Package 'ChainLadder'

December 6, 2018

Type Package

Title Statistical Methods and Models for Claims Reserving in General Insurance

Version 0.2.9

Date 2018-12-06

Description Various statistical methods and models which are typically used for the estimation of outstanding claims reserves in general insurance, including those to estimate the claims development result as required under Solvency II.

Imports Matrix, actuar, methods, stats, lattice, grid, tweedie, utils, systemfit, statmod, cplm (>= 0.7-3), ggplot2, MASS

Suggests RUnit, knitr, rmarkdown

VignetteBuilder knitr

License GPL (>= 2)

URL https://github.com/mages/ChainLadder#chainladder

BugReports https://github.com/mages/ChainLadder/issues

LazyLoad yes

LazyData yes

NeedsCompilation no

Author Markus Gesmann [aut, cre],
Daniel Murphy [aut],
Yanwei (Wayne) Zhang [aut],
Alessandro Carrato [aut],
Giuseppe Crupi [ctb],
Christophe Dutang [ctb],
Arnaud Lacoume [ctb],
Arthur Charpentier [ctb],
Mario Wuthrich [aut],
Fabio Concina [aut],
Eric Dal Moro [aut],
Yuriy Krvavych [ctb],
Vincent Goulet [ctb]

Maintainer Markus Gesmann <markus.gesmann@googlemail.com>

Repository CRAN

Date/Publication 2018-12-06 22:40:03 UTC

R topics documented:

ChainLadder-package . . . 3
ABC . . . 4
as.LongTriangle . . . 5
ata . . . 6
auto . . . 7
BootChainLadder . . . 8
CDR . . . 10
chainladder . . . 11
ClarkCapeCod . . . 14
ClarkLDF . . . 18
CLFMdelta . . . 21
coef.ChainLadder . . . 23
Cumulative and incremental triangles . . . 24
GenIns . . . 25
getLatestCumulative . . . 26
glmReserve . . . 27
Join2Fits . . . 31
JoinFitMse . . . 32
liab . . . 32
LRfunction . . . 33
M3IR5 . . . 34
MackChainLadder . . . 34
MCLpaid . . . 39
Mortgage . . . 39
Mse-methods . . . 40
MultiChainLadder . . . 42
MultiChainLadder-class . . . 47
MultiChainLadderFit-class . . . 50
MultiChainLadderMse-class . . . 51
MultiChainLadderSummary-class . . . 52
MunichChainLadder . . . 53
MW2008 . . . 56
MW2014 . . . 56
NullNum-class . . . 57
PaidIncurredChain . . . 57
plot-MultiChainLadder . . . 59
plot.BootChainLadder . . . 60
plot.clark . . . 61
plot.MackChainLadder . . . 62
plot.MunichChainLadder . . . 64
predict.TriangleModel . . . 65

print.ata . . . 66
print.clark . . . 67
qpaid . . . 68
quantile.MackChainLadder . . . 69
QuantileIFRS17 . . . 71
RAA . . . 73
residCov . . . 73
residuals.MackChainLadder . . . 74
summary-methods . . . 75
summary.ata . . . 77
summary.BootChainLadder . . . 78
summary.clark . . . 79
summary.MackChainLadder . . . 80
summary.MunichChainLadder . . . 81
Table65 . . . 82
triangle S3 Methods . . . 83
triangles-class . . . 86
tweedieReserve . . . 88
tweedieReserve methods . . . 92
UKMotor . . . 93
USAA triangle . . . 94
vcov.clark . . . 95

Index **97**

`ChainLadder-package` *Methods and Models for Claims Reserving*

Description

ChainLadder provides methods and models which are typically used in insurance claims reserving.

The package grew out of presentations given at the Stochastic Reserving Seminar at the Institute of Actuaries in 2007 and 2008 and followed by talks at CAS meetings in 2008 and 2010.

More information is available on the project web site `https://github.com/mages/ChainLadder`

For more financial packages see also CRAN Task View 'Emperical Finance' at `https://CRAN.R-project.org/view=Finance`.

Author(s)

Maintainer: Markus Gesmann <markus.gesmann@gmail.com>

References

Thomas Mack. Distribution-free calculation of the standard error of chain ladder reserve estimates. Astin Bulletin. *Vol. 23. No 2. 1993. pp.213:225*

Thomas Mack. The standard error of chain ladder reserve estimates: Recursive calculation and inclusion of a tail factor. Astin Bulletin. *Vol. 29. No 2. 1999. pp.361:366*

Gerhard Quarg and Thomas Mack. Munich Chain Ladder. Blatter DGVFM 26. *Munich. 2004.*

England, PD and Verrall, RJ. Stochastic Claims Reserving in General Insurance (with discussion). British Actuarial Journal 8. *III. 2002*

B. Zehnwirth and G. Barnett. Best Estimates for Reserves. Proceedings of the CAS. *Volume LXXXVII. Number 167.November 2000.*

Clark, David R., "LDF Curve-Fitting and Stochastic Reserving: A Maximum Likelihood Approach," CAS Forum, *Fall 2003.*

*Zhang Y. A general multivariate chain ladder model.*Insurance: Mathematics and Economics, *46, pp. 588:599, 2010.*

Zhang, Y. Likelihood-based and Bayesian Methods for Tweedie Compound Poisson Linear Mixed Models, Statistics and Computing, *forthcoming.*

Bardis, Majidi, Murphy. A Family of Chain-Ladder Factor Models for Selected Link Ratios. Variance. *Pending. Variance 6:2, 2012, pp. 143-160.* https://www.variancejournal.org/issues/06-02/143.pdf

Modelling the claims development result for solvency purposes. Michael Merz, Mario V. Wüthrich. Casualty Actuarial Society E-Forum, Fall 2008.

Claims Run-Off Uncertainty: The Full Picture. Michael Merz, Mario V. Wüthrich. Swiss Finance Institute Research Paper No. 14-69. https://ssrn.com/abstract=2524352. *2014*

Markus Gesmann. Claims Reserving and IBNR. Computational Actuarial Science with R. *Chapman and Hall/CRC. 2014*

Examples

```
## Not run:
  demo(ChainLadder)

## End(Not run)
```

ABC *Run off triangle of accumulated claims data*

Description

Run-off triangle of a worker's compensation portfolio of a large company

Usage

```
data(ABC)
```

Format

A matrix with 11 accident years and 11 development years.

Source

B. Zehnwirth and G. Barnett. Best Estimates for Reserves. Proceedings of the CAS. *Volume LXXXVII. Number 167. November 2000.*

Examples

```
ABC
plot(ABC)
plot(ABC, lattice=TRUE)
```

---

as.LongTriangle *Convert Triangle from wide to long*

Description

Given a Triangle in matrix ("wide") format, convert to data.frame ("long") format.

Usage

```
as.LongTriangle(Triangle, varnames = names(dimnames(Triangle)),
                value.name = "value", na.rm = TRUE)
```

Arguments

`Triangle`	a loss "triangle". Must be a `matrix`.
`varnames`	`character` names for the columns that will store the `rownames` and `colnames` of matrix `Triangle`. Defaults to `names(dimnames(Triangle))` if available. If not provided, uses c("origin", "dev").
`value.name`	column name to be given to the matrix values that will be stored in the data.frame. Defaults to "value".
`na.rm`	should NA values be excluded from the data.frame? Defaults to TRUE.

Details

Unlike the as.data.frame.triangle method, and Unlike the 'melt' method in the 'reshape2' package, this function returns a data.frame where the rownames and colnames of Triangle are stored as *factors*. This can be a critical feature when the order of the levels of the columns is important. For example, when a Triangle is plotted, the order of the origin and dev dimensions is important. See Examples section.

Value

A `data.frame`.

Author(s)

Daniel Murphy

See Also

`as.data.frame.triangle`

Examples

```
as.LongTriangle(GenIns)
## Not run:
ggplot(as.LongTriangle(GenIns),
       aes(x = dev, y = value, group = origin, color = origin)) + geom_line()

## End(Not run)
```

---

ata *Calculate Age-to-Age Factors*

Description

Calculate the matrix of age-to-age factors (also called "report-to-report" factors, or "link ratios") for an object of class `triangle`.

Usage

```
ata(Triangle, NArow.rm = TRUE, colname.sep = "-",
        colname.order=c("ascending","descending"))
```

Arguments

`Triangle`	a loss "triangle". Must be a `matrix`.
`NArow.rm`	`logical` indicating if rows of age-to-age (ata) factors that are all `NA` should be removed. "All-NA" rows typically occur for the most recent origin year of a loss triangle.
`colname.sep`	a `character` indicating the separator character to place between the column names of `Triangle` that will be used to lable the columns of the resulting matrix of ata factors
`colname.order`	"ascending" indicates that the less mature age comes first in the column labels of the ata matrix

Details

`ata` constructs a matrix of age-to-age (ata) factors resulting from a loss "triangle" or a `matrix`. Simple averages and volume weighted averages are saved as "smpl" and "vwtd" attributes, respectively.

Value

A matrix with "smpl" and "vwtd" attributes.

Author(s)

Daniel Murphy

See Also

summary.ata, print.ata and chainladder

Examples

```
ata(GenIns)

# Volume weighted average age-to-age factor of the "RAA" data
y <- attr(ata(RAA), "vwtd")
y
# "To ultimate" factors with a 10% tail
y <- rev(cumprod(rev(c(y, 1.1))))
names(y) <- paste(colnames(RAA), "Ult", sep="-")
y

## Label the development columns in "ratio-type" format
ata(RAA, colname.sep=":", colname.order="desc")
```

---

auto *Run off triangle of accumulated claim data*

Description

Run-off triangles of Personal Auto and Commercial Auto insurance.

Usage

```
data(auto)
```

Format

A list of three matrices, paid Personal Auto, incurred Personal Auto and paid Commercial Auto respectively.

Source

Zhang (2010). A general multivariate chain ladder model. Insurance: Mathematics and Economics, *46, pp. 588-599.*

Examples

```
data(auto)
names(auto)
```

`BootChainLadder` *Bootstrap-Chain-Ladder Model*

Description

The `BootChainLadder` procedure provides a predictive distribution of reserves or IBNRs for a cumulative claims development triangle.

Usage

```
BootChainLadder(Triangle, R = 999, process.distr=c("gamma", "od.pois"))
```

Arguments

`Triangle`	cumulative claims triangle. Assume columns are the development period, use transpose otherwise. A (mxn)-matrix C_{ik} which is filled for $k \leq n+1-i; i = 1, \ldots, m; m \geq n$. See `qpaid` for how to use (mxn)-development triangles with m<n, say higher development period frequency (e.g quarterly) than origin period frequency (e.g accident years).
`R`	the number of bootstrap replicates.
`process.distr`	character string indicating which process distribution to be assumed. One of "gamma" (default), or "od.pois" (over-dispersed Poisson), can be abbreviated

Details

The `BootChainLadder` function uses a two-stage bootstrapping/simulation approach. In the first stage an ordinary chain-ladder methods is applied to the cumulative claims triangle. From this we calculate the scaled Pearson residuals which we bootstrap R times to forecast future incremental claims payments via the standard chain-ladder method. In the second stage we simulate the process error with the bootstrap value as the mean and using the process distribution assumed. The set of reserves obtained in this way forms the predictive distribution, from which summary statistics such as mean, prediction error or quantiles can be derived.

Value

BootChainLadder gives a list with the following elements back:

`call`	matched call
`Triangle`	input triangle
`f`	chain-ladder factors
`simClaims`	array of dimension `c(m,n,R)` with the simulated claims
`IBNR.ByOrigin`	array of dimension `c(m,1,R)` with the modeled IBNRs by origin period

IBNR.Triangles	array of dimension c(m,n,R) with the modeled IBNR development triangles
IBNR.Totals	vector of R samples of the total IBNRs
ChainLadder.Residuals	adjusted Pearson chain-ladder residuals
process.distr	assumed process distribution
R	the number of bootstrap replicates

Note

The implementation of BootChainLadder follows closely the discussion of the bootstrap model in section 8 and appendix 3 of the paper by England and Verrall (2002).

Author(s)

Markus Gesmann, <markus.gesmann@gmail.com>

References

England, PD and Verrall, RJ. Stochastic Claims Reserving in General Insurance (with discussion), British Actuarial Journal *8, III. 2002*

Barnett and Zehnwirth. The need for diagnostic assessment of bootstrap predictive models, Insureware technical report. *2007*

See Also

See also summary.BootChainLadder, plot.BootChainLadder displaying results and finally CDR.BootChainLadder for the one year claims development result.

Examples

```
# See also the example in section 8 of England & Verrall (2002) on page 55.

B <- BootChainLadder(RAA, R=999, process.distr="gamma")
B
plot(B)
# Compare to MackChainLadder
MackChainLadder(RAA)
quantile(B, c(0.75,0.95,0.99, 0.995))

# fit a distribution to the IBNR
library(MASS)
plot(ecdf(B$IBNR.Totals))
# fit a log-normal distribution
fit <- fitdistr(B$IBNR.Totals[B$IBNR.Totals>0], "lognormal")
fit
curve(plnorm(x,fit$estimate["meanlog"], fit$estimate["sdlog"]), col="red", add=TRUE)

# See also the ABC example in  Barnett and Zehnwirth (2007)
A <- BootChainLadder(ABC, R=999, process.distr="gamma")
A
```

```
plot(A, log=TRUE)

## One year claims development result
CDR(A)
```

CDR *One year claims development result*

Description

Standard deviation of the claims development result after one year for the distribution-free chain-ladder model (Mack) and Bootstrap model.

Usage

```
CDR(x, ...)
## S3 method for class 'MackChainLadder'
CDR(x, dev=1, ...)
## S3 method for class 'BootChainLadder'
CDR(x, probs=c(0.75, 0.95), ...)
## Default S3 method:
CDR(x, ...)
```

Arguments

x	otput of either MackChainLadder or BootChainLadder
dev	vector of development periods or "all". Currently only applicable for MackChainLadder output. Defines the years for which the run off claims development result should be returned.
probs	only applicable for BootChainLadder output. Define quantiles to be returned.
...	other arguments

Details

Merz & Wüthrich (2008) derived analytic formulae for the mean square error of prediction of the claims development result for the Mack chain-ladder model after one year assuming:

- The opening reserves were set using the pure chain-ladder model (no tail)
- Claims develop in the year according to the assumptions underlying Mack's model
- Reserves are set after one year using the pure chain-ladder model (no tail)

Value

A data.frame with various IBNR/reserves and one-year statistics of the claims development result.

Note

Tail factors are currently not supported.

Author(s)

Mario Wüthrich and Markus Gesmann with contributions from Arthur Charpentier and Arnaud Lacoume for `CDR.MackChainLadder` and Giuseppe Crupi and Markus Gesmann for `CDR.BootChainLadder`.

References

Michael Merz, Mario V. Wüthrich. Modelling the claims development result for solvency purposes. *Casualty Actuarial Society E-Forum, Fall 2008.*

Michael Merz, Mario V. Wüthrich. Claims Run-Off Uncertainty: The Full Picture. *Swiss Finance Institute Research Paper No. 14-69.* https://ssrn.com/abstract=2524352. *2014*

See Also

See also `MackChainLadder` and `BootChainLadder`

Examples

```
# Example from the 2008 Merz, Wuthrich paper mentioned above
MW2008
M <- MackChainLadder(MW2008, est.sigma="Mack")
plot(M)
CDR(M)
# Return all run-off result developments
CDR(M, dev="all")

# Example from the 2014 Merz, Wuthrich paper mentioned above
MW2014
W <- MackChainLadder(MW2014, est.sigma="Mack")
plot(W)
CDR(W)

# Example with the BootChainLadder function, assuming overdispered Poisson model
B <- BootChainLadder(MW2008, process.distr=c("od.pois"))
B
CDR(B)
```

`chainladder` *Estimate age-to-age factors*

Description

Basic chain-ladder function to estimate age-to-age factors for a given cumulative run-off triangle. This function is used by Mack- and MunichChainLadder.

Usage

```
chainladder(Triangle, weights = 1, delta = 1)
```

Arguments

`Triangle`	cumulative claims triangle. A (mxn)-matrix C_{ik} which is filled for $k \leq n+1-i; i = 1, \ldots, m; m \geq n$, see `qpaid` for how to use (mxn)-development triangles with m<n, say higher development period frequency (e.g quarterly) than origin period frequency (e.g annual).
`weights`	weights. Default: 1, which sets the weights for all triangle entries to 1. Otherwise specify weights as a matrix of the same dimension as `Triangle` with all weight entries in [0; 1], where entry $w_{i,k}$ corresponds to the point $C_{i,k+1}/C_{i,k}$. Hence, any entry set to 0 or `NA` eliminates that age-to-age factor from inclusion in the model. See also 'Details'.
`delta`	'weighting' parameters. Default: 1; delta=1 gives the historical chain-ladder age-to-age factors, delta=2 gives the straight average of the observed individual development factors and delta=0 is the result of an ordinary regression of $C_{i,k+1}$ against $C_{i,k}$ with intercept 0, see Barnett & Zehnwirth (2000). Please note that `MackChainLadder` uses the argument `alpha`, with `alpha = 2 - delta`, following the original paper Mack (1999)

Details

The key idea is to see the chain-ladder algorithm as a special form of a weighted linear regression through the origin, applied to each development period.

Suppose `y` is the vector of cumulative claims at development period `i+1`, and `x` at development period `i`, `weights` are weighting factors and `F` the individual age-to-age factors `F=y/x`. Then we get the various age-to-age factors:

- Basic (unweighted) linear regression through the origin: `lm(y~x + 0)`
- Basic weighted linear regression through the origin: `lm(y~x + 0, weights=weights)`
- Volume weighted chain-ladder age-to-age factors: `lm(y~x + 0, weights=1/x)`
- Simple average of age-to-age factors: `lm(y~x + 0, weights=1/x^2)`

Barnett & Zehnwirth (2000) use delta = 0, 1, 2 to distinguish between the above three different regression approaches: `lm(y~x + 0, weights=weights/x^delta)`.

Thomas Mack uses the notation `alpha = 2 - delta` to achieve the same result: `sum(weights*x^alpha*F)/sum(weights*x`

Value

chainladder returns a list with the following elements:

`Models`	linear regression models for each development period
`Triangle`	input triangle of cumulative claims
`weights`	weights used
`delta`	deltas used

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>

References

Thomas Mack. The standard error of chain ladder reserve estimates: Recursive calculation and inclusion of a tail factor. Astin Bulletin. *Vol. 29. No 2. 1999. pp.361:366*

G. Barnett and B. Zehnwirth. Best Estimates for Reserves. Proceedings of the CAS. *Volume LXXXVII. Number 167. November 2000.*

See Also

See also `ata`, `predict.ChainLadder` `MackChainLadder`,

Examples

```
## Concept of different chain-ladder age-to-age factors.
## Compare Mack's and Barnett & Zehnwirth's papers.
x <- RAA[1:9,1]
y <- RAA[1:9,2]

F <- y/x
## wtd. average chain-ladder age-to-age factors
alpha <- 1 ## Mack notation
delta <- 2 - alpha ## Barnett & Zehnwirth notation

sum(x^alpha*F)/sum(x^alpha)
lm(y~x + 0 ,weights=1/x^delta)
summary(chainladder(RAA, delta=delta)$Models[[1]])$coef

## straight average age-to-age factors
alpha <- 0
delta <- 2 - alpha
sum(x^alpha*F)/sum(x^alpha)
lm(y~x + 0, weights=1/x^(2-alpha))
summary(chainladder(RAA, delta=delta)$Models[[1]])$coef

## ordinary regression age-to-age factors
alpha=2
delta <- 2-alpha
sum(x^alpha*F)/sum(x^alpha)
lm(y~x + 0, weights=1/x^delta)
summary(chainladder(RAA, delta=delta)$Models[[1]])$coef

## Compare different models
CL0 <- chainladder(RAA)
## age-to-age factors
sapply(CL0$Models, function(x) summary(x)$coef["x","Estimate"])
## f.se
sapply(CL0$Models, function(x) summary(x)$coef["x","Std. Error"])
## sigma
```

```
sapply(CL0$Models, function(x) summary(x)$sigma)
predict(CL0)

CL1 <- chainladder(RAA, delta=1)
## age-to-age factors
sapply(CL1$Models, function(x) summary(x)$coef["x","Estimate"])
## f.se
sapply(CL1$Models, function(x) summary(x)$coef["x","Std. Error"])
## sigma
sapply(CL1$Models, function(x) summary(x)$sigma)
predict(CL1)

CL2 <- chainladder(RAA, delta=2)
## age-to-age factors
sapply(CL2$Models, function(x) summary(x)$coef["x","Estimate"])
## f.se
sapply(CL2$Models, function(x) summary(x)$coef["x","Std. Error"])
## sigma
sapply(CL2$Models, function(x) summary(x)$sigma)
predict(CL2)

## Set 'weights' parameter to use only the last 5 diagonals,
## i.e. the last 5 calendar years
calPeriods <- (row(RAA) + col(RAA) - 1)
(weights <- ifelse(calPeriods <= 5, 0, ifelse(calPeriods > 10, NA, 1)))
CL3 <- chainladder(RAA, weights=weights)
summary(CL3$Models[[1]])$coef
predict(CL3)
```

`ClarkCapeCod` *Clark Cape Cod method*

Description

Analyze loss triangle using Clark's Cape Cod method.

