Package ‘MRMR’

August 29, 2016

Title Multivariate Regression Models for Reserving

Description Non-life runoff reserves may be analyzed using linear models. This generalizes the special cases of multiplicative chain ladder and
the additive model. In addition, the package provides visual and statistical diagnostics to assess the quality of modeled link ratios.

Version 0.1.4

Maintainer Brian A. Fannin <FanninQED@Yahoo.com>

Author Brian A. Fannin

License GPL (>= 2)

Depends lubridate

Imports reshape2, ggplot2, stats, lmtest, methods, plyr

Suggests testthat, mondate

Collate 'CreateOriginPeriods.R' 'CreateEvaluationDates.R'
'CreateDevelopmentLags.R' 'TriangleAdjustMeasures.R'
'Triangle.R' 'TriangleModel.R' 'TriangleProjection.R'
'CompleteTriangle.R' 'GetTriangleData.R' 'Help.R'
'LatestDiagonal.R' 'PlotModelFactors.R' 'PlotModelGoF.R'
'PlotResiduals.R' 'PlotTriangle.R' 'PlotTriangleModel.R'
'ProjectToDate.R' 'ProjectToDev.R' 'ProjectValues.R'
'SerialCorrelation.R' 'SummaryTriangleModel.R' 'TriangleMeta.R'
'data.R'

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-16 15:16:52

R topics documented:

CompleteTriangle *CompleteTriangle*

Description

This function will bind the projected values to the base triangle data for a "complete" triangle. This facilitates comparison of ultimates between multiple TriangleModels.

Usage

```
CompleteTriangle(objProjection)
```

Arguments

objProjection	A TriangleProjection object

Value

A data frame with the sample data (the "upper triangle") bound with the projected data (the "lower triangle").

`CreateCumulative` *Create cumulative*

Description

Create cumulative

Usage

```
CreateCumulative(dfTriangleData, measureCols, Groups)
```

Arguments

`dfTriangleData`	A data frame of triangle variables
`measureCols`	A character vector which holds column names identifying stochastic measures
`Groups`	A character vector which holds column names identifying groups

Value

A data frame of measures with cumulatives included

See Also

`CreateIncrementals`, `CreatePriors`

`CreateDevelopmentLags` *Create triangle development lags*

Description

If the triangle dataframe does not record development lags as lubridate periods, they must be created. Development lags may be established one of three ways: 1. The development lags are passed in as lubridate periods. Everything's cool. The evaluation dates are established by adding the periods to the starting point of the origin periods. 2. The development lags are passed in as integers, with a presumed time period. The program will establish lubridate period objects using the integers and time periods and then proceed as above. 3. An evaluation date is passed in. Here we must take the difference between the evaluation dates and the origin periods. We will assume months as the default period. The user may pass in another.

Usage

```
CreateDevelopmentLags(LagValues, DevelopmentPeriod = months(1),
  EvaluationDates = NULL, OriginPeriods = NULL, Verbose = TRUE)
```

Arguments

`LagValues`	Vector of development lags
`DevelopmentPeriod`	A period object indicating the common time period between evaluations
`EvaluationDates`	A vector of evaluation dates
`OriginPeriods`	A vector of origin periods
`Verbose`	Show warnings?

Value

A vector of intervals

See Also

`CreateDevelopmentLags`, `CreateEvaluationDates`

`CreateEvaluationDates` *Create triangle evaluation dates*

Description

Triangle evaluation dates are established by adding development lags to the starting point of the origin periods.

Usage

```
CreateEvaluationDates(OriginPeriod, DevelopmentLag)
```

Arguments

`OriginPeriod`	A vector of interval objects
`DevelopmentLag`	A vector of period objects

Value

A vector of intervals

See Also

`CreateDevelopmentLags`, `CreateOriginPeriods`

Examples

```
## Not run:
OriginStart = c(mdy("1/1/2000"), mdy("1/1/2000"), mdy("1/1/2001"))
OriginEnd = c(mdy("12/31/2000"), mdy("12/31/2000"), mdy("12/31/2001"))
OriginPeriod = CreateOriginPeriods(OriginStart, OriginEnd)
DevelopmentLag = c(months(12), months(24), months(12))

EvaluationDates = CreateEvaluationDates(OriginPeriod, DevelopmentLag)
EvaluationDates

## End(Not run)
```

