# ChainLadder CRAN 2016 Release

Changes since prior release

*ChainLadder Contributors*

*August 18, 2016*

The current release of the ChainLadder package on CRAN is 0.2.3. Changes and bug fixes since the prior release are:

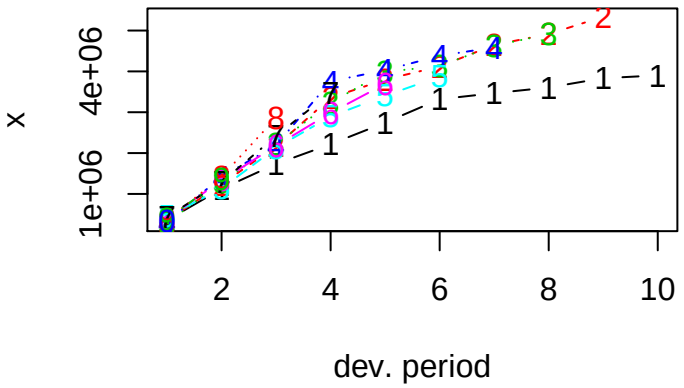
- Changes
  - Triangles may now have non-numeric rownames
  - glmReserve "exposure" attribute may now have names
  - glmReserve adds support for negative binomial GLM
  - Clarified warnings issued by MackChainLadder
  - New unit tests
- Bug Fixes
  - Fixed tail extrapolation in Vignette. Thanks to Mark Lee.

## Changes

### Triangles may now have non-numeric rownames

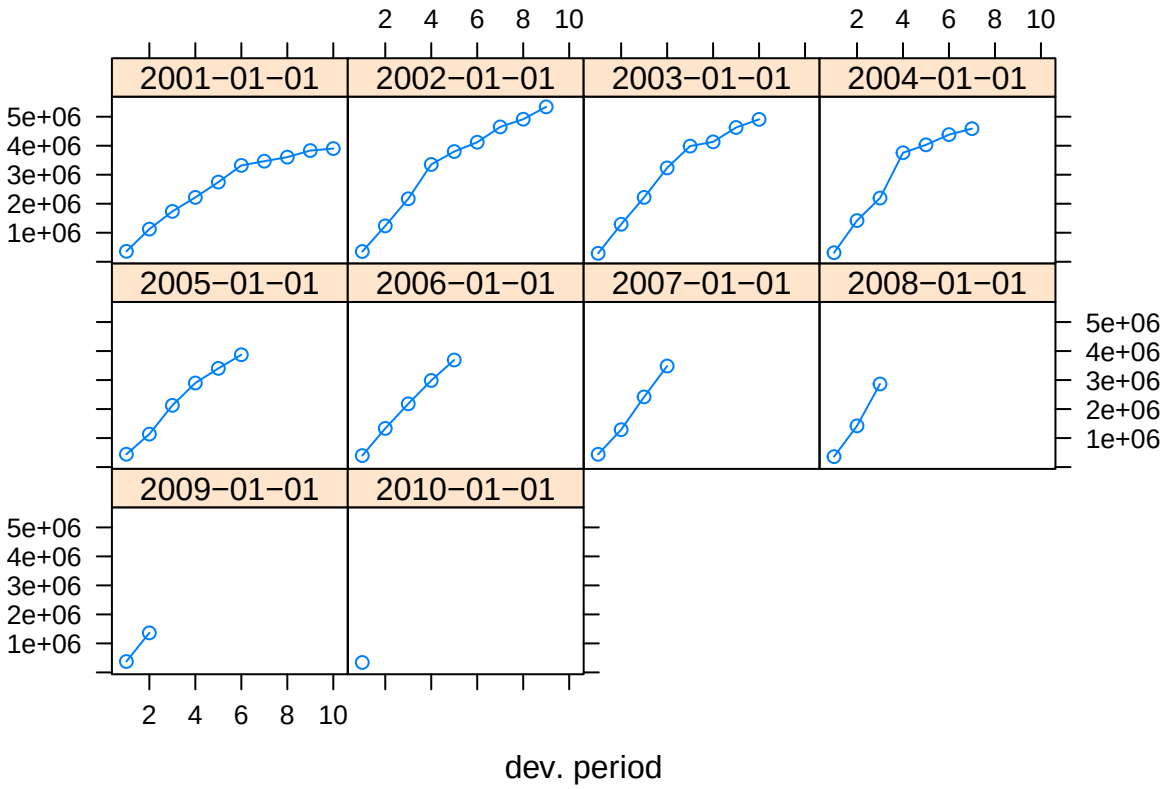
Previously it was required that the row and column names of a triangle be convertible to numeric, although that "requirement" did not always cause a problem. For example, the following sets the rownames of GenIns to the beginning Date of the accident year, and the default call to plot displays the triangle without error

```
x <- GenIns
rownames(x) <- paste0(2001:2010, "-01-01")
plot(x)
```




A plot with the lattice=TRUE option that previously would blow up now displays with nice headings.