Usage

```
ClarkCapeCod(Triangle, Premium, cumulative = TRUE, maxage = Inf,
        adol = TRUE, adol.age = NULL, origin.width = NULL,
        G = "loglogistic")
```

Arguments

`Triangle`	A loss triangle in the form of a matrix. The number of columns must be at least four; the number of rows may be as few as 1. The column names of the matrix should be able to be interpreted as the "age" of the losses in that column. The row names of the matrix should uniquely define the year of origin of the losses in that row. Losses may be inception-to-date or incremental.

Premium	The vector of premium to use in the method. If a scalar (vector of length 1) is given, that value will be used for all origin periods. (See "Examples" below.) If the length is greater than 1 but does not equal the number of rows of Triangle the Premium values will be "recycled" with a warning.
cumulative	If TRUE (the default), values in Triangle are inception to date. If FALSE, Triangle holds incremental losses.
maxage	The "ultimate" age to which losses should be projected.
adol	If TRUE (the default), the growth function should be applied to the length of time from the average date of loss ("adol") of losses in the origin year. If FALSE, the growth function should be applied to the length of time since the beginning of the origin year.
adol.age	Only pertinent if adol is TRUE. The age of the average date of losses within an origin period in the same units as the "ages" of the Triangle matrix. If NULL (the default) it will be assumed to be half the width of an origin period (which would be the case if losses can be assumed to occur uniformly over an origin period).
origin.width	Only pertinent if adol is TRUE. The width of an origin period in the same units as the "ages" of the Triangle matrix. If NULL (the default) it will be assumed to be the mean difference in the "ages" of the triangle, with a warning if not all differences are equal.
G	A character scalar identifying the "growth function." The two growth functions defined at this time are "loglogistic" (the default) and "weibull".

Details

Clark's "Cape Cod" method assumes that the incremental losses across development periods in a loss triangle are independent. He assumes that the expected value of an incremental loss is equal to the *theoretical* expected loss ratio (**ELR**) times the on-level premium for the origin year times the change in the *theoretical* underlying growth function over the development period. Clark models the growth function, also called the percent of ultimate, by either the loglogistic function (a.k.a., "the inverse power curve") or the weibull function. Clark completes his incremental loss model by wrapping the expected values within an overdispersed poisson (ODP) process where the "scale factor" sigma^2 is assumed to be a known constant for all development periods.

The parameters of Clark's "Cape Cod" method are therefore: ELR, and omega and theta (the parameters of the **loglogistic** and **weibull** growth functions). Finally, Clark uses maximum likelihood to parameterize his model, uses the ODP process to estimate process risk, and uses the Cramer-Rao theorem and the "delta method" to estimate parameter risk.

Clark recommends inspecting the residuals to help assess the reasonableness of the model relative to the actual data (see plot.clark below).

Value

A list of class "ClarkLDF" with the components listed below. ("Key" to naming convention: all caps represent parameters; mixed case represent origin-level amounts; all-lower-case represent observation-level (origin, development age) results.)

method	"CapeCod"

growthFunction	name of the growth function
Origin	names of the rows of the triangle
Premium	Premium amount for each origin year
CurrentValue	the most mature value for each row
CurrentAge	the most mature "age" for each row
CurrentAge.used	the most mature age used; differs from "CurrentAge" when adol=TRUE
MAXAGE	same as 'maxage' argument
MAXAGE.USED	the maximum age for development from the average date of loss; differs from MAXAGE when adol=TRUE
FutureValue	the projected loss amounts ("Reserves" in Clark's paper)
ProcessSE	the process standard error of the FutureValue
ParameterSE	the parameter standard error of the FutureValue
StdError	the total standard error (process + parameter) of the FutureValue
Total	a list with amounts that appear on the "Total" row for components "Origin" (="Total"), "CurrentValue", "FutureValue", "ProcessSE", "ParameterSE", and "StdError"
PAR	the estimated parameters
ELR	the estimated loss ratio parameter
THETAG	the estimated parameters of the growth function
GrowthFunction	value of the growth function as of the CurrentAge.used
GrowthFunctionMAXAGE	value of the growth function as of the MAXAGE.used
FutureGrowthFactor	the ("unreported" or "unpaid") percent of ultimate loss that has yet to be recorded
SIGMA2	the estimate of the sigma^2 parameter
Ldf	the "to-ultimate" loss development factor (sometimes called the "cumulative development factor") as defined in Clark's paper for each origin year
LdfMAXAGE	the "to-ultimate" loss development factor as of the maximum age used in the model
TruncatedLdf	the "truncated" loss development factor for developing the current diagonal to the maximum age used in the model
FutureValueGradient	the gradient of the FutureValue function
origin	the origin year corresponding to each observed value of incremental loss
age	the age of each observed value of incremental loss
fitted	the expected value of each observed value of incremental loss (the "mu's" of Clark's paper)
residuals	the actual minus fitted value for each observed incremental loss
stdresid	the standardized residuals for each observed incremental loss (= residuals/sqrt(sigma2*fitted), referred to as "normalized residuals" in Clark's paper; see p. 62)

FI	the "Fisher Information" matrix as defined in Clark's paper (i.e., without the sigma^2 value)
value	the value of the loglikelihood function at the solution point
counts	the number of calls to the loglikelihood function and its gradient function when numerical convergence was achieved

Author(s)

Daniel Murphy

References

Clark, David R., "LDF Curve-Fitting and Stochastic Reserving: A Maximum Likelihood Approach", *Casualty Actuarial Society Forum*, Fall, 2003 https://www.casact.org/pubs/forum/03fforum/03ff041.pdf

See Also

ClarkLDF

Examples

```
X <- GenIns
colnames(X) <- 12*as.numeric(colnames(X))
CC.loglogistic  <- ClarkCapeCod(X, Premium=10000000+400000*0:9, maxage=240)
CC.loglogistic

# Clark's "CapeCod method" also works with triangles that have
# more development periods than origin periods. The Premium
# is a contrived match to the "made up" 'qincurred' Triangle.
ClarkCapeCod(qincurred, Premium=1250+150*0:11, G="loglogistic")

# Method also works for a "triangle" with only one row:
# 1st row of GenIns; need "drop=FALSE" to avoid becoming a vector.
ClarkCapeCod(GenIns[1, , drop=FALSE], Premium=1000000, maxage=20)

# If one value of Premium is appropriate for all origin years
# (e.g., losses are on-level and adjusted for exposure)
# then only a single value for Premium need be provided.
ClarkCapeCod(GenIns, Premium=1000000, maxage=20)

# Use of the weibull function generates a warning that the parameter risk
# approximation results in some negative variances. This may be of small
# concern since it happens only for older years with near-zero
# estimated reserves, but the warning should not be disregarded
# if it occurs with real data.
Y <- ClarkCapeCod(qincurred, Premium=1250+150*0:11, G="weibull")

# The plot of the standardized residuals by age indicates that the more
# mature observations are more loosely grouped than the less mature, just
# the opposite of the behavior under the loglogistic curve.
```

```
# This suggests that the model might be improved by analyzing the Triangle
# in two different "blocks": less mature vs. more mature.
# The QQ-plot shows that the tails of the empirical distribution of
# standardized residuals are "fatter" than a standard normal.
# The fact that the p-value is essentially zero says that there is
# virtually no chance that the standardized residuals could be
# considered draws from a standard normal random variable.
# The overall conclusion is that Clark's ODP-based CapeCod model with
# the weibull growth function does not match up well with the qincurred
# triangle and these premiums.
plot(Y)
```

`ClarkLDF` *Clark LDF method*

Description

Analyze loss triangle using Clark's LDF (loss development factor) method.

Usage

```
ClarkLDF(Triangle, cumulative = TRUE, maxage = Inf,
        adol = TRUE, adol.age = NULL, origin.width = NULL,
        G = "loglogistic")
```

Arguments

`Triangle`	A loss triangle in the form of a matrix. The number of columns must be at least four; the number of rows may be as few as 1. The column names of the matrix should be able to be interpreted as the "age" of the losses in that column. The row names of the matrix should uniquely define the year of origin of the losses in that row. Losses may be inception-to-date or incremental. The "ages" of the triangle can be "phase shifted" – i.e., the first age need not be as at the end of the origin period. (See the Examples section.) Nor need the "ages" be uniformly spaced. However, when the ages are not uniformly spaced, it would be prudent to specify the `origin.width` argument.
`cumulative`	If `TRUE` (the default), values in `Triangle` are inception to date. If `FALSE`, `Triangle` holds incremental losses.
`maxage`	The "ultimate" age to which losses should be projected.
`adol`	If `TRUE` (the default), the growth function should be applied to the length of time from the average date of loss ("adol") of losses in the origin year. If `FALSE`, the growth function should be applied to the length of time since the beginning of the origin year.
`adol.age`	Only pertinent if `adol` is `TRUE`. The age of the average date of losses within an origin period in the same units as the "ages" of the `Triangle` matrix. If `NULL` (the default) it will be assumed to be half the width of an origin period (which would be the case if losses can be assumed to occur uniformly over an origin period).

origin.width — Only pertinent if adol is TRUE. The width of an origin period in the same units as the "ages" of the Triangle matrix. If NULL (the default) it will be assumed to be the mean difference in the "ages" of the triangle, with a warning if not all differences are equal.

G — A character scalar identifying the "growth function." The two growth functions defined at this time are "loglogistic" (the default) and "weibull".

Details

Clark's "LDF method" assumes that the incremental losses across development periods in a loss triangle are independent. He assumes that the expected value of an incremental loss is equal to the *theoretical* expected ultimate loss (**U**) (by origin year) times the change in the *theoretical* underlying growth function over the development period. Clark models the growth function, also called the percent of ultimate, by either the loglogistic function (a.k.a., "the inverse power curve") or the weibull function. Clark completes his incremental loss model by wrapping the expected values within an overdispersed poisson (ODP) process where the "scale factor" sigma^2 is assumed to be a known constant for all development periods.

The parameters of Clark's "LDF method" are therefore: U, and omega and theta (the parameters of the **loglogistic** and **weibull** growth functions). Finally, Clark uses maximum likelihood to parameterize his model, uses the ODP process to estimate process risk, and uses the Cramer-Rao theorem and the "delta method" to estimate parameter risk.

Clark recommends inspecting the residuals to help assess the reasonableness of the model relative to the actual data (see plot.clark below).

Value

A list of class "ClarkLDF" with the components listed below. ("Key" to naming convention: all caps represent parameters; mixed case represent origin-level amounts; all-lower-case represent observation-level (origin, development age) results.)

method — "LDF"

growthFunction — name of the growth function

Origin — names of the rows of the triangle

CurrentValue — the most mature value for each row

CurrentAge — the most mature "age" for each row

CurrentAge.used — the most mature age used; differs from "CurrentAge" when adol=TRUE

MAXAGE — same as 'maxage' argument

MAXAGE.USED — the maximum age for development from the average date of loss; differs from MAXAGE when adol=TRUE

FutureValue — the projected loss amounts ("Reserves" in Clark's paper)

ProcessSE — the process standard error of the FutureValue

ParameterSE — the parameter standard error of the FutureValue

StdError — the total standard error (process + parameter) of the FutureValue

Total	a list with amounts that appear on the "Total" row for components "Origin" (="Total"), "CurrentValue", "FutureValue", "ProcessSE", "ParameterSE", and "StdError"
PAR	the estimated parameters
THETAU	the estimated parameters for the "ultimate loss" by origin year ("U" in Clark's notation)
THETAG	the estimated parameters of the growth function
GrowthFunction	value of the growth function as of the CurrentAge.used
GrowthFunctionMAXAGE	value of the growth function as of the MAXAGE.used
SIGMA2	the estimate of the sigma^2 parameter
Ldf	the "to-ultimate" loss development factor (sometimes called the "cumulative development factor") as defined in Clark's paper for each origin year
LdfMAXAGE	the "to-ultimate" loss development factor as of the maximum age used in the model
TruncatedLdf	the "truncated" loss development factor for developing the current diagonal to the maximum age used in the model
FutureValueGradient	the gradient of the FutureValue function
origin	the origin year corresponding to each observed value of incremental loss
age	the age of each observed value of incremental loss
fitted	the expected value of each observed value of incremental loss (the "mu's" of Clark's paper)
residuals	the actual minus fitted value for each observed incremental loss
stdresid	the standardized residuals for each observed incremental loss (= residuals/sqrt(sigma2*fitted), referred to as "normalized residuals" in Clark's paper; see p. 62)
FI	the "Fisher Information" matrix as defined in Clark's paper (i.e., without the sigma^2 value)
value	the value of the loglikelihood function at the solution point
counts	the number of calls to the loglikelihood function and its gradient function when numerical convergence was achieved

Author(s)

Daniel Murphy

References

Clark, David R., "LDF Curve-Fitting and Stochastic Reserving: A Maximum Likelihood Approach", *Casualty Actuarial Society Forum*, Fall, 2003 https://www.casact.org/pubs/forum/03fforum/03ff041.pdf

See Also

ClarkCapeCod

Examples

```
X <- GenIns
ClarkLDF(X, maxage=20)

# Clark's "LDF method" also works with triangles that have
# more development periods than origin periods
ClarkLDF(qincurred, G="loglogistic")

# Method also works for a "triangle" with only one row:
# 1st row of GenIns; need "drop=FALSE" to avoid becoming a vector.
ClarkLDF(GenIns[1, , drop=FALSE], maxage=20)

# The age of the first evaluation may be prior to the end of the origin period.
# Here the ages are in units of "months" and the first evaluation
# is at the end of the third quarter.
X <- GenIns
colnames(X) <- 12 * as.numeric(colnames(X)) - 3
# The indicated liability increases from 1st example above,
# but not significantly.
ClarkLDF(X, maxage=240)
# When maxage is infinite, the phase shift has a more noticeable impact:
# a 4-5% increase of the overall CV.
x <- ClarkLDF(GenIns, maxage=Inf)
y <- ClarkLDF(X, maxage=Inf)
# Percent change in the bottom line CV:
(tail(y$Table65$TotalCV, 1) - tail(x$Table65$TotalCV, 1)) / tail(x$Table65$TotalCV, 1)
```

`CLFMdelta` *Find "selection consistent" values of delta*

Description

This function finds the values of delta, one for each development period, such that the model coefficients resulting from the 'chainladder' function with delta = solution are consistent with an input vector of 'selected' development age-to-age factors.

Usage

```
CLFMdelta(Triangle, selected, tolerance = .0005, ...)
```

Arguments

`Triangle`	cumulative claims triangle. A (mxn)-matrix C_{ik} which is filled for $k \leq n+1-i; i = 1, \ldots, m; m \geq n$, see `qpaid` for how to use (mxn)-development triangles with m<n, say higher development period frequency (e.g quarterly) than origin period frequency (e.g accident years).

`selected`	a vector of selected age-to-age factors or "link ratios", one for each development period of 'Triangle'
`tolerance`	a 'tolerance' parameters. Default: .0005; for each element of 'selected' a solution 'delta' will be found – if possible – so that the chainladder model indexed by 'delta' results in a multiplicative coefficient within 'tolerance' of the 'selected' factor.
`...`	not in use

Details

For a given input Triangle and vector of selected factors, a search is conducted for chainladder models that are "consistent with" the selected factors. By "consistent with" is meant that the coefficients of the `chainladder` function are equivalent to the selected factors. Not all vectors of selected factors can be considered consistent with a given Triangle; feasibility is subject to restrictions on the 'selected' factors relative to the input 'Triangle'. See the References.

The default average produced by the `chainladder` function is the volume weighted average and corresponds to `delta = 1` in the call to that function; `delta = 2` produces the simple average; and `delta = 0` produces the "regression average", i.e., the slope of a regression line fit to the data and running through the origin. By convention, if the `selected` value corresponds to the volume-weighted average, the simple average, or the regression average, then the value returned will be 1, 2, and 0, respectively.

Other real-number values for `delta` will produce a different average. The point of this function is to see if there exists a model as determined by delta whose average is consistent with the value in the `selected` vector. That is not always possible. See the References.

It can be the case that one or more of the above three "standard averages" will be quite close to each other (indistinguishable within `tolerance`). In that case, the value returned will be, in the following priority order by convention, 1 (volume weighted average), 2 (simple average), or 0 (regression average).

Value

A vector of real numbers, say delta0, such that `coef(chainladder(Triangle, delta = delta0)) = selected` within `tolerance`. A `logical` attribute 'foundSolution' indicates if a solution was found for each element of `selected`.

Author(s)

Dan Murphy

References

Bardis, Majidi, Murphy. A Family of Chain-Ladder Factor Models for Selected Link Ratios. Variance. *Pending. Variance 6:2, 2012, pp. 143-160.* https://www.variancejournal.org/issues/06-02/143.pdf

Examples

```
x <- RAA[1:9,1]
y <- RAA[1:9,2]
F <- y/x
CLFMdelta(RAA[1:9, 1:2], selected = mean(F)) # value is 2, 'foundSolution' is TRUE
CLFMdelta(RAA[1:9, 1:2], selected = sum(y) / sum(x)) # value is 1, 'foundSolution' is TRUE

selected <- mean(c(mean(F), sum(y) / sum(x))) # an average of averages
CLFMdelta(RAA[1:9, 1:2], selected) # about 1.725
CLFMdelta(RAA[1:9, 1:2], selected = 2) # negative solutions are possible

# Demonstrating an "unreasonable" selected factor.
CLFMdelta(RAA[1:9, 1:2], selected = 1.9) # NA solution, with warning
```

---

coef.ChainLadder *Extract residuals of a ChainLadder model*

Description

Extract residuals of a MackChainLadder model by origin-, calendar- and development period.

Usage

```
## S3 method for class 'ChainLadder'
coef(object, ...)
```

Arguments

object	output of the chainladder function
...	optional arguments which may become named attributes of the resulting vector

Value

The function returns a vector of the single-parameter coefficients – also called age-to-age (ATA) or report-to-report (RTR) factors – of the models produced by running the 'chainladder' function.

Author(s)

Dan Murphy

See Also

See Also chainladder

Examples

```
coef(chainladder(RAA))
```

`Cumulative and incremental triangles`
Cumulative and incremental triangles

Description

Functions to convert between cumulative and incremental triangles

Usage

```
incr2cum(Triangle, na.rm=FALSE)
cum2incr(Triangle)
```

Arguments

`Triangle`	triangle. Assume columns are the development period, use transpose otherwise.
`na.rm`	logical. Should missing values be removed?

Details

`incr2cum` transforms an incremental triangle into a cumulative triangle, `cum2incr` provides the reserve operation.

Value

Both functions return a `triangle`.