CreateIncrementals *Create incrementals*

Description

Create incrementals

Usage

```
CreateIncrementals(dfTriangleData, measureCols, Groups)
```

Arguments

`dfTriangleData`	A data frame of triangle variables
`measureCols`	A character vector which holds column names identifying stochastic measures
`Groups`	A character vector which holds column names identifying groups

Value

A data frame of measures which includes incrementals

See Also

`CreateCumulative`, `CreatePriors`

`CreateOriginPeriods` *CreateOriginPeriods*

Description

This will create a set of origin period values

Usage

```
CreateOriginPeriods(OriginStart, OriginEnd = NULL, OriginLength = years(1),
  StartDay = 1, StartMonth = 1, Verbose = FALSE)
```

Arguments

`OriginStart`	Either a vector of date-time objects, or a vector of numbers indicating the year.
`OriginEnd`	A vector of date-time objects. If this argument is supplied, it is assumed that OriginStart contains date-time objects.
`OriginLength`	A Period object. These are easily created as shown in the example below. The default is a period of one year. If OriginStart and OriginEnd are supplied, this argument is ignored.
`StartDay`	If OriginStart and OriginEnd are supplied, this argument is ignored.
`StartMonth`	If OriginStart and OriginEnd are supplied, this argument is ignored.
`Verbose`	Boolean indicating whether or not to display warning messages.

Details

If the triangle dataframe does not have lubridate intervals, they must be created. Origin periods may be established one of three ways: 1. The origin periods are passed in as POSIX dates. This is a simple matter of forming the interval using lubridate. 2. The origin periods are passed in with a start date, but no end date. We need to have a common period to establish the end date. 3. The origin periods are passed in as parts of a date. This will typically happen if we know the year, but not the month or day. In this case , the user may pass in month and day values

Value

A vector of intervals

See Also

`CreateDevelopmentLags`, `CreateEvaluationDates`

Examples

```
## Not run:
# Case 1
library(lubridate)
OriginStart = c(mdy("1/1/2000"), mdy("1/1/2000"), mdy("1/1/2001"))
OriginEnd = c(mdy("12/31/2000"), mdy("12/31/2000"), mdy("12/31/2001"))

OriginPeriods = CreateOriginPeriods(OriginStart, OriginEnd)
OriginPeriods

# Case 2
OriginStart = c(mdy("1/1/2000"), mdy("1/1/2000"), mdy("1/1/2001"))
OriginPeriods = CreateOriginPeriods(OriginStart, OriginLength = months(12))
OriginPeriods

# Case 3
OriginStartYear = c(2000, 2000, 2001)
OriginPeriods = CreateOriginPeriods(OriginStartYear, OriginLength = years(1)
                                     , StartDay = 1, StartMonth = 1)
OriginPeriods

## End(Not run)
```

CreatePriors *Create priors*

Description

Create priors

Usage

```
CreatePriors(dfTriangleData, measureCols, Groups)
```

Arguments

dfTriangleData	A data frame of triangle variables
measureCols	A character vector which holds column names identifying stochastic measures
Groups	A character vector which holds column names identifying groups

Value

A data frame of measures which includes prior values

See Also

CreateIncrementals, CreateCumulative

`FitSerialCorrelation` *Fit the serial correlation in a triangle*

Description

Fit the serial correlation in a triangle

Usage

```
FitSerialCorrelation(objTriangleModel)
```

Arguments

`objTriangleModel`	A Triangle model

`FormMeasureNames` *Form measures*

Description

Form measures

Usage

```
FormMeasureNames(Measures, Cumulative = TRUE)
```

Arguments

`Measures`	A character vector of stochastic measure names
`Cumulative`	Boolean indicating whether the measure names are cumulative or incremental

`Friedland` *Friedland data*

Description

This is a single triangle taken from blah by Jacqueline Friedland, page 65.

References

http://www.casact.org/library/studynotes/Friedland_estimating.pdf

`GetStochasticColumnNames` *GetStochasticColumnNames*

Description

GetStochasticColumnNames

Usage

```
GetStochasticColumnNames(MeasureNames)
```

Arguments

`MeasureNames`	A character vector of base measure names

Value

A character vector of measure names augmented with the words Incremental, Cumulative and Prior

`GetTriangleData` *GetTriangleData*

Description

This function will return data values from a triangle.

