`baggingReserve`: a short tour

Jan Janoušek and Michal Pešta

November 12, 2026

This vignette walks through the workflow of the package: a data contract that turns claims and payment transactions into a claim-by-development-year panel, a recursion engine that completes the payment history of every reported claim, by default with the regression trees and bagging of Janoušek and Pešta (2025), *Bagging and regression trees in individual claims reserving*, Statistical Papers 66(4), and a back-test against the payments that followed. It uses the claims of types 2 and 3 of the shipped synthetic portfolio (generated with the article's simulator under another seed; not the article's data) and few bootstrap replicates, so that it builds quickly. The accompanying Journal of Statistical Software manuscript, shipped as the vignette `baggingReserve-jss`, describes the method and the design in full and back-tests the method on the whole portfolio, on an illustrative motor third-party liability dataset (a subset of an insurer's data, drawn in an undocumented way and published in a public code repository) and on further synthetic portfolios.

From claims and payments to a panel

Two tables enter: one row per claim reported by the valuation date and one row per payment transaction. The constructor enforces the valuation date on both, sums payments per claim and development year, keeps recoveries, marks the cells after the valuation date as `NA` and refuses input it cannot interpret.

```
library("baggingReserve")
data("synthetic_claims", "synthetic_payments", package = "baggingReserve")
cl <- synthetic_claims[synthetic_claims$Type %in% c("2", "3"), ]
pay <- synthetic_payments[synthetic_payments$Id %in% cl$Id, ]
panel <- claims_panel(cl, pay, t = 12, L = 11, features = c("Type", "Age", "AccWeekday"))
panel

#> Claims panel: 6,993 claims reported by t = 12
#>   accident years 1-12 (I = 12), development years 0-11
#>   features: Type, Age, AccWeekday (+ reporting delay)
#>   observed cells: 44,246 of 83,916; recoveries: 177; claims without payment: 696

round(as_triangle(panel) / 1e6, 1)

#> Incremental payments (rows: accident years, columns: development years)
#> NA: not observable at t = 12
#>      0   1   2   3   4   5   6   7   8    9  10   11
#> 1  1.1 4.1 5.5 4.0 1.2 1.4 0.4 0.5 0.1 -0.1 0.0 -0.1
#> 2  0.8 5.5 4.8 3.2 1.6 0.9 0.9 0.3 0.1  0.1 0.1   NA
#> 3  1.2 4.2 4.0 2.6 2.1 0.9 0.4 0.2 0.0  0.0  NA   NA
#> 4  1.4 5.7 5.3 4.3 2.8 0.8 1.0 0.1 0.2   NA  NA   NA
```

```
#> 5  1.5 5.8 4.8 3.8 2.8 1.6 0.9 0.3  NA   NA  NA   NA
#> 6  1.9 5.8 5.8 3.1 1.7 1.4 0.2  NA  NA   NA  NA   NA
#> 7  1.3 5.7 6.4 4.0 2.4 1.4  NA  NA  NA   NA  NA   NA
#> 8  2.1 5.3 5.9 4.3 1.7  NA  NA  NA  NA   NA  NA   NA
#> 9  1.8 7.3 6.5 4.2  NA  NA  NA  NA  NA   NA  NA   NA
#> 10 2.4 7.6 7.3  NA  NA  NA  NA  NA  NA   NA  NA   NA
#> 11 1.7 7.3  NA  NA  NA  NA  NA  NA  NA   NA  NA   NA
#> 12 1.9  NA  NA  NA  NA  NA  NA  NA  NA   NA  NA   NA
```

One pruned tree per development year

The tree for development year l (the package calls this index its `period`) predicts the payment of the next development year, X_{l+1}, of a reported claim from its features, its reporting delay and $X_0, \ldots, X_l$; the trees are applied recursively to complete every reported claim's history.

```
tree <- reserve_tree(panel, seed = 1)
tree

#> Single-tree reserving fit: 11 trees, valuation year t = 12
#>   (period = development-year index)
#>   claims: 6,993; features: Type, Age, AccWeekday, rep_delay
#>   control: cp = 0, xval = 10, minsplit = 20; pruning: min_xerror
#>  period n_train       cp leaves
#>       0    4337 0.004778      5
#>       1    5708 0.005052     13
#>       2    5152 0.004341     19
#>       3    4510 0.009837     11
#>       4    3882 0.037620      1
#>       5    3288 0.028000      1
#>       6    2727 0.026910      1
#>       7    2122 0.065880      1
#>       8    1542 0.038610      1
#>       9    1031 0.028860      1
#>      10     515 0.141200      1

pred_tree <- predict(tree)
pred_tree

#> Reserve prediction (single-tree fit): 66 future cells; one prediction column
#>   amounts: reported-claim payments only
#>   next year (13): 20,660,824
#>   total through development year 11: 42,965,379
#>   plug-in completion: the years after next can be biased; completion = "draw" is recommer
```