Author(s)

Markus Gesmann, Christophe Dutang

See Also

See also `as.triangle`

Examples

```
# See the Taylor/Ashe example in Mack's 1993 paper

#original triangle
GenIns

#incremental triangle
cum2incr(GenIns)

#original triangle
incr2cum(cum2incr(GenIns))

# See the example in Mack's 1999 paper

#original triangle
Mortgage
incMortgage <- cum2incr(Mortgage)
#add missing values
incMortgage[1,1] <- NA
incMortgage[2,1] <- NA
incMortgage[1,2] <- NA

#with missing values argument
incr2cum(incMortgage, na.rm=TRUE)

#compared to
incr2cum(Mortgage)
```

`GenIns` *Run off triangle of claims data.*

Description

Run off triangle of accumulated general insurance claims data. `GenInsLong` provides the same data in a 'long' format.

Usage

```
GenIns
```

Format

A matrix with 10 accident years and 10 development years.

Source

TAYLOR, G.C. and ASHE, F.R. (1983) Second Moments of Estimates of Outstanding Claims. Journal of Econometrics ***23***, *37-61.*

References

See table 1 in: *Distribution-free Calculation of the Standard Error of Chain Ladder Reserve Estimates, Thomas Mack, 1993,* ASTIN Bulletin ***23****, 213 - 225*

Examples

```
GenIns
plot(GenIns)

plot(GenIns, lattice=TRUE)


head(GenInsLong)

## Convert long format into triangle
## Triangles are usually stored as 'long' tables in data bases
as.triangle(GenInsLong, origin="accyear", dev="devyear", "incurred claims")
```

`getLatestCumulative` *Triangle information for most recent calendar period.*

Description

Return most recent values for all origin periods of a cumulative development triangle.

Usage

```
getLatestCumulative(Triangle, na.values = NULL)
```

Arguments

`Triangle`	a Triangle in matrix format.
`na.values`	a vector specifying values that should be considered synonymous with NA when searching for the rightmost non-NA.

Value

A vector of most recent non-'NA' (and synonyms, if appropriate) values of a triangle for all origin periods. The `names` of the vector equal the origin names of the Triangle. The vector will have additional attributes: "latestcol" equalling the index of the column in Triangle corresponding to the row's rightmost entry; "rowsname" equalling the name of the row dimension of Triangle, if any; "colnames" equalling the corresponding column name of Triangle, if any; "colsname" equalling the name of the column dimension of Triangle, if any.

Author(s)

Ben Escoto, Markus Gesmann, Dan Murphy

See Also

See also as.triangle.

Examples

```
RAA
getLatestCumulative(RAA)
Y <- matrix(c(1,  2,  3,
              4,  5,  0,
              6, NA, NA), byrow=TRUE, nrow=3)
getLatestCumulative(Y) # c(3, 0, 6)
getLatestCumulative(Y, na.values = 0) # c(3, 5, 6)
```

---

glmReserve *GLM-based Reserving Model*

Description

This function implements loss reserving models within the generalized linear model framework. It takes accident year and development lag as mean predictors in estimating the ultimate loss reserves, and provides both analytical and bootstrapping methods to compute the associated prediction errors. The bootstrapping approach also generates the full predictive distribution for loss reserves.

Usage

```
glmReserve(triangle, var.power = 1, link.power = 0, cum = TRUE,
      mse.method = c("formula", "bootstrap"),  nsim = 1000, nb = FALSE, ...)
```

Arguments

triangle	An object of class triangle.
var.power	The index (p) of the power variance function $V(\mu) = \mu^p$. Default to p = 1, which is the over-dispersed Poisson model. If NULL, it will be assumed to be in (1, 2) and estimated using the cplm package. See tweedie.
link.power	The index of power link function. The default link.power = 0 produces a log link. See tweedie.
cum	A logical value indicating whether the input triangle is on the cumulative or the incremental scale. If TRUE, then triangle is assumed to be on the cumulative scale, and it will be converted to incremental losses internally before a GLM is fitted.
mse.method	A character indicating whether the prediction error should be computed analytically (mse.method = "formula") or via bootstrapping (mse.method = "bootstrap"). Partial match is supported.
nsim	Number of simulations to be performed in the bootstrapping, with a default value of 1000.

nb — Whether the negative binomial distribution is used. If `true`, the arguments `var.power` and `link.power` are ignored and a negative binomial GLM is fitted using `glm.nb`.

... — Arguments to be passed onto the function `glm` or `cpglm` such as `contrasts` or `control`. It is important that `offset` and `weight` should not be specified. Otherwise, an error will be reported and the program will quit.

Details

This function takes an insurance loss triangle, converts it to incremental losses internally if necessary, transforms it to the long format (see `as.data.frame`) and fits the resulting loss data with a generalized linear model where the mean structure includes both the accident year and the development lag effects. The distributions allowed are the exponential family that admits a power variance function, that is, $V(\mu) = \mu^p$. This subclass of distributions is usually called the Tweedie distribution and includes many commonly used distributions as special cases.

This function does not allow the user to specify the GLM options through the usual `family` argument, but instead, it uses the `tweedie` family internally and takes two arguments, `var.power` and `link.power`, through which the user still has full control of the distribution forms and link functions. The argument `var.power` determines which specific distribution is to be used, and `link.power` determines the form of the link function.

When the Tweedie compound Poisson distribution `1 < p < 2` is to be used, the user has the option to specify `var.power = NULL`, where the variance power `p` will be estimated from the data using the `cplm` package. The `bcplm` function in the `cplm` package also has an example for the Bayesian compound Poisson loss reserving model. See details in `tweedie`, `cpglm` and `bcplm`.

`glmReserve` allows certain measures of exposures to be used in an offset term in the underlying GLM. To do this, the user should not use the usual `offset` argument in `glm`. Instead, one specifies the exposure measure for each accident year through the `exposure` attribute of `triangle`. Make sure that these exposures are in the original scale (no log transformations for example). If the vector is named, make sure the names coincide with the rownames/origin of the triangle. If the vector is unnamed, make sure the exposures are in the order consistent with the accident years, **and the character rownames of the Triangle must be convertible to numeric.** If the `exposure` attribute is not `NULL`, the `glmReserve` function will use these exposures, link-function-transformed, in the offset term of the GLM. For example, if the link function is `log`, then the log of the exposure is used as the offset, not the original exposure. See the examples below. Moreover, the user MUST NOT supply the typical `offset` or `weight` as arguments in the list of additional arguments `....` `offset` should be specified as above, while `weight` is not implemented (due to prediction reasons).

Two methods are available to assess the prediction error of the estimated loss reserves. One is using the analytical formula (`mse.method = "formula"`) derived from the first-order Taylor approximation. The other is using bootstrapping (`mse.method = "bootstrap"`) that reconstructs the triangle `nsim` times by sampling with replacement from the GLM (Pearson) residuals. Each time a new triangle is formed, GLM is fitted and corresponding loss reserves are generated. Based on these predicted mean loss reserves, and the model assumption about the distribution forms, realizations of the predicted values are generated via the `rtweedie` function. Prediction errors as well as other uncertainty measures such as quantiles and predictive intervals can be calculated based on these samples.

Value

The output is an object of class "glmReserve" that has the following components:

call	the matched call.
summary	A data frame containing the predicted loss reserve statistics. Similar to the summary statistics from MackChainLadder.
Triangle	The input triangle.
FullTriangle	The completed triangle, where empty cells in the original triangle are filled with model predictions.
model	The fitted GLM, a class of "glm" or "cpglm". It is most convenient to work with this component when model fit information is wanted.
sims.par	a matrix of the simulated parameter values in the bootstrapping.
sims.reserve.mean	a matrix of the simulated mean loss reserves (without the process variance) for each year in the bootstrapping.
sims.par	a matrix of the simulated realizations of the loss reserves (with the process variance) for each year in the bootstrapping. This can be used to summarize the predictive uncertainty of the loss reserves.

Note

The use of GLM in insurance loss reserving has many compelling aspects, e.g.,

- when over-dispersed Poisson model is used, it reproduces the estimates from Chain Ladder;
- it provides a more coherent modeling framework than the Mack method;
- all the relevant established statistical theory can be directly applied to perform hypothesis testing and diagnostic checking;

However, the user should be cautious of some of the key assumptions that underlie the GLM model, in order to determine whether this model is appropriate for the problem considered:

- the GLM model assumes no tail development, and it only projects losses to the latest time point of the observed data. To use a model that enables tail extrapolation, please consider the growth curve model ClarkLDF or ClarkCapeCod;
- the model assumes that each incremental loss is independent of all the others. This assumption may not be valid in that cells from the same calendar year are usually correlated due to inflation or business operating factors;
- the model tends to be over-parameterized, which may lead to inferior predictive performance.

To solve these potential problems, many variants of the current basic GLM model have been proposed in the actuarial literature. Some of these may be included in the future release.

Support of the negative binomial GLM was added since version 0.2.3.

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

References

England P. and Verrall R. (1999). Analytic and bootstrap estimates of prediction errors in claims reserving. *Insurance: Mathematics and Economics*, 25, 281-293.

See Also

See also glm, glm.nb, tweedie, cpglm and MackChainLadder.

Examples

```
data(GenIns)
GenIns <- GenIns / 1000

# over-dispersed Poisson: reproduce ChainLadder estimates
(fit1 <- glmReserve(GenIns))
summary(fit1, type = "model")   # extract the underlying glm

# which:
# 1 Original triangle
# 2 Full triangle
# 3 Reserve distribution
# 4 Residual plot
# 5 QQ-plot

# plot original triangle
plot(fit1, which = 1, xlab = "dev year", ylab = "cum loss")

# plot residuals
plot(fit1, which = 4, xlab = "fitted values", ylab = "residuals")

# Gamma GLM:
(fit2 <- glmReserve(GenIns, var.power = 2))

# compound Poisson GLM (variance function estimated from the data):
(fit3 <- glmReserve(GenIns, var.power = NULL))

# Now suppose we have an exposure measure
# we can put it as an offset term in the model
# to do this, use the "exposure" attribute of the 'triangle'
expos <- (7 + 1:10 * 0.4) * 100
GenIns2 <- GenIns
attr(GenIns2, "exposure") <- expos
(fit4 <- glmReserve(GenIns2))
# If the triangle's rownames are not convertible to numeric,
# supply names to the exposures
GenIns3 <- GenIns2
rownames(GenIns3) <- paste0(2007:2016, "-01-01")
names(expos) <- rownames(GenIns3)
attr(GenIns3, "exposure") <- expos
(fit4b <- glmReserve(GenIns3))

# use bootstrapping to compute prediction error
```

```
## Not run:
set.seed(11)
(fit5 <- glmReserve(GenIns, mse.method = "boot"))

# compute the quantiles of the predicted loss reserves
t(apply(fit5$sims.reserve.pred, 2, quantile,
        c(0.025, 0.25, 0.5, 0.75, 0.975)))

# plot distribution of reserve
plot(fit5, which = 3)

## End(Not run)

# alternative over-dispersed Poisson: negative binomial GLM
(fit6 <- glmReserve(GenIns, nb = TRUE))
```

`Join2Fits` *Join Two Fitted MultiChainLadder Models*

Description

This function is created to facilitate the fitting of the multivariate functions when specifying different models in two different development periods, especially when separate chain-ladder is used in later periods.

Usage

```
Join2Fits(object1, object2)
```

Arguments

`object1`	An object of class "MultiChainLadder"
`object2`	An object of class "MultiChainLadder"

Details

The inputs must be of class "MultiChainLadder" because this function depends on the `model` slot to determine what kind of object is to be created and returned. If both objects have `"MCL"`, then an object of class "MCLFit" is created; if one has `"GMCL"` and one has `"MCL"`, then an object of class "GMCLFit" is created, where the one with `"GMCL"` is assumed to come from the first development periods; if both have `"GMCL"`, then an object of class "GMCLFit" is created.

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

See Also

See also `MultiChainLadder`

JoinFitMse *Join Model Fit and Mse Estimation*

Description

This function combines first momoent estimation from fitted regression models and second moment estimation from `Mse` method to construct an object of class "MultiChainLadder", for which a variety of methods are defined, such as `summary` and `plot`.

Usage

```
JoinFitMse(models, mse.models)
```

Arguments

`models`	fitted regression models, either of class "MCLFit" or "GMCLFit".
`mse.models`	output from a call to `Mse`, which is of class "MultiChainLadderMse".

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

See Also

See also `MultiChainLadder`.

liab *Run off triangle of accumulated claim data*

Description

Run-off triangles of General Liability and Auto Liability.

Usage

```
data(auto)
```

Format

A list of two matrices, General Liability and Auto Liability respectively.

Source

Braun C (2004). The prediction error of the chain ladder method applied to correlated run off triangles. ASTIN Bulletin *34(2): 399-423*

Examples

```
data(liab)
names(liab)
```

`LRfunction` *Calculate the Link Ratio Function*

Description

This calculates the link ratio function per the CLFM paper.

Usage

```
LRfunction(x, y, delta)
```

Arguments

`x`	beginning value of loss during a development period
`y`	ending value of loss during a development period
`delta`	numeric

Details

Calculated the link ratios resulting from a chainladder model over a development period indexed by (possibly vector valued) real number delta. See formula (5) in the References.

Value

A vector of link ratios.

Author(s)

Dan Murphy

References

Bardis, Majidi, Murphy. A Family of Chain-Ladder Factor Models for Selected Link Ratios. Variance. *Pending. 2013. pp.tbd:tbd*

Examples

```
x <- RAA[1:9,1]
y <- RAA[1:9,2]
delta <- seq(-2, 2, by = .1)
plot(delta, LRfunction(x, y, delta), type = "l")
```

M3IR5 *Run off triangle of claims data*

Description

Run off triangle of simulated incremental claims data

Usage

```
data(M3IR5)
```

Format

A matrix with simulated incremental claims of 14 accident years and 14 development years.

Source

Appendix A7 in *B. Zehnwirth. Probabilistic Development Factor Models with Applications to Loss Reserve Variability, Prediction Intervals, and Risk Based Capital.* Casualty Actuarial Science Forum. *Spring 1994. Vol. 2.*

Examples

```
M3IR5
plot(M3IR5)
plot(incr2cum(M3IR5), lattice=TRUE)
```

MackChainLadder *Mack Chain-Ladder Model*

Description

The Mack chain-ladder model forecasts future claims developments based on a historical cumulative claims development triangle and estimates the standard error around those.

Usage

```
MackChainLadder(Triangle, weights = 1, alpha=1, est.sigma="log-linear",
tail=FALSE, tail.se=NULL, tail.sigma=NULL, mse.method="Mack")
```

Arguments

`Triangle`	cumulative claims triangle. Assume columns are the development period, use transpose otherwise. A (mxn)-matrix C_{ik} which is filled for $k \leq n+1-i; i = 1, \ldots, m; m \geq n$, see `qpaid` for how to use (mxn)-development triangles with m<n, say higher development period frequency (e.g quarterly) than origin period frequency (e.g accident years).
`weights`	weights. Default: 1, which sets the weights for all triangle entries to 1. Otherwise specify weights as a matrix of the same dimension as `Triangle` with all weight entries in [0; 1]. Hence, any entry set to 0 or `NA` eliminates that age-to-age factor from inclusion in the model. See also 'Details'
`alpha`	'weighting' parameters. Default: 1 for all development periods; alpha=1 gives the historical chain-ladder age-to-age factors, alpha=0 gives the straight average of the observed individual development factors and alpha=2 is the result of an ordinary regression of $C_{i,k+1}$ against $C_{i,k}$ with intercept 0, see also 'Details' below, `chainladder` and Mack's 1999 paper
`est.sigma`	defines how to estimate $sigma_{n-1}$, the variability of the individual age-to-age factors at development time $n-1$. Default is "log-linear" for a log-linear regression, "Mack" for Mack's approximation from his 1999 paper. Alternatively the user can provide a numeric value. If the log-linear model appears to be inappropriate (p-value > 0.05) the 'Mack' method will be used instead and a warning message printed. Similarly, if Triangle is so small that log-linear regression is being attempted on a vector of only one non-NA average link ratio, the 'Mack' method will be used instead and a warning message printed.
`tail`	can be logical or a numeric value. If `tail=FALSE` no tail factor will be applied, if `tail=TRUE` a tail factor will be estimated via a linear extrapolation of $log(chain-ladder factors-1)$, if `tail` is a numeric value than this value will be used instead.
`tail.se`	defines how the standard error of the tail factor is estimated. Only needed if a tail factor > 1 is provided. Default is `NULL`. If `tail.se` is `NULL`, `tail.se` is estimated via "log-linear" regression, if `tail.se` is a numeric value than this value will be used instead.
`tail.sigma`	defines how to estimate individual tail variability. Only needed if a tail factor > 1 is provided. Default is `NULL`. If `tail.sigma` is `NULL`, `tail.sigma` is estimated via "log-linear" regression, if `tail.sigma` is a numeric value than this value will be used instead
`mse.method`	method used for the recursive estimate of the parameter risk component of the mean square error. Value "Mack" (default) coincides with Mack's formula; "Independence" includes the additional cross-product term as in Murphy and BBMW. Refer to References below.

Details

Following Mack's 1999 paper let C_{ik} denote the cumulative loss amounts of origin period (e.g. accident year) $i = 1, \ldots, m$, with losses known for development period (e.g. development year) $k \leq n+1-i$. In order to forecast the amounts C_{ik} for $k > n+1-i$ the Mack chain-ladder-model

assumes:

$$\text{CL1: } E[F_{ik}|C_{i1}, C_{i2}, \ldots, C_{ik}] = f_k \text{ with } F_{ik} = \frac{C_{i,k+1}}{C_{ik}}$$

$$\text{CL2: } Var(\frac{C_{i,k+1}}{C_{ik}}|C_{i1}, C_{i2}, \ldots, C_{ik}) = \frac{\sigma_k^2}{w_{ik} C_{ik}^{\alpha}}$$

$$\text{CL3: } \{C_{i1}, \ldots, C_{in}\}, \{C_{j1}, \ldots, C_{jn}\}, \text{ are independent for origin period } i \neq j$$

with $w_{ik} \in [0; 1]$, $\alpha \in \{0, 1, 2\}$. If these assumptions hold, the Mack chain-ladder gives an unbiased estimator for IBNR (Incurred But Not Reported) claims.

Here $w_{ik} are the$ `weights` $from above$.

The Mack chain-ladder model can be regarded as a special form of a weighted linear regression through the origin for each development period: `lm(y ~ x + 0, weights=weights/x^(2-alpha))`, where `y` is the vector of claims at development period $k + 1$ and `x` is the vector of claims at development period k.

Value

MackChainLadder returns a list with the following elements

`call`	matched call
`Triangle`	input triangle of cumulative claims
`FullTriangle`	forecasted full triangle
`Models`	linear regression models for each development period
`f`	chain-ladder age-to-age factors
`f.se`	standard errors of the chain-ladder age-to-age factors f (assumption CL1)
`F.se`	standard errors of the true chain-ladder age-to-age factors F_{ik} (square root of the variance in assumption CL2)
`sigma`	sigma parameter in CL2
`Mack.ProcessRisk`	variability in the projection of future losses not explained by the variability of the link ratio estimators (unexplained variation)
`Mack.ParameterRisk`	variability in the projection of future losses explained by the variability of the link-ratio estimators alone (explained variation)
`Mack.S.E`	total variability in the projection of future losses by the chain-ladder method; the square root of the mean square error of the chain-ladder estimate: Mack.S.E.^{2} = Mack.ProcessRisk^2 + Mack.ParameterRisk^2
`Total.Mack.S.E`	total variability of projected loss for all origin years combined
`Total.ProcessRisk`	vector of process risk estimate of the total of projected loss for all origin years combined by development period
`Total.ParameterRisk`	vector of parameter risk estimate of the total of projected loss for all origin years combined by development period

weights	weights used
alpha	alphas used
tail	tail factor used. If tail was set to TRUE the output will include the linear model used to estimate the tail factor

Note

Additional references for further reading:

England, PD and Verrall, RJ. Stochastic Claims Reserving in General Insurance (with discussion), British Actuarial Journal *8, III. 2002*

Barnett and Zehnwirth. Best estimates for reserves. Proceedings of the CAS, LXXXVI I(167), November 2000.

