A Comparison of Resampling Methods for Bootstrapping Triangle GLMs

By Thomas Hartl, Department of Mathematics, Bryant University, Smithfield, Rhode Island

Overview

Bootstrapping is often employed for quantifying the inherent variability of development triangle GLMs. While easy to implement, bootstrapping approaches frequently break down when dealing with actual data sets.

Often this happens because linear rescaling leads to negative values in the resampled incremental development data. Two alternative methods are presented: split-linear rescaling, and limited Pareto resampling.

Comparisons based on a VBA implementation show that the new methods perform about the same or are more efficient than linear rescaling.

Linear Rescaling Often Breaks Down

While more general approaches utilizing different definitions of residuals are discussed in the literature (e.g [1], [2], [3], and [6]), many practical applications implement resampling procedures based on linear rescaling of Pearson residuals.

Using a notation similar to that adopted in [5] and [6], the key equation for generating pseudo data is

$$y^* = \hat{y} + \sqrt{\phi V(\hat{y})} \cdot r,$$

where r is a randomly chosen standardized residual.

At the same time GLMs for development triangles usually use the log link function which requires that all data points have to be positive. Popular GLMs for development triangles (such as the ODP model) often have a large dispersion factor, ϕ, and small incremental data values for mature development periods. This results in negative resampling values. A detailed example of this is discussed in [3].

Split-Linear Rescaling – Intuition

The intuitive idea behind split-linear rescaling is as follows: if the standard Pearson residual rescaling procedure results in resampling values that are below a given percentage, π_{min}, of the expected mean, we split the standard Pearson residual resampling values into a lower set and an upper set.

Next we "squeeze" the lower resampling values together, so that they do no longer dip below the given percentage of the expected mean. The "squeezing" operation does preserve the mean, but it will lower the variance of the resampling distribution.

To offset this, we apply a mean preserving "expansion" operation to the upper resampling values.

Split-Linear Rescaling – Implementation

The most tricky part is finding a suitable partition into $\boldsymbol{y_l^*}$ and $\boldsymbol{y_u^*}$. At worst one can check all possible partitions such that all elements of $\boldsymbol{y_l^*}$ are strictly less than the elements of $\boldsymbol{y_u^*}$. This is not a concern from a performance perspective, because it does not affect the application of the method during the Monte Carlo resampling phase.

During the resampling iterations, split-linear rescaling differs only slightly from regular linear rescaling:

1. There is an increased memory requirement because two sets of rescaling parameters need to be maintained
2. There is an additional comparison to decide which set of rescaling parameters needs to be used.

Split-Linear Rescaling – Formulas

Assume that the original resampling distribution, $\boldsymbol{y}^*$, has m data values and mean μ. We partition $\boldsymbol{y}^*$ into two subsets, $\boldsymbol{y_l^*}$ and $\boldsymbol{y_u^*}$, with q and r values, and means μ_l and μ_u. The new distribution is given by

$$y^{*\prime} = \begin{cases} \mu_l + c_l(y^* - \mu_l), & \text{if } y^* \in \boldsymbol{y_l^*} \\ \mu_u + c_u(y^* - \mu_u), & \text{if } y^* \in \boldsymbol{y_u^*} \end{cases}.$$

The scaling factor for the lower set is given by

$$c_l = \frac{\mu_l - \pi_{min}\mu}{\mu_l - y_1^*},$$

where y_1^* is the smallest value in $\boldsymbol{y_l^*}$. The scaling factor for the upper set is obtained by

$$c_u = \sqrt{1 + (1 - c_l^2)\frac{q\sigma^2_{y_l^*}}{r\sigma^2_{y_u^*}}},$$

where $\sigma^2_{y_l^*}$ and $\sigma^2_{y_u^*}$ are the variances of the lower and upper subsets.

Note that there are two situations in which this procedure does not work. Firstly, we need a partition such that $\mu_l > \pi_{min}\mu$. Once we have found that, we may discover that $\sigma^2_{y_u^*} = 0$. This means that all the values in $\boldsymbol{y_u^*}$ are the same (including the degenerate case where there are no values in $\boldsymbol{y_u^*}$). In this case we cannot expand the upper subset. Passing the $\sigma^2_{y_u^*} > 0$ hurdle, however, still leaves open the possibility that the procedure breaks down because ${y_u^*}_1{}' < \pi_{min}\mu$. This means that the smallest value in $\boldsymbol{y_u^*}$ gets mapped to a value below $\pi_{min}\mu$, thus violating the very constraint we wanted to satisfy by using split-linear rescaling.

So, split-linear resampling is no cure all, but it does significantly expand the class of triangle GLMs that can be bootstrapped.

Limited Pareto Resampling

An alternative to the messiness of rescaling residuals is to use parametric resampling: we postulate that pseudo-data can be sampled from a distribution that has a mean variance-relationship in line with our model assumptions. As it turns out, a shifted and limited Pareto distribution with a Pareto index of one is easy to work with from a computational point of view. The CDF is given by

$$F(x) = \begin{cases} 0 & x + c < a \\ 1 - \dfrac{a}{x + c} & a \le x + c < b. \\ 1 & b \le x + c \end{cases}$$

Note that $P(X = b - c) = {}^a\!/_b$ (i.e. there is probability mass).