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




It can often be useful to have "origin" values that are not necessarily convertible to numeric. For example, suppose you have a table of claim detail at various evaluation dates. Invariably, such a table will have a Date field holding the date of loss. It would be nice to be able to summarize that data by accident year "cuts". It turns out there's a builtin function in R that will get you most of the way there. It's called 'cut'.

Here we take the GenIns data in long format and generate 50 claims per accident period. We assign each claim a random date within the year. The incurred (or paid) "value" given is a random perturbation of one-fiftieth of GenInsLong$value. We accumulate the detail into an accident year triangle using ChainLadder's as.triangle method. The summarized triangle displayed at the end is very similar to GenIns, and has informative row labels.

```
x <- GenInsLong
# start off y with x's headings
y <- x[0,]
names(y)[1] <- "lossdate"
set.seed(1234)
n = 50 # number of simulated claims per accident perior
for (i in 1:nrow(x)) {
  y <- rbind(y,
             data.frame(
               lossdate = as.Date(
                 as.numeric(as.Date(paste0(x[i, "accyear"]+2000, "-01-01"))) +
                   round(runif(n, 0, 364),0), origin = "1970-01-01"),
               devyear = x[i, "devyear"],
               incurred.claims = rnorm(n, mean = x[i, "incurred claims"] / n,
                                       sd = x[i, "incurred claims"]/(10*n))
             ))
```

```
}
# here's the magic cut
y$ay <- cut(y$lossdate, breaks = "years")
# this summarized triangle is very similar to GenIns
as.triangle(y, origin = "ay", dev = "devyear", value = "incurred.claims")
```

```
##             devyear
## ay                  1       2       3       4       5       6       7
##   2001-01-01 349741.1 1109368 1737850 2265706 2749056 3318464 3469142
##   2002-01-01 352821.5 1245621 2132200 3377061 3820987 4148933 4610189
##   2003-01-01 296547.8 1275881 2198221 3235844 3944931 4113276 4623159
##   2004-01-01 313669.5 1392038 2171462 3774168 4035879 4461897 4661352
##   2005-01-01 443940.5 1138787 2190873 2905444 3371444 3849587      NA
##   2006-01-01 391526.6 1324732 2230006 3000719 3742811      NA      NA
##   2007-01-01 446941.9 1292116 2416001 3404734      NA      NA      NA
##   2008-01-01 349330.2 1425022 2844242      NA      NA      NA      NA
##   2009-01-01 369893.1 1368242      NA      NA      NA      NA      NA
##   2010-01-01 346492.8      NA      NA      NA      NA      NA      NA
##             devyear
## ay                  8       9      10
##   2001-01-01 3549578 3769684 3980606
##   2002-01-01 4891852 5311927      NA
##   2003-01-01 4900318      NA      NA
##   2004-01-01      NA      NA      NA
##   2005-01-01      NA      NA      NA
##   2006-01-01      NA      NA      NA
##   2007-01-01      NA      NA      NA
##   2008-01-01      NA      NA      NA
##   2009-01-01      NA      NA      NA
##   2010-01-01      NA      NA      NA
```

The user is encouraged to experiment with other cut's – e.g., breaks = "quarters" will generate accident quarter triangles.

## glmReserve "exposure" attribute may now have names

Previously, when an "exposure" attribute was assigned to a triangle for use with glmReserve, it was assumed/expected that the user would supply the values in the same order as the accident years. Then, behind the scenes, glmReserve would use an arithmetic formula to match the exposure with the appropriate accident year using the numeric "origin" values after the triangle had been converted to long format.

glmReserve now allows for "exposure" to have "names" that coincide with the rownames of the triangle, which are used to match to origin in Long format. Here is an example, newly found in ?glmReserve.

```
GenIns2 <- GenIns
rownames(GenIns2) <- paste0(2001:2010, "-01-01")
expos <- (7 + 1:10 * 0.4) * 10
names(expos) <- rownames(GenIns2)
attr(GenIns2, "exposure") <- expos
glmReserve(GenIns2)
```

```
##              Latest Dev.To.Date Ultimate     IBNR       S.E        CV
```

```
## 2002-01-01  5339085  0.98258394  5433719    94634  110099.9 1.1634283
## 2003-01-01  4909315  0.91271125  5378826   469511  216043.4 0.4601455
## 2004-01-01  4588268  0.86605312  5297906   709638  260872.1 0.3676129
## 2005-01-01  3873311  0.79727286  4858200   984889  303550.0 0.3082073
## 2006-01-01  3691712  0.72228301  5111171  1419459  375013.9 0.2641949
## 2007-01-01  3483130  0.61531018  5660771  2177641  495378.0 0.2274838
## 2008-01-01  2864498  0.42219349  6784799  3920301  789961.1 0.2015052
## 2009-01-01  1363294  0.24162172  5642266  4278972 1046513.8 0.2445713
## 2010-01-01   344014  0.06922055  4969825  4625811 1980101.4 0.4280550
## total      30456627  0.61982473 49137483 18680856 2945660.9 0.1576834
```