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>

References

Thomas Mack. Distribution-free calculation of the standard error of chain ladder reserve estimates. Astin Bulletin. *Vol. 23. No 2. 1993. pp.213:225*

Thomas Mack. The standard error of chain ladder reserve estimates: Recursive calculation and inclusion of a tail factor. Astin Bulletin. *Vol. 29. No 2. 1999. pp.361:366*

Murphy, Daniel M. Unbiased Loss Development Factors. Proceedings of the Casualty Actuarial Society Casualty Actuarial Society - *Arlington, Virginia 1994: LXXXI 154-222*

Buchwalder, Bühlmann, Merz, and Wüthrich. The Mean Square Error of Prediction in the Chain Ladder Reserving Method (Mack and Murphy Revisited). Astin Bulletin *Vol. 36. 2006. pp.521:542*

See Also

See also qpaid for dealing with non-square triangles, chainladder for the underlying chain-ladder method, summary.MackChainLadder, quantile.MackChainLadder, plot.MackChainLadder and residuals.MackChainLadder displaying results and finally CDR.MackChainLadder for the one year claims development result.

Examples

```
## See the Taylor/Ashe example in Mack's 1993 paper
GenIns
plot(GenIns)
plot(GenIns, lattice=TRUE)
GNI <- MackChainLadder(GenIns, est.sigma="Mack")
GNI$f
GNI$sigma^2
GNI # compare to table 2 and 3 in Mack's 1993 paper
plot(GNI)
plot(GNI, lattice=TRUE)
```

```
## Different weights
## Using alpha=0 will use straight average age-to-age factors
MackChainLadder(GenIns, alpha=0)$f
# You get the same result via:
apply(GenIns[,-1]/GenIns[,-10],2, mean, na.rm=TRUE)

## Only use the last 5 diagonals, i.e. the last 5 calendar years
calPeriods <- (row(GenIns) + col(GenIns) - 1)
(weights <- ifelse(calPeriods <= 5, 0, ifelse(calPeriods > 10, NA, 1)))
MackChainLadder(GenIns, weights=weights, est.sigma = "Mack")

## Tail
## See the example in Mack's 1999 paper
Mortgage
m <- MackChainLadder(Mortgage)
round(summary(m)$Totals["CV(IBNR)",], 2) ## 26% in Table 6 of paper
plot(Mortgage)
# Specifying the tail and its associated uncertainty parameters
MRT <- MackChainLadder(Mortgage, tail=1.05, tail.sigma=71, tail.se=0.02, est.sigma="Mack")
MRT
plot(MRT, lattice=TRUE)
# Specify just the tail and the uncertainty parameters will be estimated
MRT <- MackChainLadder(Mortgage, tail=1.05)
MRT$f.se[9] # close to the 0.02 specified above
MRT$sigma[9] # less than the 71 specified above
# Note that the overall CV dropped slightly
round(summary(MRT)$Totals["CV(IBNR)",], 2) ## 24%
# tail parameter uncertainty equal to expected value
MRT <- MackChainLadder(Mortgage, tail=1.05, tail.se = .05)
round(summary(MRT)$Totals["CV(IBNR)",], 2) ## 27%

## Parameter-risk (only) estimate of the total reserve = 3142387
tail(MRT$Total.ParameterRisk, 1) # located in last (ultimate) element
#  Parameter-risk (only) CV is about 19%
tail(MRT$Total.ParameterRisk, 1) / summary(MRT)$Totals["IBNR", ]

## Three terms in the parameter risk estimate
## First, the default (Mack) without the tail
m <- MackChainLadder(RAA, mse.method = "Mack")
summary(m)$Totals["Mack S.E.",]
## Then, with the third term
m <- MackChainLadder(RAA, mse.method = "Independence")
summary(m)$Totals["Mack S.E.",] ## Not significantly greater

## One year claims development results
M <- MackChainLadder(MW2014, est.sigma="Mack")
CDR(M)

## For more examples see:
## Not run:
 demo(MackChainLadder)
```

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

`MCLpaid` *Run off triangles of accumulated paid and incurred claims data.*

Description

Run-off triangles based on a fire portfolio

Usage

```
data(MCLpaid)
data(MCLincurred)
```

Format

A matrix with 7 origin years and 7 development years.

Source

Gerhard Quarg and Thomas Mack. Munich Chain Ladder. Blatter DGVFM. *26, Munich, 2004.*

Examples

```
MCLpaid
MCLincurred
op=par(mfrow=c(2,1))
plot(MCLpaid)
plot(MCLincurred)
par(op)
```

`Mortgage` *Run off triangle of accumulated claims data*

Description

Development triangle of a mortgage guarantee business

Usage

```
data(Mortgage)
```

Format

A matrix with 9 accident years and 9 development years.

Source

Competition Presented at a London Market Actuaries Dinner, D.E.A. Sanders, 1990

References

See table 4 in: *Distribution-free Calculation of the Standard Error of Chain Ladder Reserve Estimates, Thomas Mack, 1993*, ASTIN Bulletin ***23***, *213 - 225*

Examples

```
Mortgage
Mortgage
plot(Mortgage)
plot(Mortgage, lattice=TRUE)
```

`Mse-methods` *Methods for Generic Function Mse*

Description

`Mse` is a generic function to calculate mean square error estimations in the chain-ladder framework.

Usage

```
Mse(ModelFit, FullTriangles, ...)

## S4 method for signature 'GMCLFit,triangles'
Mse(ModelFit, FullTriangles, ...)
## S4 method for signature 'MCLFit,triangles'
Mse(ModelFit, FullTriangles, mse.method="Mack", ...)
```

Arguments

`ModelFit`	An object of class "GMCLFit" or "MCLFit".
`FullTriangles`	An object of class "triangles". Should be the output from a call of `predict`.
`mse.method`	Character strings that specify the MSE estimation method. Only works for "MCLFit". Use `"Mack"` for the generazliation of the Mack (1993) approach, and `"Independence"` for the conditional resampling approach in Merz and Wuthrich (2008).
`...`	Currently not used.

Details

These functions calculate the conditional mean square errors using the recursive formulas in Zhang (2010), which is a generalization of the Mack (1993, 1999) formulas. In the GMCL model, the conditional mean square error for single accident years and aggregated accident years are calcualted as:

$$\hat{mse}(\hat{Y}_{i,k+1}|D) = \hat{B}_k \hat{mse}(\hat{Y}_{i,k}|D)\hat{B}_k + (\hat{Y}'_{i,k} \otimes I)\hat{\Sigma}_{B_k}(\hat{Y}_{i,k} \otimes I) + \hat{\Sigma}_{\epsilon_{i_k}}.$$

$$\hat{mse}(\sum_{i=a_k}^{I} \hat{Y}_{i,k+1}|D) = \hat{B}_k \hat{mse}(\sum_{i=a_k+1}^{I} \hat{Y}_{i,k}|D)\hat{B}_k + (\sum_{i=a_k}^{I} \hat{Y}'_{i,k} \otimes I)\hat{\Sigma}_{B_k}(\sum_{i=a_k}^{I} \hat{Y}_{i,k} \otimes I) + \sum_{i=a_k}^{I} \hat{\Sigma}_{\epsilon_{i_k}}.$$

In the MCL model, the conditional mean square error from Merz and Wüthrich (2008) is also available, which can be shown to be equivalent as the following:

$$\hat{mse}(\hat{Y}_{i,k+1}|D) = (\hat{\beta}_k \hat{\beta}'_k) \odot \hat{mse}(\hat{Y}_{i,k}|D) + \hat{\Sigma}_{\beta_k} \odot (\hat{Y}_{i,k}\hat{Y}'_{i,k}) + \hat{\Sigma}_{\epsilon_{i_k}} + \hat{\Sigma}_{\beta_k} \odot \hat{mse}^E(\hat{Y}_{i,k}|D).$$

$$\hat{mse}(\sum_{i=a_k}^{I} \hat{Y}_{i,k+1}|D) = (\hat{\beta}_k \hat{\beta}'_k) \odot \sum_{i=a_k+1}^{I} \hat{mse}(\hat{Y}_{i,k}|D) + \hat{\Sigma}_{\beta_k} \odot (\sum_{i=a_k}^{I} \hat{Y}_{i,k} \sum_{i=a_k}^{I} \hat{Y}'_{i,k}) + \sum_{i=a_k}^{I} \hat{\Sigma}_{\epsilon_{i_k}} + \hat{\Sigma}_{\beta_k} \odot \sum_{i=a_k}^{I} \hat{mse}^E(\hat{Y}_{i,k}|D).$$

For the Mack approach in the MCL model, the cross-product term $\hat{\Sigma}_{\beta_k} \odot \hat{mse}^E(\hat{Y}_{i,k}|D)$in the above two formulas will drop out.

Value

`Mse` returns an object of class "MultiChainLadderMse" that has the following elements:

`mse.ay`	condtional mse for each accdient year
`mse.ay.est`	conditional estimation mse for each accdient year
`mse.ay.proc`	conditional process mse for each accdient year
`mse.total`	condtional mse for aggregated accdient years
`mse.total.est`	conditional estimation mse for aggregated accdient years
`mse.total.proc`	conditional process mse for aggregated accdient years
`FullTriangles`	completed triangles

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

References

*Zhang Y (2010). A general multivariate chain ladder model.*Insurance: Mathematics and Economics, *46, pp. 588-599.*

Zhang Y (2010). Prediction error of the general multivariate chain ladder model.

See Also

See also `MultiChainLadder`.

`MultiChainLadder` *Multivariate Chain-Ladder Models*

Description

The function `MultiChainLadder` implements multivariate methods to forecast insurance loss payments based on several cumulative claims development triangles. These methods are multivariate extensions of the chain-ladder technique, which develop several correlated triangles simultaneously in a way that both contemporaneous correlations and structural relationships can be accounted for. The estimated conditional Mean Square Errors (MSE) are also produced.

Usage

```
MultiChainLadder(Triangles, fit.method = "SUR", delta = 1,
  int = NULL, restrict.regMat = NULL, extrap = TRUE,
  mse.method = "Mack", model = "MCL", ...)

MultiChainLadder2(Triangles, mse.method = "Mack", last = 3,
  type = c("MCL", "MCL+int", "GMCL-int", "GMCL"), ...)
```

Arguments

`Triangles`	a list of cumulative claims triangles of the same dimensions.
`fit.method`	the method used to fit the multivariate regression in each development period. The default is `"SUR"` - seemingly unrelated regressions. When `"OLS"` (Ordinary Least Squares) is used, this is the same as developing each triangle separately.
`delta`	parameter for controlling weights. It is used to determine the covariance structure $D(Y_{i,k}^{\delta/2})\Sigma_k D(Y_{i,k}^{\delta/2})$. The default value `1` means that the variance is proportional to the cumulative loss from the previous period.
`int`	a numeric vector that indicates which development periods have intercepts specified. This only takes effect for `model = "GMCL"`. The default `NULL` means that no intercepts are specified.
`restrict.regMat`	a list of matrix specifying parameter restriction matrix for each period. This is only used for `model = "GMCL"`. The default value `NULL` means no restriction is imposed on the development matrix. For example, if there are 3 triangles, there will be 9 parameters in the development matrix for each period if `restrict.regMat = NULL`. See `systemfit` for how to specify the appropriate parameter constraints.
`extrap`	a logical value indicating whether to use Mack's extrapolation method for the last period to get the residual variance estimation. It only takes effect for `model = "MCL"`. If the data are trapezoids, it is set to be `FALSE` automatically and a warning message is given.
`mse.method`	method to estimate the mean square error. It can be either `"Mack"` or `"Independence"`, which are the multivariate generalization of Mack's formulas and the conditional re-sampling approach, respectively.

model	the structure of the model to be fitted. It is either "MCL" or "GMCL". See details below.
last	an integer. The MultiChainLadder2 function splits the triangles into 2 parts internally (see details below), and the last argument indicates how many of the development periods in the tail go into the second part of the split. The default is 3.
type	the type of the model structure to be specified for the first part of the split model in MultiChainLadder2. "MCL"- the multivariate chain-ladder with diagonal development matrix; "MCL+int"- the multivariate chain-ladder with additional intercepts; "GMCL-int"- the general multivariate chain-ladder without intercepts; and "GMCL" - the full general multivariate chain-ladder with intercepts and non-diagonal development matrix.
...	arguments passed to systemfit.

Details

This function implements multivariate loss reserving models within the chain-ladder framework. Two major models are included. One is the Multivariate chain-ladder (MCL) model proposed by Prohl and Schmidt (2005). This is a direct multivariate generalization of the univariate chain-ladder model in that losses from different triangles are assumed to be correlated but the mean development in one triangle only depends on its past values, not on the observed values from other triangles. In contrast, the other model, the General Multivariate chain-ladder (GMCL) model outlined in Zhang (2010), extends the MCL model by allowing development dependencies among different triangles as well as the inclusion of regression intercepts. As a result, structurally related triangles, such as the paid and incurred loss triangles or the paid loss and case reserve triangles, can be developed together while still accounting for the potential contemporaneous correlations. While the MCL model is a special case of the GMCL model, it is programmed and listed separately because: a) it is an important model for its own sake; b) different MSE methods are only available for the MCL model; c) extrapolation of the residual variance estimation can be implemented for the MCL model, which is considerably difficult for the GMCL model.

We introduce some details of the GMCL model in the following. Assume N triangles are available. Denote $Y_{i,k} = (Y_{i,k}^{(1)}, \ldots, Y_{i,k}^{(N)})$ as an $N \times 1$ vector of cumulative losses at accident year i and development year k, where (n) refers to the n-th triangle. The GMCL model in development period k (from development year k to year k+1) is:

$$Y_{i,k+1} = A_k + B_k \cdot Y_{i,k} + \epsilon_{i,k},$$

where A_k is a column of intercepts and B_k is the $N \times N$ development matrix. By default, MultiChainLadder sets A_k to be zero. This behavior can be changed by appropriately specifying the int argument. Assumptions for this model are:

$$E(\epsilon_{i,k}|Y_{i,1}, \ldots, Y_{i,I+1-k}) = 0.$$

$$cov(\epsilon_{i,k}|Y_{i,1}, \ldots, Y_{i,I+1-k}) = \Sigma_{\epsilon_{i,k}} = D(Y_{i,k}^{\delta/2})\Sigma_k D(Y_{i,k}^{\delta/2}).$$

losses of different accident years are independent.

$\epsilon_{i,k}$ are symmetrically distributed.

The GMCL model structure is generally over-parameterized. Parameter restrictions are usually necessary for the estimation to be feasible, which can be specified through the `restrict.regMat` argument. We refer the users to the documentation for `systemfit` for details and the demo of the present function for examples.

In particular, if one restricts the development matrix to be diagonal, the GMCL model will reduce to the MCL model. When non-diagonal development matrix is used and the GMCL model is applied to paid and incurred loss triangles, it can reflect the development relationship between the two triangles, as described in Quarg and Mack (2004). The full bivariate model is identical to the "double regression" model described by Mack (2003), which is argued by him to be very similar to the Munich chain-ladder (MuCL) model. The GMCL model with intercepts can also help improve model adequacy as described in Barnett and Zehnwirth (2000).

Currently, the GMCL model only works for trapezoid data, and only implements `mse.method = "Mack"`. The MCL model allows an additional mse estimation method that assumes independence among the estimated parameters. Further, the MCL model using `fit.method = "OLS"` will be equivalent to running univariate chain-ladders separately on each triangle. Indeed, when only one triangle is specified (as a list), the MCL model is equivalent to `MackChainLadder`.

The GMCL model allows different model structures to be specified in each development period. This is generally achieved through the combination of the `int` argument, which specifies the periods that have intercepts, and the `restrict.regMat` argument, which imposes parameter restrictions on the development matrix.

In using the multivariate methods, we often specify separate univariate chain-ladders for the tail periods to stabilize the estimation - there are few data points in the tail and running a multivariate model often produces extremely volatile estimates or even fails. In this case, we can use the subset operator `"["` defined for class `triangles` to split the input data into two parts. We can specify a multivariate model with rich structures on the first part to reflect the multivariate dependencies, and simply apply multiple univariate chain-ladders on the second part. The two models are subsequently joined together using the `Join2Fits` function. We can then invoke the `predict` and `Mse` methods to produce loss predictions and mean square error estimations. They can further be combined via the `JoinFitMse` function to construct an object of class `MultiChainLadder`. See the demo for such examples.

To facilitate such a split-and-join process for most applications, we have created the function `MultiChainLadder2`. This function splits the data according to the `last` argument (e.g., if `last = 3`, the last three periods go into the second part), and fits the first part according to the structure indicated in the `type` argument. See the 'Arguments' section for details.

Value

`MultiChainLadder` returns an object of class `MultiChainLadder` with the following slots:

`model`	the model structure used, either `"MCL"` or `"GMCL"`
`Triangles`	input triangles of cumulative claims that are converted to class `triangles` internally.
`models`	fitted models for each development period. This is the output from the call of `systemfit`.
`coefficients`	estimated regression coefficients or development parameters. They are put into the matrix format for the GMCL model.

coefCov	estimated variance-covariance matrix for the regression coefficients.
residCov	estimated residual covariance matrix.
fit.method	multivariate regression estimation method
delta	the value of delta
mse.ay	mean square error matrix for each accident year
mse.ay.est	estimation error matrix for each accident year
mse.ay.proc	process error matrix for each accident year
mse.total	mean square error matrix for all accident years combined
mse.total.est	estimation error matrix for all accident years combined
mse.total.proc	process error matrix for all accident years combined
FullTriangles	the forecasted full triangles of class triangles
int	intercept indicators

Note

When MultiChainLadder or MultiChainLadder2 fails, the most possible reason is that there is little or no development in the tail periods. That is, the development factor is 1 or almost equal to 1. In this case, the systemfit function may fail even for fit.method = "OLS", because the residual covariance matrix Σ_k is singular. The simplest solution is to remove these columns using the "[" operator and fit the model on the remaining part.

Also, we recommend the use of MultiChainLadder2 over MultiChainLadder. The function MultiChainLadder2 meets the need for most applications, is relatively easy to use and produces more stable but very similar results to MultiChainLadder. Use MultiChainLadder only when non-standard situation arises, e.g., when different parameter restrictions are needed for different periods. See the demo for such examples.

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

References

Buchwalder M, Bühlmann H, Merz M, Wüthrich M.V (2006). The mean square error of prediction in the chain-ladder reserving method (Mack and Murphy revisited), ASTIN Bulletin, *36(2), 521-542.*

Prohl C, Schmidt K.D (2005). Multivariate chain-ladder, Dresdner Schriften zur Versicherungsmathematik.

Mack T (1993). Distribution-free calculation of the standard error, ASTIN Bulletin, *23, No.2.*

Mack T (1999). The standard error of chain-ladder reserve estimates: recursive calculation and inclusion of a tail factor, ASTIN Bulletin, *29, No.2, 361-366.*

Merz M, Wüthrich M (2008). Prediction error of the multivariate chain ladder reserving method, North American Actuarial Journal, *12, No.2, 175-197.*

*Zhang Y (2010). A general multivariate chain-ladder model.*Insurance: Mathematics and Economics, *46, pp. 588-599.*

Zhang Y (2010). Prediction error of the general multivariate chain ladder model.

See Also

See also MackChainLadder, MunichChainLadder, triangles, MultiChainLadder, summary,MultiChainLadder-method and plot,MultiChainLadder,missing-method.