The mean and variance are given by

$$E(X) = a\left(1 - \ln\frac{a}{b}\right) - c, \quad Var(X) = 2ab - a^2 - a^2\left(1 - \ln\frac{a}{b}\right)^2.$$

For robust simulations it is desirable for ${}^a\!/_b$ not to be too small. One strategy for finding suitable parameters a, b, c is to solve ${}^a\!/_b = 0.001$, $E(X) = \mu$, and $Var(X) = V(\mu)$. Note that this system of equations can be solved in closed form. Often the solution will satisfy $a - c \ge \pi_{min}\mu$.

If the first strategy fails, we solve $a - c = \pi_{min}\mu$, $E(X) = \mu$, and $Var(X) = V(\mu)$. This system of equations does not have a closed form solution, but it can readily be solved numerically using the Newton-Raphson method (we combine this with the bisection method for increased stability).

Once the a, b, c parameters have been determined for each cell of our triangle GLM, sampling from the distribution during the Monte Carlo phase can efficiently be accomplished by the inverse transform method. Given a uniform random number, u, we generate x by

$$x = \begin{cases} b - c & u \le \dfrac{a}{b} \\ \dfrac{a}{u} - c & \dfrac{a}{b} < u \end{cases}.$$

At least in VBA for Excel this is actually more efficient than linear rescaling. The reason is that u is used directly (division operation), whereas for linear rescaling u needs to be converted to an index that is used to access the needed standardized residual (multiplication plus array access).

Performance – Resampling Only

Friedland Data

Starting test script at 7/10/2014 6:53:30 PM

Starting test of linear rescaling at 7/10/2014 6:53:30 PM
The current state of the RngStream "<default>":
Cg = {3868657861, 3805326431, 1882214433, 3305730503, 2989917098, 1133457997}

Starting 20 test runs with 5000 iterations of resampling triangle and future cells using linear Pearson residual rescaling:

1.77;1.76;1.75;1.77;1.76;1.78;1.75;1.77;1.77;1.77;1.77;1.78;1.77;1.78;1.78;1.77;1.78;1.78;1.80;1.77;
The average time was 1.771 with a sample standard deviation of 1.29E-02.

Starting test of split-linear rescaling at 7/10/2014 6:54:06 PM
The current state of the RngStream "<default>":
Cg = {3868657861, 3805326431, 1882214433, 3305730503, 2989917098, 1133457997}

Starting 20 test runs with 5000 iterations of resampling triangle and future cells using split-linear residual rescaling:

1.80;1.81;1.83;1.80;1.80;1.83;1.82;1.81;1.80;1.81;1.80;1.80;1.80;1.81;1.80;1.86;1.80;1.83;1.83;1.82;
The average time was 1.814 with a sample standard deviation of 1.51E-02.

Starting test of limited Pareto resampling at 7/10/2014 6:54:42 PM
The current state of the RngStream "<default>":
Cg = {3868657861, 3805326431, 1882214433, 3305730503, 2989917098, 1133457997}

Starting 20 test runs with 5000 iterations of resampling triangle and future cells using limited Pareto resampling:

1.75;1.73;1.74;1.75;1.72;1.74;1.75;1.73;1.75;1.73;1.73;1.75;1.75;1.74;1.75;1.73;1.74;1.75;1.73;1.73;
The average time was 1.740 with a sample standard deviation of 1.01E-02.

Completed test script at 7/10/2014 6:55:17 PM

Performance – Full Bootstrap

Friedland Data

Staring test script at 7/10/2014 7:24:51 PM

Starting test of linear rescaling at 7/10/2014 7:24:51 PM
State of random number generator:
lg = {3868657861, 3805326431, 1882214433, 3305730503, 3305730503, 1133457997}

20 test runs with 5000 iterations of full bootstrap using linear Pearson residual rescaling:

13.70;13.71;13.85;13.91;13.87;13.91;13.91;13.91;13.74;13.77;13.70;13.66;13.82;13.73;13.68;13.71;13.71;13.82;
13.96;13.89;13.87;
Average values: time = 13.797 reserve = 74,876 st.err. outcome = 1,143

Starting test of split-linear rescaling at 7/10/2014 7:29:28 PM
State of random number generator:
lg = {3868657861, 3805326431, 1882214433, 3305730503, 3305730503, 1133457997}

20 test runs with 5000 iterations of full bootstrap using split-linear residual rescaling:

13.85;13.93;13.98;13.96;13.97;13.92;13.91;13.88;13.91;13.74;13.80;13.92;13.96;13.90;13.95;13.84;13.76;
13.74;13.77;13.73;
Average values: time = 13.871 reserve = 74,872 st.err. outcome = 1,142

Starting test of limited Pareto resampling at 7/10/2014 7:34:06 PM
State of random number generator:
lg = {3868657861, 3805326431, 1882214433, 3305730503, 3305730503, 1133457997}

20 test runs with 5000 iterations of full parametric bootstrap using limited Pareto distribution:

12.05;12.03;12.16;12.28;12.16;12.28;12.18;12.20;12.21;12.19;12.16;12.25;12.20;12.27;12.15;12.08;12.04;
12.03;12.12;12.03;
Average values: time = 12.154 reserve = 74,867 st.err. outcome = 1,133

Completed test script at 7/10/2014 7:38:10 PM

Performance Tests

The data set from Friedland was chosen because all three methods work for it. Two test scripts were run: one to directly compare the time spent resampling, and one to show the impact on running a full bootstrap (with model fit and reserve projection).