Usage

```
GetTriangleData(Triangle, OriginPeriodStart = NULL, DevInteger = NULL,
  EvaluationDate = NULL, Measure)
```

Arguments

`Triangle`	A Triangle object
`OriginPeriodStart`	A vector of origin years. This parameter may be null.
`DevInteger`	A vector of development integers. This parameter may be null.
`EvaluationDate`	A vector of evaluation dates. This parameter may be null.
`Measure`	A character vector with the names of measures to return.

Value

A data frame

`is.Triangle` *is.Triangle*

Description

Tests whether the object is a triangle

Usage

```
is.Triangle(object)
```

Arguments

`object`	The object to be tested

Value

TRUE if the object is a triangle, FALSE if it is not

`LatestDiagonal` *LatestDiagonal*

Description

This function will return all of the values for the most recent evaluation date. Note that this applies for each origin period individually. For example, if some origin periods have an evaluation at December 31, 2010, but others only have evaluations at December 31, 2009, the data frame which is returned will have two different evaluation dates present.

Usage

```
LatestDiagonal(x)
```

Arguments

`x`	a data frame or a triangle

Value

A data frame

`Mack` *Mack data*

Description

RAA data from Mack's paper

`MRMR` *Multivariate Regression Models for Reserving*

Description

MRMR allows an actuary to create sets of loss data and forecast liabilities. It uses a set of 3 S4 objects to store data, models and predictions.

Details

Triangle

A Triangle is a collection of aggregate loss data. All triangles must have a defined set of OriginPeriods, a defined set of DevelopmentIntervals and data along those axes. A triangle may carry additional descriptive information such as line of business, geographic region and so on.

TriangleModel

A TriangleModel is a statistical model fit to triangle data. The formula may be defined by the user and will generally be a linear or generalized linear model. A triangle may have more than one model. It usually will.

TriangleProjection

A TriangleProjection is a prediction based on a TriangleModel. A TriangleModel may have more than one projection.

`Multiline` *Multiline data*

Description

This data set is taken from the NAIC data prepared by Glenn Meyers and Peng Shi

References

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

`newTriangle` *Create a Triangle object.*

Description

Create a Triangle object.

Usage

```
newTriangle(TriangleData, OriginPeriods = NULL, DevelopmentLags = NULL,
  OriginEnd = NULL, OriginLength = years(1), StartDay = 1,
  StartMonth = 1, DevelopmentPeriod = months(1), EvaluationDates = NULL,
  OriginPeriodType = "Accident Year", TriangleName = NULL,
  StaticMeasures = NULL, StochasticMeasures = NULL, Groups = NULL,
  Cumulative = TRUE, Verbose = TRUE)
```

Arguments

`TriangleData` A dataframe

`OriginPeriods` The name of the column in the TriangleData which holds the origin period.

`DevelopmentLags` The column which holds the development lags.

`OriginEnd` If the OriginPeriods argument refers to the start date of an origin period, this column holds the end dates.

`OriginLength` If origin period is not an interval, this is used to construct the origin period.

`StartDay` If origin period is not an interval, this is used to construct the origin period.

`StartMonth` If origin period is not an interval, this is used to construct the origin period.

`DevelopmentPeriod` If DevelopmentLags is not a period object, this is used to contruct DevelopmentLags.

`EvaluationDates` A vector of dates corresponding to the data in TriangleData.

`OriginPeriodType` A character value describing the type of origin period.

`TriangleName` A character value used to refer to the Triangle object.

`StaticMeasures` A character vector which names the static measures in the Triangle object.

`StochasticMeasures` A character vector which names the stochastic measures in the Triangle object.

`Groups` A character vector which names the column which contains grouping data.

`Cumulative` Boolean indicating if the stochastic measures are cumulative or incremental.