Examples

```
# This shows that the MCL model using "OLS" is equivalent to
# the MackChainLadder when applied to one triangle

data(GenIns)
(U1 <- MackChainLadder(GenIns, est.sigma = "Mack"))
(U2 <- MultiChainLadder(list(GenIns), fit.method = "OLS"))

# show plots
parold <- par(mfrow = c(2, 2))
plot(U2, which.plot = 1:4)
plot(U2, which.plot = 5)
par(parold)


# For mse.method = "Independence", the model is equivalent
# to that in Buchwalder et al. (2006)

(B1 <- MultiChainLadder(list(GenIns), fit.method = "OLS",
    mse.method = "Independence"))

# use the unbiased residual covariance estimator
# in Merz and Wuthrich (2008)
(W1 <- MultiChainLadder2(liab, mse.method = "Independence",
     control = systemfit::systemfit.control(methodResidCov = "Theil")))

## Not run:
# use the iterative residual covariance estimator
for (i in 1:5){
  W2 <- MultiChainLadder2(liab, mse.method = "Independence",
      control = systemfit::systemfit.control(
      methodResidCov = "Theil", maxiter = i))
  print(format(summary(W2)@report.summary[[3]][15, 4:5],
          digits = 6, big.mark = ","))
}

# The following fits an MCL model with intercepts for years 1:7
# and separate chain-ladder models for the rest periods
f1 <- MultiChainLadder2(auto, type = "MCL+int")

# compare with the model without intercepts through residual plots
f0 <- MultiChainLadder2(auto, type = "MCL")

parold <- par(mfrow = c(2, 3), mar = c(3, 3, 2, 1))
mt <- list(c("Personal Paid", "Personal Incured", "Commercial Paid"))
plot(f0, which.plot = 3, main = mt)
plot(f1, which.plot = 3, main = mt)
```

```
par(parold)

## summary statistics
summary(f1, portfolio = "1+3")@report.summary[[4]]


# model for joint development of paid and incurred triangles
da <- auto[1:2]
# MCL with diagonal development
M0 <- MultiChainLadder(da)
# non-diagonal development matrix with no intercepts
M1 <- MultiChainLadder2(da, type = "GMCL-int")
# Munich chain-ladder
M2 <- MunichChainLadder(da[[1]], da[[2]])
# compile results and compare projected paid to incurred ratios
r1 <- lapply(list(M0, M1), function(x){
          ult <- summary(x)@Ultimate
          ult[, 1] / ult[, 2]
      })
names(r1) <- c("MCL", "GMCL")
r2 <- summary(M2)[[1]][, 6]
r2 <- c(r2, summary(M2)[[2]][2, 3])
print(do.call(cbind, c(r1, list(MuCl = r2))) * 100, digits = 4)


## End(Not run)

# To reproduce results in Zhang (2010) and see more examples, use:
## Not run:
 demo(MultiChainLadder)

## End(Not run)
```

`MultiChainLadder-class` *Class "MultiChainLadder" of Multivariate Chain-Ladder Results*

Description

This class includes the first and second moment estimation result using the multivariate reserving methods in chain-ladder. Several primitive methods and statistical methods are also created to facilitate further analysis.

Objects from the Class

Objects can be created by calls of the form `new("MultiChainLadder", ...)`, or they could also be a result of calls from `MultiChainLadder` or `JoinFitMse`.

Slots

`model`: Object of class `"character"`. Either "MCL" or "GMCL".

`Triangles`: Object of class `"triangles"`. Input triangles.

`models`: Object of class `"list"`. Fitted regression models using `systemfit`.

`coefficients`: Object of class `"list"`. Estimated regression coefficients.

`coefCov`: Object of class `"list"`. Estimated variance-covariance matrix of coefficients.

`residCov`: Object of class `"list"`. Estimated residual covariance matrix.

`fit.method`: Object of class `"character"`. Could be values of "SUR" or "OLS".

`delta`: Object of class `"numeric"`. Parameter for weights.

`int`: Object of class `"NullNum"`. Indicator of which periods have intercepts.

`mse.ay`: Object of class `"matrix"`. Conditional mse for each accident year.

`mse.ay.est`: Object of class `"matrix"`. Conditional estimation mse for each accident year.

`mse.ay.proc`: Object of class `"matrix"`. Conditional process mse for each accident year.

`mse.total`: Object of class `"matrix"`. Conditional mse for aggregated accident years.

`mse.total.est`: Object of class `"matrix"`. Conditional estimation mse for aggregated accident years.

`mse.total.proc`: Object of class `"matrix"`. Conditional process mse for aggregated accident years.

`FullTriangles`: Object of class `"triangles"`. Completed triangles.

`restrict.regMat`: Object of class `"NullList"`

Extends

Class `"MultiChainLadderFit"`, directly. Class `"MultiChainLadderMse"`, directly.

Methods

$ `signature(x = "MultiChainLadder")`: Method for primitive function `"$"`. It extracts a slot of `x` with a specified slot name, just as in list.

[[`signature(x = "MultiChainLadder", i = "numeric", j = "missing")`: Method for primitive function `"[["`. It extracts the i-th slot of a `"MultiChainLadder"` object, just as in list. `i` could be a vector.

[[`signature(x = "MultiChainLadder", i = "character", j = "missing")`: Method for primitive function `"[["`. It extracts the slots of a `"MultiChainLadder"` object with names in `i`, just as in list. `i` could be a vector.

coef `signature(object = "MultiChainLadder")`: Method for function `coef`, to extract the estimated development matrix. The output is a list.

fitted `signature(object = "MultiChainLadder")`: Method for function `fitted`, to calculate the fitted values in the original triangles. Note that the return value is a list of fitted valued based on the original scale, not the model scale which is first divided by $Y_{i,k}^{\delta/2}$.

names `signature(x = "MultiChainLadder")`: Method for function `names`, which returns the slot names of a `"MultiChainLadder"` object.

plot `signature(x = "MultiChainLadder", y = "missing")`: See `plot,MultiChainLadder,missing-method`.

residCov `signature(object = "MultiChainLadder")`: S4 generic function and method to extract residual covariance from a `"MultiChainLadder"` object.

residCor `signature(object = "MultiChainLadder")`: S4 generic function and method to extract residual correlation from a `"MultiChainLadder"` object.

residuals `signature(object = "MultiChainLadder")`: Method for function `residuals`, to extract residuals from a system of regression equations. These residuals are based on model scale, and will not be equivalent to those on the original scale if δ is not set to be 0. One should use `rstandard` instead, which is independent of the scale.

resid `signature(object = "MultiChainLadder")`: Same as `residuals`.

rstandard `signature(model = "MultiChainLadder")`: S4 generic function and method to extract standardized residuals from a `"MultiChainLadder"` object.

show `signature(object = "MultiChainLadder")`: Method for show.

summary `signature(object = "MultiChainLadder")`: See `summary,MultiChainLadder-method`.

vcov `signature(object = "MultiChainLadder")`: Method for function `vcov`, to extract the variance-covariance matrix of a `"MultiChainLadder"` object. Note that the result is a list of `Bcov`, that is the variance-covariance matrix of the vectorized B.

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

See Also

See also `MultiChainLadder`,`summary,MultiChainLadder-method` and `plot,MultiChainLadder,missing-method`.

Examples

```
# example for class "MultiChainLadder"
data(liab)
fit.liab <-  MultiChainLadder(Triangles = liab)
fit.liab

names(fit.liab)
fit.liab[[1]]
fit.liab$model
fit.liab@model

do.call("rbind",coef(fit.liab))
vcov(fit.liab)[[1]]
residCov(fit.liab)[[1]]
head(do.call("rbind",rstandard(fit.liab)))
```

`MultiChainLadderFit-class`
Class "MultiChainLadderFit", "MCLFit" and "GMCLFit"

Description

"MultiChainLadderFit" is a virtual class for the fitted models in the multivariate chain ladder reserving framework, "MCLFit" is a result from the interal call `.FitMCL` to store results in model `MCL` and "GMCLFit" is a result from the interal call `.FitGMCL` to store results in model `GMCL`. The two classes "MCLFit" and "GMCLFit" differ only in the presentation of B_k and Σ_{B_k}, and different methods of `Mse` and `predict` will be dispatched according to these classes.

Objects from the Class

"MultiChainLadderFit" is a virtual Class: No objects may be created from it. For "MCLFit" and "GMCLFit", objects can be created by calls of the form `new("MCLFit", ...)` and `new("GMCLFit", ...)` respectively.

Slots

`Triangles`: Object of class `"triangles"`

`models`: Object of class `"list"`

`B`: Object of class `"list"`

`Bcov`: Object of class `"list"`

`ecov`: Object of class `"list"`

`fit.method`: Object of class `"character"`

`delta`: Object of class `"numeric"`

`int`: Object of class `"NullNum"`

`restrict.regMat`: Object of class `"NullList"`

Extends

"MCLFit" and "GMCLFit" extends class `"MultiChainLadderFit"`, directly.

Methods

No methods defined with class "MultiChainLadderFit" in the signature.

For "MCLFit", the following methods are defined:

`Mse signature(ModelFit = "MCLFit", FullTriangles = "triangles")`: Calculate Mse estimations.

`predict signature(object = "MCLFit")`: Predict ultimate losses and complete the triangles. The output is an object of class "triangles".

For "GMCLFit", the following methods are defined:

`Mse signature(ModelFit = "GMCLFit", FullTriangles = "triangles")`: Calculate Mse estimations.

`predict signature(object = "GMCLFit")`: Predict ultimate losses and complete the triangles. The output is an object of class "triangles".

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

See Also

See also `Mse`.

Examples

```
showClass("MultiChainLadderFit")
```

MultiChainLadderMse-class

Class "MultiChainLadderMse"

Description

This class is used to define the structure in storing the MSE results.

Objects from the Class

Objects can be created by calls of the form `new("MultiChainLadderMse", ...)`, or as a result of a call to `Mse`.

Slots

`mse.ay`: Object of class `"matrix"`

`mse.ay.est`: Object of class `"matrix"`

`mse.ay.proc`: Object of class `"matrix"`

`mse.total`: Object of class `"matrix"`

`mse.total.est`: Object of class `"matrix"`

`mse.total.proc`: Object of class `"matrix"`

`FullTriangles`: Object of class `"triangles"`

Methods

No methods defined with class "MultiChainLadderMse" in the signature.

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

See Also

See Also `MultiChainLadder` and `Mse`.

Examples

```
showClass("MultiChainLadderMse")
```

`MultiChainLadderSummary-class`
Class "MultiChainLadderSummary"

Description

This class stores the summary statistics from a "MultiChainLadder" object. These summary statistics include both model summary and report summary.

Objects from the Class

Objects can be created by calls of the form `new("MultiChainLadderSummary", ...)`, or a call from `summary`.

Slots

`Triangles`: Object of class `"triangles"`

`FullTriangles`: Object of class `"triangles"`

`S.E.Full`: Object of class `"list"`

`S.E.Est.Full`: Object of class `"list"`

`S.E.Proc.Full`: Object of class `"list"`

`Ultimate`: Object of class `"matrix"`

`IBNR`: Object of class `"matrix"`

`S.E.Ult`: Object of class `"matrix"`

`S.E.Est.Ult`: Object of class `"matrix"`

`S.E.Proc.Ult`: Object of class `"matrix"`

`report.summary`: Object of class `"list"`

`coefficients`: Object of class `"list"`

`coefCov`: Object of class `"list"`

`residCov`: Object of class `"list"`

`rstandard`: Object of class `"matrix"`

`fitted.values`: Object of class `"matrix"`

`residCor`: Object of class `"matrix"`

`model.summary`: Object of class `"matrix"`

`portfolio`: Object of class `"NullChar"`

Methods

$ `signature(x = "MultiChainLadderSummary")`: Method for primitive function `"$"`. It extracts a slot of `x` with a specified slot name, just as in list.

[[`signature(x = "MultiChainLadderSummary", i = "numeric", j = "missing")`: Method for primitive function `"[["`. It extracts the i-th slot of a `"MultiChainLadder"` object, just as in list. `i` could be a vetor.

[[`signature(x = "MultiChainLadderSummary", i = "character", j = "missing")`: Method for primitive function `"[["`. It extracts the slots of a `"MultiChainLadder"` object with names in `i`, just as in list. `i` could be a vetor.

names `signature(x = "MultiChainLadderSummary")`: Method for function `names`, which returns the slot names of a `"MultiChainLadder"` object.

show `signature(object = "MultiChainLadderSummary")`: Method for `show`.

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

See Also

See also `summary,MultiChainLadder-method`, `MultiChainLadder-class`

Examples

```
showClass("MultiChainLadderSummary")
```

MunichChainLadder *Munich-chain-ladder Model*

Description

The Munich-chain-ladder model forecasts ultimate claims based on a cumulative paid and incurred claims triangle. The model assumes that the Mack-chain-ladder model is applicable to the paid and incurred claims triangle, see `MackChainLadder`.

Usage

```
MunichChainLadder(Paid, Incurred,
                  est.sigmaP = "log-linear", est.sigmaI = "log-linear",
                  tailP=FALSE, tailI=FALSE)
```

Arguments

Paid	cumulative paid claims triangle. Assume columns are the development period, use transpose otherwise. A (mxn)-matrix P_{ik} which is filled for $k \leq n+1-i; i = 1, \ldots, m; m \geq n$
Incurred	cumulative incurred claims triangle. Assume columns are the development period, use transpose otherwise. A (mxn)-matrix I_{ik} which is filled for $k \leq n+1-i; i = 1, \ldots, m, m \geq n$
est.sigmaP	defines how $sigma_{n-1}$ for the Paid triangle is estimated, see est.sigma in MackChainLadder for more details, as est.sigmaP gets passed on to MackChainLadder
est.sigmaI	defines how $sigma_{n-1}$ for the Incurred triangle is estimated, see est.sigma in MackChainLadder for more details, as est.sigmaI is passed on to MackChainLadder
tailP	defines how the tail of the Paid triangle is estimated and is passed on to MackChainLadder, see tail just there.
tailI	defines how the tail of the Incurred triangle is estimated and is passed on to MackChainLadder, see tail just there.

Value

MunichChainLadder returns a list with the following elements

call	matched call
Paid	input paid triangle
Incurred	input incurred triangle
MCLPaid	Munich-chain-ladder forecasted full triangle on paid data
MCLIncurred	Munich-chain-ladder forecasted full triangle on incurred data
MackPaid	Mack-chain-ladder output of the paid triangle
MackIncurred	Mack-chain-ladder output of the incurred triangle
PaidResiduals	paid residuals
IncurredResiduals	incurred residuals
QResiduals	paid/incurred residuals
QinverseResiduals	incurred/paid residuals
lambdaP	dependency coefficient between paid chain-ladder age-to-age factors and incurred/paid age-to-age factors
lambdaI	dependency coefficient between incurred chain-ladder ratios and paid/incurred ratios
qinverse.f	chain-ladder-link age-to-age factors of the incurred/paid triangle
rhoP.sigma	estimated conditional deviation around the paid/incurred age-to-age factors
q.f	chain-ladder age-to-age factors of the paid/incurred triangle
rhoI.sigma	estimated conditional deviation around the incurred/paid age-to-age factors

Author(s)

Markus Gesmann <markus.gesmann@gmail.com>

References

Gerhard Quarg and Thomas Mack. Munich Chain Ladder. Blatter DGVFM *26, Munich, 2004.*

See Also

See also summary.MunichChainLadder, plot.MunichChainLadder , MackChainLadder

Examples

```
MCLpaid
MCLincurred
op <- par(mfrow=c(1,2))
plot(MCLpaid)
plot(MCLincurred)
par(op)

# Following the example in Quarg's (2004) paper:
MCL <- MunichChainLadder(MCLpaid, MCLincurred, est.sigmaP=0.1, est.sigmaI=0.1)
MCL
plot(MCL)
# You can access the standard chain-ladder (Mack) output via
MCL$MackPaid
MCL$MackIncurred

# Input triangles section 3.3.1
MCL$Paid
MCL$Incurred
# Parameters from section 3.3.2
# Standard chain-ladder age-to-age factors
MCL$MackPaid$f
MCL$MackIncurred$f
MCL$MackPaid$sigma
MCL$MackIncurred$sigma
# Check Mack's assumptions graphically
plot(MCL$MackPaid)
plot(MCL$MackIncurred)

MCL$q.f
MCL$rhoP.sigma
MCL$rhoI.sigma

MCL$PaidResiduals
MCL$IncurredResiduals

MCL$QinverseResiduals
MCL$QResiduals
```

```
MCL$lambdaP
MCL$lambdaI
# Section 3.3.3 Results
MCL$MCLPaid
MCL$MCLIncurred
```

MW2008 *Run-off claims triangle*

Description

Cumulative claims development triangle

Format

A matrix with 9 accident years and 9 development years.

Source

Modelling the claims development result for solvency purposes. Michael Merz, Mario V. Wüthrich. Casualty Actuarial Society E-Forum, Fall 2008.

Examples

```
MW2008
plot(MW2008, lattice=TRUE)
```

MW2014 *Run-off claims triangle*

Description

Cumulative claims development triangle

Format

A matrix with 17 accident years and 17 development years.

Source

Claims Run-Off Uncertainty: The Full Picture. Michael Merz, Mario V. Wüthrich. Swiss Finance Institute Research Paper No. 14-69. https://ssrn.com/abstract=2524352. *2014*

Examples

```
MW2014
plot(MW2014, lattice=TRUE)
```

NullNum-class *Class "NullNum", "NullChar" and "NullList"*

Description

Virtual class for `c("null", "numeric")`, `c("null","character"` and `c("null","list"`

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "NullNum" in the signature.

PaidIncurredChain *PaidIncurredChain*

Description

The Paid-incurred Chain model (Merz, Wuthrich (2010)) combines claims payments and incurred losses information to get a unified ultimate loss prediction.

Usage

```
PaidIncurredChain(triangleP, triangleI)
```

Arguments

`triangleP`	Cumulative claims payments triangle
`triangleI`	Incurred losses triangle.

Details

The method uses some basic properties of multivariate Gaussian distributions to obtain a mathematically rigorous and consistent model for the combination of the two information channels.

We assume as usual that I=J. The model assumptions for the Log-Normal PIC Model are the following:

- Conditionally, given $latex$ we have
  - the random vector $latex$ has multivariate Gaussian distribution with uncorrelated components given by

$$latex$$

$$latex$$

  - cumulative payments are given by the recursion

    $$latex$$

    with initial value $P_{i,0} = \exp(\xi_{i,0})$;
  - incurred losses $I_{i,j}$ are given by the backwards recursion

    $$latex$$

    with initial value $I_{i,I} = P_{i,I}$.
- The components of $latex$ are indipendent and $latex$ for all j.

Parameters $latex$ in the model are in general not known and need to be estimated from observations. They are estimated in a Bayesian framework. In the Bayesian PIC model they assume that the previous assumptions hold true with deterministic $latex$ and $latex$ and

$$latex$$

$$latex$$

This is not a full Bayesian approach but has the advantage to give analytical expressions for the posterior distributions and the prediction uncertainty.

Value

The function returns:

- **Ult.Loss.Origin** Ultimate losses for different origin years.
- **Ult.Loss** Total ultimate loss.
- **Res.Origin** Claims reserves for different origin years.
- **Res.Tot** Total reserve.
- **s.e.** Square root of mean square error of prediction for the total ultimate loss.

Note

The model is implemented in the special case of non-informative priors.

Author(s)

Fabio Concina, <fabio.concina@gmail.com>

References

Merz, M., Wuthrich, M. (2010). Paid-incurred chain claims reserving method. Insurance: Mathematics and Economics, 46(3), 568-579.

See Also

MackChainLadder,MunichChainLadder

Examples

```
PaidIncurredChain(USAApaid, USAAincurred)
```

plot-MultiChainLadder *Methods for Function plot*

Description

Methods for function plot to produce different diagonostic plots for an object of class "MultiChain-Ladder".

Usage

```
## S4 method for signature 'MultiChainLadder,missing'
plot(x, y, which.plot=1:4,
which.triangle=NULL,
main=NULL,
portfolio=NULL,
lowess=TRUE,
legend.cex=0.75,...)
```

Arguments

x	An object of class "MultiChainLadder".
y	"missing"
which.plot	This specifies which type of plot is desired. Its range is 1:5, but defaults to 1:4. "1" is the barplot of observed losses and predicted IBNR stacked and MSE predictions as error bars; "2" is a trajectory plot of the development pattern; "3" is the residual plot of standardized residuals against the fitted values; "4" is the Normal-QQ plot of the standardized residuals. "5" is the "xyplot" of development with confidence intervals for each accident year. Note that "3" and "4" are not available for portfolio.
which.triangle	This specifies which triangles are to be plotted. Default value is NULL, where all triangles plus the portfolio result will be plotted.
main	It should be a list of titles for each plot. If not supplied, use default titles.
portfolio	It specifies which triangles are to be summed as the portfolio, to be passed on to summary.
lowess	Logical. If TRUE, smoothing lines will be added on residual plots.
legend.cex	plotting parameter to be passes on to cex in legend if which.plot=1.
...	optional graphical arguments.