The data set from Taylor and Ashe demonstrates that split-linear rescaling and limited Pareto resampling do extend the scope of bootstrapping methods. Only the full bootstrap script was run in this case.

For all scripts only the time spent during the Monte Carlo iterations was measured (i.e. time spent on initializing the data structures was excluded).

Performance – Full Bootstrap

Taylor and Ashe Data

Staring test script at 7/10/2014 10:06:46 PM

Starting test of split-linear rescaling at 7/10/2014 10:06:46 PM
State of RNG: lg = {3692455944, 1366884236, 2968912127, 335948734, 335948734, 475798818}

20 test runs with 5000 iterations of full bootstrap using split-linear residual rescaling:

16.6;16.6;16.6;16.6;16.4;16.6;16.6;16.6;16.6;16.6;16.6;16.6;16.6;16.6;16.7;16.7;16.6;16.6;16.6;16.5;16.6;
Average values: time = 16.600 reserve = 18,846,504 st.err. outcome = 3,002,492

Starting test of limited Pareto resampling at 7/10/2014 10:12:18 PM
State of RNG: lg = {3692455944, 1366884236, 2968912127, 335948734, 335948734, 475798818}

20 test runs with 5000 iterations of full parametric bootstrap using limited Pareto distribution:

14.4;14.4;14.4;14.5;14.4;14.6;14.5;14.5;14.6;14.5;14.5;14.6;14.4;14.5;14.5;14.5;14.4;14.3;14.4;14.4;
Average values: time = 14.466 reserve = 18,797,464 st.err. outcome = 2,904,446

Completed test script at 7/10/2014 10:17:08 P

Bibliography

[1] A. C. Davison and D. V. Hinkley, Bootstrap methods and their application. Cambridge; New York, NY, USA: Cambridge University Press, 1997.

[2] P. D. England and R. J. Verrall, "Stochastic claims reserving in general insurance," British Actuarial Journal, vol. 8, no. 03, pp. 443–518, 2002.

[3] T. Hartl, "Bootstrapping generalized linear models for development triangles using deviance residuals," in CAS E–Forum Fall, 2010.

[4] T. Hartl, "GLMs for Incomplete Development Triangles," presented at the 2013 Casualty Loss Reserving Seminar, 2013.

[5] P. McCullagh and J. A. Nelder, Generalized linear models. London; New York: Chapman and Hall, 1989.

[6] P. J. R. Pinheiro, J. M. A. e Silva, and M. de L. Centeno, "Bootstrap Methodology in Claim Reserving," The Journal of Risk and Insurance, vol. 70, no. 4, pp. 701–714, Dec. 2003.

US Industry Auto Paid Claims ($M)

Incremental Input Values

Period	Dev	1	2	3	4	5	6	7	8	9	10
Exp											
1		18,539	14,692	6,831	3,830	2,004	869	456	226	109	89
2		20,410	15,680	7,169	3,900	2,049	954	464	253	122	
3		22,121	16,855	7,413	4,173	2,173	1,005	544	249		
4		22,992	17,104	7,672	4,326	2,270	1,015	500			
5		24,093	17,703	8,108	4,449	2,401	1,053				
6		24,084	17,315	7,671	4,514	2,346					
7		24,370	17,120	7,747	4,538						
8		25,101	17,602	7,943							
9		25,609	17,998								
10		27,230									

Taken from J. Friedland, Estimating Unpaid Claims Using Basic Techniques, page 107.

Data from Taylor and Ashe (1983)

Incremental Input Values

Period	Dev	1	2	3	4	5	6	7	8	9	10
Exp											
1		357,848	766,940	610,542	482,940	527,326	574,398	146,342	139,950	227,229	67,948
2		352,118	884,021	933,894	1,183,289	445,745	320,996	527,804	266,172	425,046	
3		290,507	1,001,799	926,219	1,016,654	750,816	146,923	495,992	280,405		
4		310,608	1,108,250	776,189	1,562,400	272,482	352,053	206,286			
5		443,160	693,190	991,983	769,488	504,851	470,639				
6		396,132	937,085	847,498	805,037	705,960					
7		440,832	847,631	1,131,398	1,063,269						
8		359,480	1,061,648	1,443,370							
9		376,686	986,608								
10		344,014									

This data set has been used as a benchmark by multiple authors, e.g. table 1 in [6]

A copy of the VBA for Excel application used to generate the results presented is available at request. Contact "Thomas Hartl" <thartl@bryant.edu>. All questions and feedback are welcome.

The 49th Actuarial Research Conference
• UC Santa Barbara • July 13-16, 2014 •