the probably that the payment is greater than zero.

The tail factor is the ratio of the estimated ultimate loss to cumulative loss at some point in development time. This is a method to allow for the retrieval and illustration of the tail factor by exposure year.

Because the model is Bayesian, each tail factor comes as a distribution. To ease graphical interpretation, only the median for each factor is plotted/returned. See for more details `tailFactor`.

For comparison purposes, the function returns three separated tail factors for three scenarios. Theses three tail factors are returned as a list with the following names and meanings:

"Actual" These are the tail factors estimated when taking the break into consideration.

"AsIfPostBreak" These are the tail factors estimated when assuming all years where in the post-break regime.

"AsIfPreBreak" These are the tail factors estimated when assuming all years where in the pre-break regime.

Value

Mainly called for the side effect of plotting. Also returns tail factors for *all* attachment points through `finalAttachment`. See Details. Returned invisibly.

See Also

`accountForZeroPayments` `tailFactor` `tailFactor("BreakAnnualAggLossDevModelOutput")` `tailFactor("BreakAnnualAggLossDevModelOutputWithZeros")` `tailFactor("StandardAnnualAggLossDevModelOut`

`triResi` *A generic function to plot and/or return residuals for models in the BALD package.*

Description

A generic function to plot and/or return residuals for models in the **BALD** package. See `vignette('BALD')`.

Arguments

`object`	The object from which to plot and/or return the residuals.
`standardize`	A logical value. If `TRUE`, the plotted and returned residuals are normalized to their respective standard deviation.
`timeAxis`	A character value describing along which of the three time axes to plot the residuals: 'dy' for development year time, 'cy' for calendar year time, 'ey' for exposure year time.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Value

Mainly called for the side effect of plotting.

See Also

triResi("AnnualAggLossDevModelOutput") QQPlot

Examples

```
rm(list=ls())
options(device.ask.default=FALSE)
library(BALD)
data(IncrementalGeneralLiablityTriangle)
IncrementalGeneralLiablityTriangle <- as.matrix(IncrementalGeneralLiablityTriangle)
print(IncrementalGeneralLiablityTriangle)
data(PCE)
PCE <- as.matrix(PCE)[,1]
PCE.rate <- PCE[-1] / PCE[-length(PCE)] - 1
PCE.rate.length <- length(PCE.rate)
PCE.years <- as.integer(names(PCE.rate))
years.available <- PCE.years <= max(as.integer(
dimnames(IncrementalGeneralLiablityTriangle)[[1]]))
PCE.rate <- PCE.rate[years.available]
PCE.rate.length <- length(PCE.rate)
standard.model.input <- makeStandardAnnualInput(
incremental.payments = IncrementalGeneralLiablityTriangle,
stoch.inflation.weight = 1,
non.stoch.inflation.weight = 0,
stoch.inflation.rate = PCE.rate,
exp.year.type = 'ay',
extra.dev.years=5,
use.skew.t=TRUE)
## Not run:
standard.model.output <- runLossDevModel(
standard.model.input,
burnIn=30.0E+3,
sampleSize=30.0E+3,
thin=10)
#residule plot by Development Year
triResi(standard.model.output, timeAxis='dy')
#residule plot by exposure year
triResi(standard.model.output, timeAxis='ey')
#residule plot by calendar year
triResi(standard.model.output, timeAxis='cy')

## End(Not run)
```

`triResi,AnnualAggLossDevModelOutput-method`

A method to plot and/or return residuals for models in the BALD package.

Description

A method to plot and/or return residuals for models in the **BALD** package.

Arguments

`object`	The object of type `AnnualAggLossDevModelOuput` from which to plot and/or return the residuals.
`timeAxis`	A character value describing along which of the three (3) time axis to plot the residuals. 'dy' for development year time, 'cy' for calendar year time, 'ey' for exposure year time.
`standardize`	A logical value. If `TRUE`, the plotted and returned residuals are normalized to their respective standard deviation.
`plot`	A logical value. If `TRUE`, the plot is generated and the statistics are returned; otherwise only the statistics are returned.

Details

Because the model is Bayesian, each residual comes as a distribution. To ease graphical interpretation, only the median for each residual is plotted/returned. The residual is defined as the observed value minus the posterior mean; if standardized, it is also divided by its posterior standard deviation.

Value

Mainly called for the side effect of plotting. Also returns a named array with the same structure as the input triangle. Returned invisibly.

See Also

`triResi` `QQPlot`

Index