See Also

See also MultiChainLadder

Examples

```
## Not run:
data(liab)
fit.liab <- MultiChainLadder(liab)

# generate diagonostic plots
par(mfcol=(c(3,2)))
plot(fit.liab,which.plot=1:2)

par(mfrow=(c(2,2)))
plot(fit.liab,which.plot=3:4)

plot(fit.liab,which.triangle=1,which.plot=5)
graphics.off()

## End(Not run)
```

---

plot.BootChainLadder *Plot method for a BootChainLadder object*

Description

plot.BootChainLadder, a method to plot the output of BootChainLadder. It is designed to give a quick overview of a BootChainLadder object and to check the model assumptions.

Usage

```
## S3 method for class 'BootChainLadder'
plot(x, mfrow=NULL, title=NULL, log=FALSE,
    which=1:4, ...)
```

Arguments

x	output from BootChainLadder
mfrow	see par
title	see title
log	logical. If TRUE the y-axes of the 'latest incremental actual vs. simulated' plot will be on a log-scale
which	if a subset of the plots is required, specify a subset of the numbers 1:4.
...	optional arguments. See plot.default for more details.

Details

plot.BootChainLadder shows four graphs, starting with a histogram of the total simulated IBNRs over all origin periods, including a rug plot; a plot of the empirical cumulative distribution of the total IBNRs over all origin periods; a box-whisker plot of simulated ultimate claims costs against origin periods; and a box-whisker plot of simulated incremental claims cost for the latest available calendar period against actual incremental claims of the same period. In the last plot the simulated data should follow the same trend as the actual data, otherwise the original data might have some intrinsic trends which are not reflected in the model.

Note

The box-whisker plot of latest actual incremental claims against simulated claims follows is based on ideas from Barnett and Zehnwirth in: *Barnett and Zehnwirth. The need for diagnostic assessment of bootstrap predictive models,* Insureware technical report. *2007*

Author(s)

Markus Gesmann

See Also

See also BootChainLadder

Examples

```
B <- BootChainLadder(RAA)
plot(B)
plot(B, log=TRUE)
```

---

plot.clark *Plot Clark method residuals*

Description

Function to plot the residuals of the Clark LDF and Cape Cod methods.

Usage

```
## S3 method for class 'clark'
plot(x, ...)
```

Arguments

x	object resulting from a run of the ClarkLDF or ClarkCapeCod functions.
...	not used.

Details

If Clark's model is appropriate for the actual data, then the standardized residuals should appear as independent standard normal random variables. This function creates four plots of standardized residuals on a single page:

1. By origin
2. By age
3. By fitted value
4. Normal Q-Q plot with results of Shapiro-Wilk test

If the model is appropriate then there should not appear to be any trend in the standardized residuals or any systematic differences in the spread about the line y = 0. The Shapiro-Wilk p-value shown in the fourth plot gives an indication of how closely the standardized residuals can be considered "draws" from a standard normal random variable.

Author(s)

Daniel Murphy

References

Clark, David R., "LDF Curve-Fitting and Stochastic Reserving: A Maximum Likelihood Approach", *Casualty Actuarial Society Forum*, Fall, 2003

See Also

`ClarkLDF`, `ClarkCapeCod`

Examples

```
X <- GenIns
Y <- ClarkLDF(GenIns, maxage=Inf, G="weibull")
plot(Y)  # One obvious outlier, shapiro test flunked
X[4,4] <- NA  # remove the outlier
Z <- ClarkLDF(GenIns, maxage=Inf, G="weibull")
plot(Z)  # Q-Q plot looks good
```

---

plot.MackChainLadder *Plot method for a MackChainLadder object*

Description

`plot.MackChainLadder`, a method to plot the output of `MackChainLadder`. It is designed to give a quick overview of a `MackChainLadder` object and to check Mack's model assumptions.

Usage

```
## S3 method for class 'MackChainLadder'
plot(x, mfrow=NULL, title=NULL,
lattice=FALSE, which=1:6, ...)
```

Arguments

x	output from MackChainLadder
mfrow	see par
title	see title
lattice	logical. Default is set to FALSE and plots as described in the details section are produced. If lattice=TRUE, the function xyplot of the lattice package is used to plot developments by origin period in different panels, plus Mack's S.E.
which	if a subset of the plots is required, specify a subset of the numbers 1:6.
...	optional arguments. See plot.default for more details.

Details

plot.MackChainLadder shows six graphs, starting from the top left with a stacked bar-chart of the latest claims position plus IBNR and Mack's standard error by origin period; next right to it is a plot of the forecasted development patterns for all origin periods (numbered, starting with 1 for the oldest origin period), and 4 residual plots. The residual plots show the standardised residuals against fitted values, origin period, calendar period and development period. All residual plot should show no patterns or directions for Mack's method to be applicable. Pattern in any direction can be the result of trends and should be further investigated, see *Barnett and Zehnwirth. Best estimates for reserves. Proceedings of the CAS, LXXXVI I(167), November 2000.* for more details on trends.

Author(s)

Markus Gesmann

See Also

See Also MackChainLadder, residuals.MackChainLadder

Examples

```
plot(MackChainLadder(RAA))
```

plot.MunichChainLadder *Plot method for a MunichChainLadder object*

Description

plot.MunichChainLadder, a method to plot the output of MunichChainLadder object. It is designed to give a quick overview of a MunichChainLadder object and to check the correlation between the paid and incurred residuals.

Usage

```
## S3 method for class 'MunichChainLadder'
plot(x,  mfrow=c(2,2), title=NULL, ...)
```

Arguments

x	output from MunichChainLadder
mfrow	see par
title	see title
...	optional arguments. See plot.default for more details.

Details

plot.MunichChainLadder shows four plots, starting from the top left with a barchart of forecasted ultimate claims costs by Munich-chain-ladder (MCL) on paid and incurred data by origin period; the barchart next to it compares the ratio of forecasted ultimate claims cost on paid and incurred data based on the Mack-chain-ladder and Munich-chain-ladder methods; the two residual plots at the bottom show the correlation of (incurred/paid)-chain-ladder factors against the paid-chain-ladder factors and the correlation of (paid/incurred)-chain-ladder factors against the incurred-chain-ladder factors.

Note

The design of the plots follows those in Quarg's (2004) paper: *Gerhard Quarg and Thomas Mack. Munich Chain Ladder. Blatter DGVFM 26, Munich, 2004.*

Author(s)

Markus Gesmann

See Also

See also MunichChainLadder

Examples

```
M <- MunichChainLadder(MCLpaid, MCLincurred)
plot(M)
```

`predict.TriangleModel` *Prediction of a claims triangle*

Description

The function is internally used by `MackChainLadder` to forecast future claims.

Usage

```
## S3 method for class 'TriangleModel'
predict(object,...)
## S3 method for class 'ChainLadder'
predict(object,...)
```

Arguments

`object`	a list with two items: `Models`, `Triangle`
	`Models` list of linear models for each development period
	`Triangle` input triangle to forecast
`...`	not in use

Value

`FullTriangle`	forecasted claims triangle

Author(s)

Markus Gesmann

See Also

See also `chainladder`, `MackChainLadder`

Examples

```
RAA

CL <- chainladder(RAA)
CL
predict(CL)
```

print.ata *Print Age-to-Age factors*

Description

Function to print the results of a call to the ata function.

Usage

```
## S3 method for class 'ata'
print(x, ...)
```

Arguments

x	object resulting from a call to the ata function
...	further arguments passed to print

Details

print.ata simply prints summary.ata.

Value

A summary.ata matrix, invisibly.

Author(s)

Daniel Murphy

See Also

ata and summary.ata

Examples

```
x <- ata(GenIns)

## Print ata factors rounded to 3 decimal places, the summary.ata default
print(x)

## Round to 4 decimal places and print cells corresponding
## to future observations as blanks.
print(summary(x, digits=4), na.print="")
```

`print.clark` *Print results of Clark methods*

Description

Functions to print the results of the ClarkLDF and ClarkCapeCod methods.

Usage

```
## S3 method for class 'ClarkLDF'
print(x, Amountdigits=0, LDFdigits=3, CVdigits=3,
            row.names = FALSE, ...)

## S3 method for class 'ClarkCapeCod'
print(x, Amountdigits=0, ELRdigits=3, Gdigits=4, CVdigits=3,
            row.names = FALSE, ...)
```

Arguments

`x`	object resulting from a run of the ClarkLDF or ClarkCapeCod function.
`Amountdigits`	number of digits to display to the right of the decimal point for "amount" columns
`LDFdigits`	number of digits to display to the right of the decimal point for the loss development factor (LDF) column
`CVdigits`	number of digits to display to the right of the decimal point for the coefficient of variation (CV) column
`ELRdigits`	number of digits to display to the right of the decimal point for the expected loss ratio (ELR) column
`Gdigits`	number of digits to display to the right of the decimal point for the "growth function factor" column; default of 4 conforms with the table on pp. 67, 68 of Clark's paper
`row.names`	logical (or character vector), indicating whether (or what) row names should be printed (same as for `print.data.frame`)
`...`	further arguments passed to `print`

Details

Display the default information in "pretty format" resulting from a run of the "LDF Method" or "Cape Cod Method" – a "Development-type" exhibit for Clark's "LDF Method," a "Bornhuetter-Ferguson-type" exhibit for Clark's "Cape Cod Method."

As usual, typing the name of such an object at the console invokes its `print` method.

Value

`data.frames` whose columns are the `character` representation of their respective `summary.ClarkLDF` or `summary.ClarkCapeCod` `data.frames`.

Author(s)

Daniel Murphy

References

Clark, David R., "LDF Curve-Fitting and Stochastic Reserving: A Maximum Likelihood Approach", *Casualty Actuarial Society Forum*, Fall, 2003

See Also

`summary.ClarkLDF` and `summary.ClarkCapeCod`

Examples

```
X <- GenIns
colnames(X) <- 12*as.numeric(colnames(X))
y <- ClarkCapeCod(X, Premium=10000000+400000*0:9, maxage=240)
summary(y)
print(y)  # (or simply 'y') Same as summary(y) but with "pretty formats"

## Greater growth factors when projecting to infinite maximum age
ClarkCapeCod(X, Premium=10000000+400000*0:9, maxage=Inf)
```

---

qpaid *Quarterly run off triangle of accumulated claims data*

Description

Sample data to demonstrate how to work with triangles with a higher development period frequency than origin period frequency

Usage

```
data(qpaid); data(qincurred)
```

Format

A matrix with 12 accident years and 45 development quarters of claims costs.

Source

Made up data for testing purpose

Examples

```
dim(qpaid)
dim(qincurred)
op=par(mfrow=c(1,2))
ymax <- max(c(qpaid,qincurred),na.rm=TRUE)*1.05
matplot(t(qpaid), type="l", main="Paid development",
  xlab="Dev. quarter", ylab="$", ylim=c(0,ymax))
matplot(t(qincurred), type="l", main="Incurred development",
      xlab="Dev. quarter", ylab="$", ylim=c(0,ymax))
par(op)
## MackChainLadder expects a quadratic matrix so let's expand
## the triangle to a quarterly origin period.
n <- ncol(qpaid)
Paid <- matrix(NA, n, n)
Paid[seq(1,n,4),] <- qpaid
M <- MackChainLadder(Paid)
plot(M)

# We expand the incurred triangle in the same way
Incurred <- matrix(NA, n, n)
Incurred[seq(1,n,4),] <- qincurred

# With the expanded triangles we can apply MunichChainLadder
MunichChainLadder(Paid, Incurred)

# In the same way we can apply BootChainLadder
# We reduce the size of bootstrap replicates R
# from the default of 999 to 99 purely to reduce run time.
BootChainLadder(Paid, R=99)
```

---

quantile.MackChainLadder

quantile function for Mack-chain-ladder

Description

`quantile` methods for a `MackChainLadder` object

Usage

```
## S3 method for class 'MackChainLadder'
quantile(x, probs=c(0.75, 0.95), na.rm = FALSE,
              names = TRUE, type = 7,...)
```

Arguments

x	object of class "MackChainLadder"
probs	numeric vector of probabilities with values in [0,1], see quantile for more help
na.rm	not used
names	not used
type	not used
...	not used

Details

Reserves at the desired quantile using the Cornish-Fisher expansion.

The Cornish-Fisher expansion relies on the first three moments of the reserve risk distribution: The Best estimate resulting from the Chain-Ladder projection, the Mack standard deviation and the skewness of the distribution (for skewness estimation, see references below).

The quantile estimation requires only that the standard Mack assumptions are met.

For details of the underlying calculations, see references below.

Value

quantile.MackChainLadder gives a list with two elements back:

ByOrigin	data frame with skewness and quantile statistics by origin period
Totals	data frame with total skewness and quantile statistics across all origin periods

Author(s)

Eric Dal Moro <eric_dal_moro@yahoo.com>

References

Eric Dal Moro and Yuriy Krvavych. Probability of sufficiency of Solvency II Reserve risk margins: Practical approximations. ASTIN Bulletin, *47(3), 737-785*

Dal Moro, Eric, A Closed-Form Formula for the Skewness Estimation of Non-Life Reserve Risk Distribution (September 15, 2013). Available at SSRN: https://ssrn.com/abstract=2344297 or https://dx.doi.org/10.2139/ssrn.

See Also

See also MackChainLadder

Examples

```
M <- MackChainLadder(GenIns, est.sigma="Mack")
quantile(M, c(0.65, 0.75, 0.9))
```

QuantileIFRS17 *Quantile estimation for the IFRS 17 Risk Adjustment*

Description

The Quantile IFRS 17 function provides an estimate of the quantile attained on the reserve risk distribution that corresponds to the booked Risk Adjustment plus the Best Estimate.

Usage

```
QuantileIFRS17(MCL, Correlation, RiskMargin)
```

Arguments

MCL	a list of MackChainLadder objects
Correlation	Correlation matrix depicting the correlations between each triangle imported. The correlation matrix is of dimension n x n, with n the number of items in the list of MackChainLadder objects. For correlation estimations between P&C risks, please refer to the article of Arbenz et al. below.
RiskMargin	Input the risk margin as a single number. The risk margin corresponds to the IFRS 17 risk adjustment. It is estimated outside this function and can come from e.g. Solvency 2 standard formula. See International Actuarial Association reference below for details on risk adjustment calculations.

Details

The IFRS 17 quantile is a mandatory disclosure when producing Financial Statements under the IFRS 17 framework: Such quantile reflects the Probability of Sufficiency of the reserves defined as Best Estimate plus Risk Adjustment i.e. the probability that the reserves will cover any negative deviations up to the disclosed quantile.

When a risk measure other than the quantile measure (Value At risk) is used for determining the Risk Adjustment, the quantile has to be estimated. The purpose of this function is to provide such an estimation on deriving the first three moments of the reserve risk distribution. These moments are estimated on the triangles input into the function. These triangles are projected using chain-ladder methods and the standard Best Estimate, Mack volatility and skewness are estimated. The resulting moments of the different triangles are then aggregated using Fleishman polynomials.

On using a Cornish-Fisher expansion based on the three aggregated moments, the Probability of Sufficiency of the reserves including the Risk Adjustment (given as an input to the function) can be easily derived.

Value

QuantileIFRS17 returns a vector with the following elements

QuantileIFRS_17	Quantile attained on the reserve risk distribution with the booked Risk Adjustment

Skewness	Skewness of the overall aggregated risk distribution across all triangles
CoV	Coefficient of Variation of the overall aggregated risk distribution across all triangles
Reserve	Sum of reserves of the input MackChainLadder objects

Note

The use of Fleishman polynomials and Cornish-Fisher expansion imply that the different risks involved in the triangles inputs should be "close to normality". If the risks involved in the input triangles are far from normal distributions (e.g. extreme events, nat cats ...), the proposed framework will not apply and the quantile derived from the function will not be relevant.

Author(s)

Eric Dal Moro, Yuriy Krvavych

References

Thomas Mack. Distribution-free calculation of the standard error of chain ladder reserve estimates. Astin Bulletin. *Vol. 23. No 2. 1993. pp.213:225*

Thomas Mack. The standard error of chain ladder reserve estimates: Recursive calculation and inclusion of a tail factor. Astin Bulletin. *Vol. 29. No 2. 1999. pp.361:366*

Dal Moro, Krvavych. Probability of sufficiency of Solvency II Reserve risk margins: Practical approximations. ASTIN Bulletin, *47(3), 737-785*

P. Arbenz, D. Canestraro (2012) Estimating Copulas for Insurance from Scarce Observations, Expert Opinion and Prior Information: A Bayesian Approach, Astin Bulletin, *vol. 42(1), pages 271-290.*

International Actuarial Association (2018) Risk Adjustments for Insurance Contracts under IFRS 17

See Also

See also MackChainLadder, quantile.MackChainLadder

Examples

```
QuantileIFRS17(MCL=list(M1=MackChainLadder(RAA, est.sigma = "Mack"),
         M2=MackChainLadder(GenIns/1000, est.sigma = "Mack")),
         Correlation=matrix(c(1,0.3, 0.3, 1), ncol=2),
         RiskMargin = 20000)
```

RAA *Run off triangle of accumulated claims data*

Description

Run-off triangle of Automatic Factultative business in General Liability

Usage

```
data(RAA)
```

Format

A matrix with 10 accident years and 10 development years.

Source

Historical Loss Development, Reinsurance Association of Ammerica (RAA), ***1991***, *p.96*

References

See Also: *Which Stochastic Model is Underlying the Chain Ladder Method?*, *Thomas Mack*, Insurance Mathematics and Economics, ***15, 2/3***, *pp133-138*, *1994*

P.D.England and R.J.Verrall, *Stochastic Claims Reserving in General Insurance*, British Actuarial Journal, ***Vol. 8***, *pp443-544*, *2002*

Examples

```
RAA
plot(RAA)
plot(RAA, lattice=TRUE)
```

residCov *Generic function for residCov and residCor*

Description

`residCov` and `residCov` are a generic functions to extract residual covariance and residual correlation from a system of fitted regressions respectively.

Usage

```
residCov(object,...)
residCor(object,...)

## S4 method for signature 'MultiChainLadder'
residCov(object,...)
## S4 method for signature 'MultiChainLadder'
residCor(object,...)
```

Arguments

object	An object of class "MultiChainLadder".
...	Currently not used.

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

See Also

See also MultiChainLadder.

---

residuals.MackChainLadder

Extract residuals of a MackChainLadder model

Description

Extract residuals of a MackChainLadder model by origin-, calendar- and development period.

Usage

```
## S3 method for class 'MackChainLadder'
residuals(object, ...)
```

Arguments

object	output of MackChainLadder
...	not in use

Value

The function returns a data.frame of residuals and standardised residuals by origin-, calendar- and development period.

Author(s)

Markus Gesmann

See Also

See Also MackChainLadder

Examples

```
RAA
MCL=MackChainLadder(RAA)
MCL

residuals(MCL)
```

---

summary-methods *Methods for Function summary*

Description

Methods for function summary to calculate summary statistics from a "MultiChainLadder" object.

Usage

```
## S4 method for signature 'MultiChainLadder'
summary(object, portfolio=NULL,...)
```

Arguments

object	object of class "MultiChainLadder"
portfolio	character strings specifying which triangles to be summed up as portfolio.
...	optional arguments to summary methods

Details

summary calculations the summary statistics for each triangle and the whole portfolio from portfolio. portfolio defaults to the sum of all input triangles. It can also be specified as "i+j" format, which means the sum of the i-th and j-th triangle as portfolio. For example, "1+3" means the sum of the first and third triangle as portfolio.