`Verbose` Boolean indicating whether or not warnings should be displayed.

newTriangleModel *Create a new TriangleModel object*

Description

Create a new TriangleModel object

Usage

```
newTriangleModel(Triangle, Response, Predictor, FitCategory,
  Intercept = FALSE, Alpha = 0, Tail = NULL)
```

Arguments

`Triangle`	A Triangle object
`Response`	Character vector indicating the response being measured
`Predictor`	Character vector indicating the variable used to predict the response
`FitCategory`	Character vector indicating the column used to categorize the predictor variable
`Intercept`	Boolean indicating whether or not to include an intercept
`Alpha`	Numeric indicating the parameter used to weight the predictors
`Tail`	Integer indicating the maximum development lag for grouping

PlotModelFactors *PlotModelFactors*

Description

This function will plot the model factors associated with a triangle model.

Usage

```
PlotModelFactors(objTriangleModel)
```

Arguments

`objTriangleModel`	A TriangleModel object

Value

A ggplot2 plot object

See Also

`PlotModelFactors`

`PlotModelGoF` *PlotModelGoF*

Description

This function will plot the F distribution associated with the TriangleModel, along with a vertical line indicating the F statistic for this model.

Usage

```
PlotModelGoF(objTriangleModel)
```

Arguments

`objTriangleModel`
A TriangleModel object

Value

A vector of intervals

See Also

`PlotModelFactors`

`PlotResiduals` *PlotResiduals*

Description

This will produce a 2x2 set of residual graphs.

Usage

```
PlotResiduals(objTriangleModel)
```

Arguments

`objTriangleModel`
A TriangleModel object

Details

This function will produce four charts.

Value

This function does not return a value.

`plotSerialCorrelation` *Plot the serial correlation in a triangle*

Description

Plot the serial correlation in a triangle

Usage

```
plotSerialCorrelation(objTriangleModel)
```

Arguments

`objTriangleModel`	A Triangle model

`plotTriangle` *plot.Triangle*

Description

plot.Triangle

Usage

```
plotTriangle(objTriangle, Response, Predictor, Group = "OriginPeriodStart",
  Lines = TRUE, FitLines = FALSE)
```

Arguments

`objTriangle`	A triangle object
`Response`	The measure being plotted
`Predictor`	The variable used to predict the response
`Group`	The name of the group column used to group the data. The default is OriginPeriodStart
`Lines`	Draw lines to connect the observations?
`FitLines`	Draw a line of best fit? Note that fit lines will have an intercept

plotTriangleModel *plotTriangleModel*

Description

plotTriangleModel

Usage

```
plotTriangleModel(objTriangleModel)
```

Arguments

objTriangleModel	A TriangleModel object

Value

None

See Also

PlotModelGoF, PlotModelFactors

ProjectToDate *ProjectToDate*

Description

This function

Usage

```
ProjectToDate(objTriangleModel, lOriginYears, AsOfDate)
```

Arguments

objTriangleModel	A TriangleModel object
lOriginYears	A list of origin years
AsOfDate	A date to which to project

Value

A data frame which has projected dates and columns for the new stochastic values

`summaryTriangleModel` *summaryTriangleModel*

Description

summaryTriangleModel

Usage

```
summaryTriangleModel(objTriangleModel)
```

Arguments

`objTriangleModel`	TriangleModel object

Value

A vector of intervals

See Also

`CreateCumulative`, `CreatePriors`

`Triangle-class` *Triangle class*

Description

Triangle is an S4 class used to store aggregated loss data. All triangles must have a defined set of OriginPeriods, a defined set of DevelopmentIntervals and data along those axes. A triangle may carry additional descriptive information such as line of business, geographic region and so on.

Details

One will rarely, if ever use the setClass method directly. The function `newTriangle` will generally be used to create a new Triangle object

See Also

`newTriangle`

TriangleModel-class *TriangleModel class*

Description

Triangle is an S4 class used to store a model fit to a Triangle object.

Details

Some stuff

See Also

`Triangle-class`

TriangleProjection *TriangleProjection*

Description

This will construct a TriangleProjection object

Usage

```
TriangleProjection(objTriangleModel, ProjectToDev = TRUE, MaxDev = 10,
  AsOfDate = NULL)
```

Arguments

`objTriangleModel`	A TriangleModel object
`ProjectToDev`	Boolean indicating whether one is projecting to a maximum development interval. If this paramter is FALSE, there must be an argument for AsOfDate
`MaxDev`	The maximum development interval to which to project.
`AsOfDate`	The date to which one wants to project.

`TriangleProjection-class`

TriangleProjection class

Description

TriangleProjection is an S4 class used to project values.

See Also

`newTriangle`

Index