## glmReserve adds support for negative binomial GLM

The glmReserve function now supports the negative binomial GLM, a more natural way to model over-dispersion in count data. The model is fitted through the `glm.nb` function from the `MASS` package.

To fit the negative binomial GLM to the loss triangle, simply set `nb = TRUE` in calling the glmReserve function:

```
(fit6 <- glmReserve(GenIns, nb = TRUE))
```

```
##         Latest Dev.To.Date Ultimate     IBNR        S.E        CV
## 10     5339085  0.54175517  9855162  4516077 1380681.59 0.3057259
## 2      4909315  0.98134662  5002631    93316   37402.11 0.4008113
## 3      4588268  0.91131576  5034773   446505  132949.43 0.2977557
## 4      3873311  0.86371868  4484459   611148  147083.10 0.2406669
## 5      3691712  0.78819814  4683736   992024  210714.29 0.2124085
## 6      3483130  0.70562755  4936216  1453086  290921.41 0.2002094
## 7      2864498  0.56715320  5050660  2186162  435789.89 0.1993402
## 8      1363294  0.27112062  5028367  3665073  779454.57 0.2126710
## 9       344014  0.07702236  4466417  4122403  973734.25 0.2362055
## total 30456627  0.62742290 48542422 18085795 2237970.23 0.1237419
```

## New unit tests

New files in the /inst/unittests/ folder can be used for future enhancements

```
* runit.Triangles.R for Triangles.R
* runit.glmReserve.R for glmReserve.R
```

Contributors of new contributions to those R files are encouraged to utilize those runit scripts for testing, and, of course, add other runit scripts as warrantted. (Thank)

## Clarified warnings issued by MackChainLadder

By default, R's 'lm' method generates a warning when it detects an "essentially perfect fit". This can happen when one column of a triangle is identical to the previous column; i.e., when all link ratios are the same. In the example below, the second column is a fixed constant, 1.05, times the first column. ChainLadder previously issued the lm warning below.

```
x <- matrix(byrow = TRUE, nrow = 4, ncol = 4,
            dimnames = list(origin = LETTERS[1:4], dev = 1:4),
            data = c(
              100, 105, 105.00001, 105.00001,
              200, 210, 210.00001, NA,
              300, 315, NA, NA,
              400, NA, NA, NA)
            )
mcl <- MackChainLadder(x, est.sigma = "Mack")

Warning messages:
1: In summary.lm(x) : essentially perfect fit: summary may be unreliable
2: In summary.lm(x) : essentially perfect fit: summary may be unreliable
3: In summary.lm(x) : essentially perfect fit: summary may be unreliable
```

which may have raised a concern with the user when none was warranted.

Now ChainLadder issues an informational message at the console:

```
mcl <- MackChainLadder(x, est.sigma = "Mack")


## Information: essentially no development in data for period(s):
##  '1-2'
```

# Bug fixes

## Fixed tail extrapolation

Fixed tail extrapolation in Vignette. (Thanks to Mark Lee.)

```
* Fixed summary calls.
* Updated documentation for weights parameter of chainladder method.
* Fixes for tail extrapolation in Vignette and Chainladder
  1) The calculation for the tail log-linear extrapolation given in
       the vignette had a minor error.  This has been corrected,
       and the result now agrees with the results of
       MackChainLadder(RAA, tail=TRUE).
  2) The calculation of the tail using the log-linear extrapolation
       in ChainLadder.R had a potential error - when clratios has
       values of less than unity they are dropped, but the extrapolation
       was started from a quantity indexed by the length of f, not
       the value of fn.  This changes the results if clratios has a
       pattern like e.,g.: ... 1.1, 0.98,1.01,0.005 (i.e. a link ratio
       less than unity which is not the last value)
  3) Minor fix to the comments in ChainLadder.R and MackChainLadder.R,
       fixing notation for alpha which is now consistent with the
       documentation and Mack's original paper.
```