Value

The summary function returns an object of class "MultiChainLadderSummary" that has the following slots:

Triangles	input triangles
FullTriangles	predicted triangles
S.E.Full	a list of prediction errors for each cell

S.E.Est.Full	a list of estimation errors for each cell
S.E.Proc.Full	a list of process errors for each cell
Ultimate	predicted ultimate losses for each triangle and portfolio
Latest	latest observed losses for each triangle and portfolio
IBNR	predicted IBNR for each triangle and portfolio
S.E.Ult	a matrix of prediction errors of ultimate losses for each triangle and portfolio
S.E.Est.Ult	a matrix of estimation errors of ultimate losses for each triangle and portfolio
S.E.Proc.Ult	a matrix of process errors of ultimate losses for each triangle and portfolio
report.summary	summary statistics for each triangle and portfolio
coefficients	estimated coefficients from systemfit. They are put into the matrix format for GMCL
coefCov	estimated variance-covariance matrix returned by systemfit
residCov	estimated residual covariance matrix returned by systemfit
rstandard	standardized residuals
fitted.values	fitted.values
residCor	residual correlation
model.summary	summary statistics for the cofficients including p-values
portfolio	how portfolio is calculated

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

See Also

See Also MultiChainLadder

Examples

```
data(GenIns)
fit.bbmw=MultiChainLadder(list(GenIns),fit.method="OLS", mse.method="Independence")
summary(fit.bbmw)
```

summary.ata *Summary method for object of class 'ata'*

Description

Summarize the age-to-age factors resulting from a call to the ata function.

Usage

```
## S3 method for class 'ata'
summary(object, digits=3, ...)
```

Arguments

object	object resulting from a call to ata
digits	integer indicating the number of decimal places for rounding the factors. The default is 3. NULL indicates that rounding should take place.
...	not used

Details

A call to ata produces a matrix of age-to-age factors with two attributes – the simple and volume weighted averages. summary.ata creates a new matrix with the averages appended as rows at the bottom.

Value

A matrix.

Author(s)

Dan Murphy

See Also

See also ata and print.ata

Examples

```
y <- ata(RAA)
summary(y, digits=4)
```

summary.BootChainLadder

Methods for BootChainLadder objects

Description

summary, print, mean, and quantile methods for BootChainLadder objects

Usage

```
## S3 method for class 'BootChainLadder'
summary(object, probs=c(0.75,0.95), ...)

## S3 method for class 'BootChainLadder'
print(x, probs=c(0.75,0.95), ...)

## S3 method for class 'BootChainLadder'
quantile(x, probs=c(0.75, 0.95), na.rm = FALSE,
              names = TRUE, type = 7,...)

## S3 method for class 'BootChainLadder'
mean(x, ...)

## S3 method for class 'BootChainLadder'
residuals(object, ...)
```

Arguments

x, object	output from BootChainLadder
probs	numeric vector of probabilities with values in [0,1], see quantile for more help
na.rm	logical; if true, any NA and NaN's are removed from 'x' before the quantiles are computed, see quantile for more help
names	logical; if true, the result has a names attribute. Set to FALSE for speedup with many 'probs', see quantile for more help
type	an integer between 1 and 9 selecting one of the nine quantile algorithms detailed below to be used, see quantile
...	further arguments passed to or from other methods

Details

print.BootChainLadder calls summary.BootChainLadder and prints a formatted version of the summary. residuals.BootChainLadder gives the residual triangle of the expected chain-ladder minus the actual triangle back.

Value

summary.BootChainLadder, mean.BootChainLadder, and quantile.BootChainLadder, give a list with two elements back:

ByOrigin	data frame with summary/mean/quantile statistics by origin period
Totals	data frame with total summary/mean/quantile statistics for all origin period

Author(s)

Markus Gesmann

See Also

See also BootChainLadder

Examples

```
B <- BootChainLadder(RAA, R=999, process.distr="gamma")
B
summary(B)
mean(B)
quantile(B, c(0.75,0.95,0.99, 0.995))
```

---

summary.clark *Summary methods for Clark objects*

Description

summary methods for ClarkLDF and ClarkCapeCod objects

Usage

```
## S3 method for class 'ClarkLDF'
summary(object, ...)

## S3 method for class 'ClarkCapeCod'
summary(object, ...)
```

Arguments

object	object resulting from a run of the ClarkLDF or ClarkCapeCod functions.
...	not currently used

Details

summary.ClarkLDF returns a data.frame that holds the columns of a typical "Development-type" exhibit.

summary.ClarkCapeCod returns a data.frame that holds the columns of a typical "Bornhuetter-Ferguson-type" exhibit.

Value

summary.ClarkLDF and summary.ClarkCapeCod return data.frames whose columns are objects of the appropriate mode (i.e., character for "Origin", otherwise numeric)

Author(s)

Dan Murphy

See Also

See also ClarkLDF

Examples

```
y <- ClarkLDF(RAA)
summary(y)
```

---

summary.MackChainLadder

Summary and print function for Mack-chain-ladder

Description

summary and print methods for a MackChainLadder object

Usage

```
## S3 method for class 'MackChainLadder'
summary(object, ...)

## S3 method for class 'MackChainLadder'
print(x, ...)
```

Arguments

x, object	object of class "MackChainLadder"
...	optional arguments to print or summary methods

Details

print.MackChainLadder calls summary.MackChainLadder and prints a formatted version of the summary.

Value

summary.MackChainLadder gives a list of two elements back

ByOrigin	data frame with Latest (latest actual claims costs), Dev.To.Date (chain-ladder development to date), Ultimate (estimated ultimate claims cost), IBNR (estimated IBNR), Mack.S.E (Mack's estimation of the standard error of the IBNR), and CV(IBNR) (Coefficient of Variance=Mack.S.E/IBNR)
Totals	data frame of totals over all origin periods. The items follow the same naming convention as in ByOrigin above

Author(s)

Markus Gesmann

See Also

See also MackChainLadder, plot.MackChainLadder

Examples

```
R <- MackChainLadder(RAA)
R
summary(R)
summary(R)$ByOrigin$Ultimate
```

---

summary.MunichChainLadder

Summary and print function for Munich-chain-ladder

Description

summary and print methods for a MunichChainLadder object

Usage

```
## S3 method for class 'MunichChainLadder'
summary(object, ...)

## S3 method for class 'MunichChainLadder'
print(x, ...)
```

Arguments

x, object	object of class "MunichChainLadder"
...	optional arguments to print or summary methods

Details

print.MunichChainLadder calls summary.MunichChainLadder and prints a formatted version of the summary.

Value

summary.MunichChainLadder gives a list of two elements back

ByOrigin	data frame with *Latest Paid* (latest actual paid claims costs), *Latest Incurred* (latest actual incurred claims position), *Latest P/I Ratio* (ratio of latest paid/incurred claims), *Ult. Paid* (estimate ultimate claims cost based on the paid triangle), *Ult. Incurred* (estimate ultimate claims cost based on the incurred triangle),*Ult. P/I Ratio* (ratio of ultimate paid forecast / ultimate incurred forecast)
Totals	data frame of totals over all origin periods. The items follow the same naming convention as in ByOrigin above

Author(s)

Markus Gesmann

See Also

See also MunichChainLadder, plot.MunichChainLadder

Examples

```
M <- MunichChainLadder(MCLpaid, MCLincurred)
M
summary(M)
summary(M)$ByOrigin
```

Table65 *Functions to Reproduce Clark's Tables*

Description

Print the tables on pages 64, 65, and 68 of Clark's paper.

Usage

```
Table64(x)
Table65(x)
Table68(x)
```

Arguments

x	an object resulting from ClarkLDF or ClarkCapeCod

Details

These exhibits give some of the details behind the calculations producing the estimates of future values (a.k.a. "Reserves" in Clark's paper). Table65 works for both the "LDF" and the "CapeCod" methods. Table64 is specific to "LDF", Table68 to "CapeCod".

Value

A data.frame.

Author(s)

Daniel Murphy

References

Clark, David R., "LDF Curve-Fitting and Stochastic Reserving: A Maximum Likelihood Approach", *Casualty Actuarial Society Forum*, Fall, 2003 https://www.casact.org/pubs/forum/03fforum/03ff041.pdf

Examples

```
Table65(ClarkLDF(GenIns, maxage=20))
Table64(ClarkLDF(GenIns, maxage=20))

X <- GenIns
colnames(X) <- 12*as.numeric(colnames(X))
Table65(ClarkCapeCod(X, Premium=10000000+400000*0:9, maxage=Inf))
Table68(ClarkCapeCod(X, Premium=10000000+400000*0:9, maxage=Inf))
```

---

triangle S3 Methods *Generic functions for triangles*

Description

Functions to ease the work with triangle shaped matrix data. A 'triangle' is a matrix with some generic functions.

triangle creates a triangle from the given set of vectors of observed data.

as.triangle attempts to turn its argument into a triangle. Triangles are usually stored in a "long" format in data bases. The function can transform a data.frame into a triangle shape.

as.data.frame turns a triangle into a data frame.

Usage

```
triangle(..., bycol=FALSE, origin="origin", dev="dev", value="value")

## S3 method for class 'matrix'
as.triangle(Triangle, origin="origin", dev="dev", value="value", ...)
## S3 method for class 'data.frame'
as.triangle(Triangle, origin="origin", dev="dev", value="value", ...)
## S3 method for class 'triangle'
as.data.frame(x, row.names=NULL, optional, lob=NULL, na.rm=FALSE, ...)
as.triangle(Triangle, origin="origin", dev="dev", value="value", ...)
## S3 method for class 'triangle'
plot(x, type = "b", xlab = "dev. period", ylab = NULL, lattice=FALSE, ...)
```

Arguments

Triangle	a triangle
bycol	logical. If FALSE (the default) the triangle is filled by rows, otherwise the triangle is filled by columns.
origin	name of the origin period, default is "origin".
dev	name of the development period, default is "dev".
value	name of the value, default is "value".
row.names	default is set to NULL and will merge origin and dev. period to create row names.
lob	default is NULL. The idea is to use lob (line of business) as an additional column to label a triangle in a long format, see the examples for more details.
optional	not used
na.rm	logical. Remove missing values?
x	a matrix of class 'triangle'
xlab	a label for the x axis, defaults to 'dev. period'
ylab	a label for the y axis, defaults to NULL
lattice	logical. If FALSE the function matplot is used to plot the developments of the triangle in one graph, otherwise the xyplot function of the lattice package is used, to plot developments of each origin period in a different panel.
type	type, see plot.default
...	vectors of data in triangle, see details; arguments to be passed to other methods everywhere else.

Details

Function triangle builds a triangle matrix from the vectors of *known* data provided in .... Normally, each of these vectors should be one shorter than the preceeding one. The length of the first vector dictates the number of development periods or origin periods (respectively when bycol is FALSE or TRUE). As a special case, the function will build an $n \times n$ triangle from a single vector of $n(n+1)/2$ data points.

The names of the arguments in ... for function triangle (when there are more than one) are retained for row/column names. Similarly, the names of the elements of the *first* argument are used as column/row names.

Warning

Please note that for the function `as.triangle` the origin and dev. period columns have to be of type numeric or a character which can be converted into numeric.

Also note that when converting from a data.frame to a matrix with `as.triangle`, multiple records with the same `origin` and `dev` will be aggregated.

Author(s)

Markus Gesmann, Dan Murphy, Vincent Goulet

Examples

```
GenIns
plot(GenIns)
plot(GenIns, lattice=TRUE)


## Convert long format into triangle
## Triangles are usually stored as 'long' tables in data bases
head(GenInsLong)
as.triangle(GenInsLong, origin="accyear", dev="devyear", "incurred claims")

X <- as.data.frame(RAA)
head(X)

Y <- as.data.frame(RAA, lob="General Liability")
head(Y)

## Basic creation of a triangle from loss development data
triangle(c(100, 150, 175, 180, 200),
         c(110, 168, 192, 205),
         c(115, 169, 202),
         c(125, 185),
         150)

## Same, with named origin periods
triangle("2012" = c(100, 150, 175, 180, 200),
         "2013" = c(110, 168, 192, 205),
         "2014" = c(115, 169, 202),
         "2015" = c(125, 185),
         "2016" = 150)

## Again, with also named development periods
triangle("2012" = c("12 months" = 100,
                    "24 months" = 150,
                    "36 months" = 175,
                    "48 months" = 180,
                    "60 months" = 200),
         "2013" = c(110, 168, 192, 205),
         "2014" = c(115, 169, 202),
         "2015" = c(125, 185),
```

```
          "2016" = 150)

## Quick, simplified usage
triangle(c(100, 150, 175, 110, 168, 115))
```

`triangles-class` *S4 Class "triangles"*

Description

This is a S4 class that has "list" in the data part. This class is created to facilitate validation and extraction of data.

Objects from the Class

Objects can be created by calls of the form `new("triangles", ...)`, or use `as(...,"triangles")`, where `...` is a "list".

Slots

`.Data`**:** Object of class `"list"`

Extends

Class `"list"`, from data part. Class `"vector"`, by class "list", distance 2.

Methods

Mse `signature(ModelFit = "GMCLFit", FullTriangles = "triangles")`: See `Mse`

Mse `signature(ModelFit = "MCLFit", FullTriangles = "triangles")`: See `Mse`

[`signature(x = "triangles", i = "missing", j = "numeric", drop = "logical")`: Method for primitive function "[" to subset certain columns. If `drop=TRUE`, rows composed of all "NA"s are removed. Dimensions are not dropped.

[`signature(x = "triangles", i = "missing", j = "numeric", drop = "missing")`: Method for primitive function "[" to subset certain columns, where rows composed of all "NA"s are removed. Dimensions are not dropped.

[`signature(x = "triangles", i = "numeric", j = "missing", drop = "logical")`: Method for primitive function "[" to subset certain rows. If `drop=TRUE`, columns composed of all "NA"s are removed. Dimensions are not dropped.

[`signature(x = "triangles", i = "numeric", j = "missing", drop = "missing")`: Method for primitive function "[" to subset certain rows, where columns composed of all "NA"s are removed. Dimensions are not dropped.

[`signature(x = "triangles", i = "numeric", j = "numeric", drop = "missing")`: Method for primitive function "[" to subset certain rows and columns. Dimensions are not dropped.

[<- `signature(x = "triangles", i = "numeric", j = "numeric", value = "list")`: Method for primitive function "[<-" to replace one cell in each triangle with values specified in `value`.

coerce `signature(from = "list", to = "triangles")`: Method to construct a "triangles" object from "list".

dim `signature(x = "triangles")`: Method to get the dimensions. The return value is a vector of length 3, where the first element is the number of triangles, the sencond is the number of accident years, and the third is the number of development years.

cbind2 `signature(x = "triangles", y="missing")`: Method to column bind all triangles using `cbind` internally.

rbind2 `signature(x = "triangles", y="missing")`: Method to row bind all triangles using `rbind` internally.

Author(s)

Wayne Zhang <actuary_zhang@hotmail.com>

See Also

See also `MultiChainLadder`

Examples

```
data(auto)

# "coerce"
auto <- as(auto,"triangles")  # transform "list" to be "triangles"

# method for "["
auto[,4:6,drop=FALSE] # rows of all NA's not dropped
auto[,4:6] # drop rows of all NA's

auto[8:10, ,drop=FALSE] #columns of all NA's not dropped
auto[8:10, ]  #columns of all NA's  dropped

auto[1:2,1]

# replacement method
auto[1:2,1] <- list(1,2,3)
auto[1,2]

dim(auto)

cbind2(auto[1:2,1])
rbind2(auto[1:2,1])
```

`tweedieReserve` *Tweedie Stochastic Reserving Model*

Description

This function implements loss reserving models within the generalized linear model framework in order to generate the full predictive distribution for loss reserves. Besides, it generates also the one year risk view useful to derive the reserve risk capital in a Solvency II framework. Finally, it allows the user to validate the model error while changing different model parameters, as the regression structure and diagnostics on the Tweedie p parameter.

Usage

```
tweedieReserve(triangle, var.power = 1,
                    link.power = 0, design.type = c(1, 1, 0),
                    rereserving = FALSE, cum = TRUE, exposure = FALSE,
                    bootstrap = 1, boot.adj = 0, nsim = 1000,
                    proc.err = TRUE, p.optim = FALSE,
                    p.check = c(0, seq(1.1, 2.1, by = 0.1), 3),
                    progressBar = TRUE, ...)
```

Arguments

`triangle`	An object of class `triangle`.
`var.power`	The index (p) of the power variance function $V(\mu) = \mu^p$. Default to `p = 1`, which is the over-dispersed Poisson model. If `NULL`, it will be assumed to be in `(1, 2)` and estimated using the `cplm` package. See `tweedie`.
`link.power`	The index of power link function. The default `link.power = 0` produces a log link. See `tweedie`.
`design.type`	It's a 3 dimension array that specifies the design matrix underlying the GLM. The dimensions represent respectively: origin period, development and calendar period. Accepted values are: `0` (not modelled), `1` (modelled as factor) and `2` (modelled as variable). Default to `c(1,1,0)`, which is the common specification in actuarial literature (origin and development period as factors, calendar period not modelled). If a parameter for the calendar period is specified, a linear regression on the log CY parameter is fitted to estimate future values, thus is recommended to validate them running a plot of the gamma values (see output `gamma_y`) .
`rereserving`	Boolean, if `TRUE` the one year risk view loss reserve distribution is derived. Default to `FALSE`. Note, the runtime can materially increase if set to `TRUE`.
`cum`	Boolean, indicating whether the input triangle is cumulative or incremental along the development period. If `TRUE`, then `triangle` is assumed to be on the cumulative scale, and it will be converted to incremental losses internally before a GLM is fitted.
`exposure`	Boolean, if `TRUE` the `exposure` defined in the `triangle` object is specified as `offset` in the GLM model specification. Default to `FALSE`.

`bootstrap` Integer, it specifies the type of bootstrap for parameter error. Accepted values are: `0` (disabled), `1` (parametric), `2` (semi-parametric). Default to `1`.

`boot.adj` Integer, it specified the methodology when using semi-parametric bootstrapping. Accepted values are: `0` (cycles until all the values of the pseudo-triangle are >= 0), `1` (overwrite negative values to 0.01). Default to `0`. Note, runtime can materially increase when set to `0`, as it could struggle to find pseudo-triangles >= 0)

`nsim` Integer, number of simulations to derive the loss reserve distribution. Default to `1000`. Note, high num of simulations could materially increase runtime, in particular if a re-reserving algorithm is used as well.

`proc.err` Boolean, if `TRUE` a process error (coherent with the specified model) is added to the forecasted distribution. Default to `TRUE`.

`p.optim` Boolean, if `TRUE` the model estimates the MLE for the Tweedie's `p` parameter. Default to `FALSE`. Recommended to use to validate the Tweedie's `p` parameter.

`p.check` If `p.optim=TRUE`, a vector of `p` values for consideration. The values must all be larger than one (if the response variable has exact zeros, the values must all be between one and two). Default to `c(0,seq(1.1,2.1,by=0.1),3)`. As fitting the Tweedie p-value isn't a straightforward process, please refer to `tweedie.profile`, `p.vec` argument.

`progressBar` Boolean, if `TRUE` a progress bar will be shown in the console to give an indication of bootstrap progress.

`...` Arguments to be passed onto the function `glm` or `cpglm` such as `contrasts` or `control`. It is important that `offset` and `weight` should not be specified. Otherwise, an error will be reported and the program will quit.

Value

The output is an object of class `"glm"` that has the following components:

`call` the matched call.

`summary` A data frame containing the predicted loss reserve statistics. The following items are displayed:

- `Latest`: Latest paid
- `Det.Reserve`: Deterministic reserve, i.e. the MLE GLM estimate of the Reserve
- `Ultimate`: Ultimate cost, defined as `Latest+Det.Reserve`
- `Dev.To.Date`: Development to date, defined as `Latest/Ultimate`

The following items are available if `bootstrap>0`

- `Expected.Reserve`: The expected reserve, defined as the average of the reserve simulations. Should be roughly as `Det.Reserve`.
- `Prediction.Error`: The prediction error of the reserve, defined as sqrt of the simulations. Please note that if `proc.err=FALSE`, this field contains only the parameter error given by the bootstrap.
- `CoV`: Coefficient of Variation, defined as `Prediction. Error/Expected.Reserve`.
- `Expected Ultimate`: The expected ultimate, defined as `Expected.Reserve+Latest`.

The following items are availbale if `bootstrap>0 & reserving=TRUE`

- `Expected.Reserve_1yr`: The reserve derived as sum of next year payment and the expected value of the re-reserve at the end of the year. It should be similar to both `Expected.Reserve` and `Det.Reserve`. If it isn't, it's recommended to change regression structure and parameters.
- `Prediction.Error_1yr`: The prediction error of the prospective Claims Development Result (CDR), as defined by Wüthrich (CDR=R(0)-X-R(1)).
- `Emergence.Pattern`: It's the emergence pattern defined as `Prediction.Error_1yr/Prediction.E`

`Triangle` The input triangle.

`FullTriangle` The completed triangle, where empty cells in the original triangle are filled with model predictions.

`model` The fitted GLM, a class of `glm` or `cpglm`. It is most convenient to work with this component when model fit information is wanted.

`scale` The dispersion parameter phi

`bias` The model bias, defined as `bias<-sqrt(n/d.f)`

`GLMReserve` Deterministic reserve, i.e. the MLE GLM estimate of the Reserve

`gamma_y` When the calendar year is used, it displays the observed and fitted calendar year (usually called "gamma"") factors.

`res.diag` It's a data frame for residual diagnostics. It contains:

- `unscaled`: The GLM Pearson residuals.
- `unscaled.biasadj`: The GLM Person residuals adjusted by the bias, i.e. `unscaled.biasadj=unscaled*bias`.
- `scaled`: The GLM Person scaled residuals, i.e. `scaled=unscaled/sqrt(phi)`.
- `scaled, biasadj`: The GLM Person scaled residuals adjusted by the bias, i.e. `scaled.biasadj=scaled*bias`.
- `dev`: Development year.
- `origin`: Origin year.
- `cy`: Calendar year.

[If `boostrap>1`]

`distr.res_ult` The full distribution "Ultimate View"

[If `rereserve=TRUE`]

`distr.res_1yr` The full distribution "1yr View"

Warning

Note that the runtime can materially increase for certain parameter setting. See above for more details.

Note

This function was born initially as a fork of the `glmReserve` by Wayne Zhang. I would like to thank him for his work that permitted me to speed up my coding.

Author(s)

Alessandro Carrato MSc FIA OA <alessandro.carrato@gmail.com>

References

Gigante, Sigalotti. *Model risk in claims reserving with generalized linear models*. Giornale dell'Istituto Italiano degli Attuari, Volume LXVIII. 55-87. 2005

England, Verrall. *Stochastic claims reserving in general insurance*. B.A.J. 8, III. 443-544. 2002

England, Verrall. *Predictive distributions of outstanding liabilities in general insurance*. A.A.S. 1, II. 221-270. 2006

Peters, Shevchenko, Wüthrich, *Model uncertainty in claims reserving within Tweedie's compound poisson models*. Astin Bulletin 39(1). 1-33. 2009

Renshaw, Verrall. *A stochastic model underlying the chain-ladder technique*. B.A.J. 4, IV. 903-923. 1998

See Also

See also summary.tweedieReserve.

Examples

```
## Not run:
## Verrall's ODP Model is a Tweedie with p=1, log link and
## origin/development periods as factors, thus c(1,1,0)
res1 <- tweedieReserve(MW2008, var.power=1, link.power=0,
                           design.type=c(1,1,0), rereserving=TRUE,
                           progressBar=TRUE)

## To get directly ultimate view and respective one year view
## at selected percentiles
summary(res1)

#To get other interesting statistics
res1$summary

## In order to validate the Tweedie parameter 'p', it is interesting to
## review its loglikelihood profile. Please note that, given the nature
## of our data, it is expected that we may have some fitting issues for
## given 'p' parameters, thus any results/errors should be considered
## only indicatively. Considering different regression structures is anyway
## recommended. Different 'p' values can be defined via the p.check array
## as input of the function.
## See help(tweedie.profile), p.vec parameter, for further information.
## Note: The parameters rereserving and bootstrap can be set to 0 to speed up
## the process, as they aren't needed.

## Runs a 'p' loglikelihood profile on the parameters
## p=c(0,1.1,1.2,1.3,1.4,1.5,2,3)
res2 <- tweedieReserve(MW2008, p.optim=TRUE,
                       p.check=c(0,1.1,1.2,1.3,1.4,1.5,2,3),
```

```
                        design.type=c(1,1,0),
                         rereserving=FALSE, bootstrap=0,
                         progressBar=FALSE)

## As it is possible to see in this example, the MLE of p (or xi) results
## between 0 and 1, which is not possible as Tweedie models aren't
## defined for 0 < p < 1, thus the Error message.
## But, despite this, we can conclude that overall the value p=1 could be
## reasonable for this dataset, as anyway it seems to be near the MLE.

## In order to consider an inflation parameter across the origin period,
## it may be interesting to change the regression structure to c(0,1,1)
## to get the same estimates of the Arithmetic Separation Method, as
## referred in Gigante/Sigalotti.
res3 <- tweedieReserve(MW2008, var.power=1, link.power=0,
                           design.type=c(0,1,1), rereserving=TRUE,
                           progressBar=TRUE)
res3

## An assessment on future fitted calendar year values (usually defined
## as "gamma") is recommended
plot(res3$gamma_y)

## Model residuals can be plotted using the res.diag output
plot(scaled.biasadj ~ dev, data=res3$res.diag) # Development year
plot(scaled.biasadj ~ cy, data=res3$res.diag) # Calendar year
plot(scaled.biasadj ~ origin, data=res3$res.diag) # Origin year

## End(Not run)
```

`tweedieReserve methods`

Reserve Risk Capital Report

Description

Main purpose of this function is to create a report to assess the reserve risk capital given an object of the `tweedieReserve` class. It displays both the ultimate and one year risk views at given percentiles.

Usage

```
## S3 method for class 'tweedieReserve'
print(x, ...)
## S3 method for class 'tweedieReserve'
summary(object, q = c(0.5, 0.75, 0.9, 0.95, 0.995),...)
```

Arguments

`x`	An object of class `tweedieReserve`.

object	An object of class tweedieReserve.
q	Array of percentiles to be displayed.
...	Not used

Value

A list with two items

Predicton	a data.frame with ultimate view reserve risk and the one year view reserve risk at the given percentiles.
Diagnostic	Quick diagnostic to show the deterministic reserve vs ultimate view and one year view best estimate. If the model is working properly, then these three value shouldn't be much different.

Author(s)

Alessandro Carrato MSc FIA OA <alessandro.carrato@gmail.com>

See Also

See also tweedieReserve.

Examples

```
## Not run:
tw <- tweedieReserve(MW2008, rereserving = TRUE)
summary(tw)
# For comparison
CDR.BootChainLadder(BootChainLadder(MW2008))

## End(Not run)
```

UKMotor *UK motor claims triangle*

Description

Triangle of cumulative claims payments for four origin (accident) years over time (development years).

Usage

```
data("UKMotor")
```

Format

A matrix with 7 accident years and 7 development years.

Source

https://www.actuaries.org.uk/research-and-resources/documents/claims-reserving-manual-vol2-section-

References

Stavros Christofides. Regression models based on log-incremental payments. Claims Reserving Manual. Volume 2 D5. September 1997

Examples

```
data(UKMotor)
plot(UKMotor)
MackChainLadder(UKMotor, est.sigma="Mack")
```

USAA triangle *Example paid and incurred triangle data from CAS web site.*

Description

Paid and incurred triangle data from the United Services Automobile Association company for the private passenger auto liability/medical line of business.

Usage

```
data("USAApaid")
```

Format

A triangle with 10 accident years and 10 development years.

Details

The claims data comes from Schedule P - Analysis of Losses and Loss Expenses in the National Association of Insurance Commissioners (NAIC) database. CAS obtained permission from the NAIC to make this data available to all interested researchers on the CAS website. NAIC Schedule P contains information on claims for major personal and commercial lines for all property-casualty insurers that write business in US.

Source

https://www.casact.org/research/index.cfm?fa=loss_reserves_data

References

CAS website.

Examples

```
data(USAApaid)
```

vcov.clark *Covariance Matrix of Parameter Estimates – Clark's methods*

Description

Function to compute the covariance matrix of the parameter estimates for the ClarkLDF and ClarkCapeCod methods.

Usage

```
## S3 method for class 'clark'
vcov(object, ...)
```

Arguments

object	object resulting from a run of the ClarkLDF or ClarkCapeCod functions.
...	not used.

Details

The covariance matrix of the estimated parameters is estimated by the inverse of the Information matrix (see Clark, p. 53). This function uses the "FI" and "sigma2" values returned by ClarkLDF and by ClarkCapeCod and calculates the matrix
-sigma2*FI^-1.

Author(s)

Daniel Murphy

References

Clark, David R., "LDF Curve-Fitting and Stochastic Reserving: A Maximum Likelihood Approach", *Casualty Actuarial Society Forum*, Fall, 2003

See Also

ClarkLDF, ClarkCapeCod

Examples

```
x <- GenIns
colnames(x) <- 12*as.numeric(colnames(x))
Y <- ClarkCapeCod(x, Premium=10000000+400000*0:9, maxage=240)
round(vcov(Y),6) ## Compare to matrix on p. 69 of Clark's paper

# The estimates of the loglogistic parameters
Y$THETAG
```

```
# The standard errors of the estimated parameters
sqrt(tail(diag(vcov(Y)), 2))

# The parameter risks of the estimated reserves are calculated
# according to the formula on p. 54 of Clark's paper. For example, for
# the 5th accident year, pre- and post-multiply the covariance matrix
# by a matrix consisting of the gradient entries for just that accident year
FVgrad5 <- matrix(Y$FutureValueGradient[, 5], ncol=1)
sqrt(t(FVgrad5) %*% vcov(Y) %*% FVgrad5) ## compares to 314,829 in Clark's paper

# The estimated reserves for accident year 5:
Y$FutureValue[5]   ## compares to 2,046,646 in the paper

# Recalculate the parameter risk CV for all accident years in total (10.6% in paper):
sqrt(sum(t(Y$FutureValueGradient) %*% vcov(Y) %*% Y$FutureValueGradient)) /
    Y$Total$FutureValue
```

Index

∗Topic **aplot**
plot.BootChainLadder, 60
plot.clark, 61
plot.MackChainLadder, 62
plot.MunichChainLadder, 64
∗Topic **array**
Cumulative and incremental triangles, 24
∗Topic **classes**
MultiChainLadder-class, 47
MultiChainLadderFit-class, 50
MultiChainLadderMse-class, 51
MultiChainLadderSummary-class, 52
NullNum-class, 57
triangles-class, 86
∗Topic **datasets**
ABC, 4
auto, 7
GenIns, 25
liab, 32
M3IR5, 34
MCLpaid, 39
Mortgage, 39
MW2008, 56
MW2014, 56
qpaid, 68
RAA, 73
UKMotor, 93
USAA triangle, 94
∗Topic **methods**
getLatestCumulative, 26
Mse-methods, 40
plot-MultiChainLadder, 59
plot.clark, 61
plot.MackChainLadder, 62
print.clark, 67
quantile.MackChainLadder, 69
residCov, 73
summary-methods, 75
summary.BootChainLadder, 78
summary.clark, 79
summary.MackChainLadder, 80
summary.MunichChainLadder, 81
Table65, 82
triangle S3 Methods, 83
∗Topic **models**
BootChainLadder, 8
CDR, 10
chainladder, 11
ClarkCapeCod, 14
ClarkLDF, 18
CLFMdelta, 21
coef.ChainLadder, 23
glmReserve, 27
Join2Fits, 31
JoinFitMse, 32
LRfunction, 33
MackChainLadder, 34
Mse-methods, 40
MultiChainLadder, 42
MunichChainLadder, 53
predict.TriangleModel, 65
residuals.MackChainLadder, 74
tweedieReserve, 88
∗Topic **package**
ChainLadder-package, 3
∗Topic **print**
print.ata, 66
print.clark, 67
summary.BootChainLadder, 78
summary.MackChainLadder, 80
summary.MunichChainLadder, 81
tweedieReserve methods, 92
∗Topic **summary**
summary.ata, 77
summary.clark, 79
[,triangles,missing,numeric,logical-method *(*triangles-class*)*, 86

[,triangles,missing,numeric,missing-method (triangles-class), 86
[,triangles,numeric,missing,logical-method (triangles-class), 86
[,triangles,numeric,missing,missing-method (triangles-class), 86
[,triangles,numeric,numeric,missing-method (triangles-class), 86
[<-,triangles,numeric,numeric,list-method (triangles-class), 86
[[,MultiChainLadder,character,missing-method (MultiChainLadder-class), 47
[[,MultiChainLadder,numeric,missing-method (MultiChainLadder-class), 47
[[,MultiChainLadderSummary,character,missing-method (MultiChainLadderSummary-class), 52
[[,MultiChainLadderSummary,numeric,missing-method (MultiChainLadderSummary-class), 52
$,MultiChainLadder-method (MultiChainLadder-class), 47
$,MultiChainLadderSummary-method (MultiChainLadderSummary-class), 52

ABC, 4
as.data.frame.triangle (triangle S3 Methods), 83
as.LongTriangle, 5
as.triangle, *24*, *27*
as.triangle (triangle S3 Methods), 83
ata, 6, *13*, *66*, *77*
auto, 7

bcplm, *28*
BootChainLadder, 8, *10, 11*, *60, 61*, *78, 79*

cbind2,triangles,missing-method (triangles-class), 86
CDR, 10
CDR.BootChainLadder, *9*
CDR.MackChainLadder, *37*
ChainLadder (ChainLadder-package), 3
chainladder, *7*, 11, *22, 23*, *35*, *37*, *65*
ChainLadder-package, 3
ClarkCapeCod, 14, *20*, *29*, *62*, *79*, *95*
ClarkLDF, *17*, 18, *29*, *62*, *79, 80*, *95*
CLFMdelta, 21
coef,MultiChainLadder-method (MultiChainLadder-class), 47
coef.ChainLadder, 23
coerce,list,triangles-method (triangles-class), 86
cpglm, *28*, *30*
cum2incr (Cumulative and incremental triangles), 24
Cumulative and incremental triangles, 24

dim,triangles-method (triangles-class), 86

family, *28*
fitted,MultiChainLadder-method (MultiChainLadder-class), 47

GenIns, 25
GenInsLong (GenIns), 25
getLatestCumulative, 26
glm, *28*, *30*, *89*
glm.nb, *30*
glmReserve, 27
GMCLFit-class (MultiChainLadderFit-class), 50

incr2cum (Cumulative and incremental triangles), 24

Join2Fits, 31
JoinFitMse, 32

liab, 32
list, *86*
LRfunction, 33

M3IR5, 34
MackChainLadder, *10–13*, *23*, *30*, 34, *46*, *53–55*, *58*, *62, 63*, *65*, *70*, *72*, *74, 75*, *81*
matplot, *84*
MCLFit-class (MultiChainLadderFit-class), 50
MCLincurred (MCLpaid), 39
MCLpaid, 39
mean.BootChainLadder (summary.BootChainLadder), 78
Mortgage, 39
Mse, *51, 52*, *86*

Mse (Mse-methods), 40
Mse,GMCLFit,triangles-method (Mse-methods), 40
Mse,MCLFit,triangles-method (Mse-methods), 40
Mse-methods, 40
MultiChainLadder, *31*, *32*, *41*, 42, *46*, *49*, *52*, *59*, *74*, *76*, *87*
MultiChainLadder-class, 47
MultiChainLadder2 (MultiChainLadder), 42
MultiChainLadderFit, *48*, *50*
MultiChainLadderFit-class, 50
MultiChainLadderMse, *48*
MultiChainLadderMse-class, 51
MultiChainLadderSummary-class, 52
MunichChainLadder, *46*, 53, *58*, *64*, *82*
MW2008, 56
MW2014, 56

names,MultiChainLadder-method (MultiChainLadder-class), 47
names,MultiChainLadderSummary-method (MultiChainLadderSummary-class), 52
NullChar-class (NullNum-class), 57
NullList-class (NullNum-class), 57
NullNum-class, 57

PaidIncurredChain, 57
par, *60*, *63*, *64*
plot,MultiChainLadder,missing-method (plot-MultiChainLadder), 59
plot-methods (plot-MultiChainLadder), 59
plot-MultiChainLadder, 59
plot.BootChainLadder, *9*, 60
plot.clark, *15*, *19*, 61
plot.default, *60*, *63*, *64*, *84*
plot.glmReserve (glmReserve), 27
plot.MackChainLadder, *37*, 62, *81*
plot.MunichChainLadder, *55*, 64, *82*
plot.triangle (triangle S3 Methods), 83
predict,GMCLFit-method (MultiChainLadderFit-class), 50
predict,MCLFit-method (MultiChainLadderFit-class), 50
predict.ChainLadder, *13*
predict.ChainLadder (predict.TriangleModel), 65
predict.TriangleModel, 65

print.ata, *7*, 66, *77*
print.BootChainLadder (summary.BootChainLadder), 78
print.clark, 67
print.ClarkCapeCod (print.clark), 67
print.ClarkLDF (print.clark), 67
print.data.frame, *67*
print.glmReserve (glmReserve), 27
print.MackChainLadder (summary.MackChainLadder), 80
print.MunichChainLadder (summary.MunichChainLadder), 81
print.tweedieReserve (tweedieReserve methods), 92

qincurred (qpaid), 68
qpaid, *8*, *12*, *21*, *35*, *37*, 68
quantile, *70*, *78*
quantile.BootChainLadder (summary.BootChainLadder), 78
quantile.MackChainLadder, *37*, 69, *72*
QuantileIFRS17, 71

RAA, 73
rbind2,triangles,missing-method (triangles-class), 86
resid,MultiChainLadder-method (MultiChainLadder-class), 47
resid.glmReserve (glmReserve), 27
residCor (residCov), 73
residCor,MultiChainLadder-method (residCov), 73
residCor-methods (residCov), 73
residCov, 73
residCov,MultiChainLadder-method (residCov), 73
residCov-methods (residCov), 73
residuals,MultiChainLadder-method (MultiChainLadder-class), 47
residuals.BootChainLadder (summary.BootChainLadder), 78
residuals.MackChainLadder, *37*, *63*, 74
rstandard,MultiChainLadder-method (MultiChainLadder-class), 47

show,MultiChainLadder-method (MultiChainLadder-class), 47
show,MultiChainLadderSummary-method (MultiChainLadderSummary-class), 52

summary,MultiChainLadder-method
(summary-methods), 75
summary-methods, 75
summary.ata, *7*, *66*, 77
summary.BootChainLadder, *9*, 78
summary.clark, 79
summary.ClarkCapeCod, *68*
summary.ClarkCapeCod (summary.clark), 79
summary.ClarkLDF, *68*
summary.ClarkLDF (summary.clark), 79
summary.glmReserve (glmReserve), 27
summary.MackChainLadder, *37*, 80
summary.MunichChainLadder, *55*, 81
summary.tweedieReserve, *91*
summary.tweedieReserve (tweedieReserve
methods), 92
systemfit, *44*

Table64 (Table65), 82
Table65, 82
Table68 (Table65), 82
title, *60*, *63*, *64*
triangle, *27*, *88*
triangle (triangle S3 Methods), 83
triangle S3 Methods, 83
triangles, *46*
triangles-class, 86
tweedie, *27*, *28*, *30*, *88*
tweedie.profile, *89*
tweedieReserve, 88, *92*, *93*
tweedieReserve methods, 92

UKMotor, 93
USAA triangle, 94
USAAincurred (USAA triangle), 94
USAApaid (USAA triangle), 94

vcov,MultiChainLadder-method
(MultiChainLadder-class), 47
vcov.clark, 95
vector, *86*

xyplot, *63*, *84*