By default each predicted mean is plugged into the history used by the next tree, as in the source article. Where the occurrence of a payment depends on the payment history, this biases the payments after next year; simulating the payments instead (`completion = "draw"`) removes that mechanism, and it is the recommended completion for totals beyond next year. The next year's payments are the same in expectation under both.

```r
pred_draw <- predict(tree, completion = "draw", R = 50, seed = 1)
cbind(plug_in = calendar_year_payments(pred_tree)[, 1],
      simulated = rowMeans(calendar_year_payments(pred_draw)))[c(1, 11, 12), ]

#>         plug_in simulated
#> 13    20660824  20749948
#> 23      -63268    -63157
#> total 42965379  48569384
```

Bagging, out-of-bag scores, IBNR

`bag_reserve()` grows a bootstrap tree per development year and replicate, completes the panel once per replicate, and records the out-of-bag predictions ("votes"). The $B = 20$ replicates used here demonstrate the interface; they are far too few for stable tail quantiles. The manuscript uses 1000 for its live run and 10,000 for the deposited full run.

```r
bag <- bag_reserve(panel, B = 20, seed = 2026)
oob_score(bag)[, c("period", "n_train", "coverage", "oob_score")]

#>    period n_train coverage oob_score
#> 1       0    4337   1.0000  -0.02231
#> 2       1    5708   1.0000   0.05092
#> 3       2    5152   1.0000   0.16501
#> 4       3    4510   1.0000   0.16681
#> 5       4    3882   1.0000   0.15570
#> 6       5    3288   0.9997   0.06609
#> 7       6    2727   1.0000  -0.10153
#> 8       7    2122   0.9995  -0.09126
#> 9       8    1542   1.0000  -0.09076
#> 10      9    1031   1.0000  -0.06287
#> 11     10     515   1.0000  -0.14709

res <- ibnr_reserve(predict(bag), claim_counts(panel))
res

#> Reserve prediction (bagged fit): 66 future cells; replicates retained: B = 20
#>   amounts: reported-claim payments + IBNR component
#>   spread of the replicates: bootstrap resampling variation of the bagged trees (no paymer
#>   next year (13): 23,874,685 (bagged mean)
#>   total through development year 11: 67,006,669 (bagged mean)
#>   plug-in completion: the years after next can be biased; completion = "draw" is recommer

summary(res, probs = c(0.025, 0.5, 0.975))$tables$total

#>           mean      sd      min    q2.5%     q50%   q97.5%      max
#> 13    23874685  986931 22322171 22365171 23834232 25380427 25515260
#> 14    16244211 2112020 12025538 12168315 16757594 19339890 19736335
#> 15    10288320 1929519  7173892  7194540 10029303 13257801 13653159
#> 16     7669556 1752444  5123846  5185403  7604978 10852312 10969818
#> 17     5287332 1855667  2129489  2178581  5497958  8179289  8471889
#> 18     2708536 1520290   493362   692478  2471108  5874390  6151895
```

```
#> 19     1335003 1053346   -52500   164280  1064798  3986911  4023083
#> 20      265307  509032  -647026  -513880   166744  1270168  1476810
#> 21     -156406  344449  -730982  -722491  -138718   482179   508060
#> 22     -263648  281218 -1039221  -836239  -216657    94070   132459
#> 23     -246227  295384 -1094569  -962227   -86770    -2395    -2395
#> total 67006669 8561194 54426074 54546512 67349527 81567238 83896525
```

The quantiles are empirical quantiles, over the replicates, of the reserve predicted by one set of unpruned bootstrap trees (one per development year); they describe this resampling variation and are neither prediction intervals for the future payments nor confidence intervals for the bagged mean. For a bagged fit the completion is chosen when fitting:

```
bag_draw <- bag_reserve(panel, B = 20, seed = 2026, completion = "draw")
res_draw <- ibnr_reserve(predict(bag_draw), claim_counts(panel))
```

Comparators and back-test

The shipped truth allows a back-test against the payments that actually followed. The truth tables must describe the same portfolio as the prediction, including the claims reported after the valuation date: a subset is selected by the same criterion as the panel (here the claim type), never by the identifiers of the reported claims. The chain ladder on the panel's paid triangle is scored against the same truth.

```
data("synthetic_claims_full", "synthetic_payments_full", package = "baggingReserve")
cl_full <- synthetic_claims_full[synthetic_claims_full$Type %in% c("2", "3"), ]
pay_full <- synthetic_payments_full[synthetic_payments_full$Id %in% cl_full$Id, ]
cl_ref <- chain_ladder(as_triangle(panel))
bt <- backtest(res, cl_full, pay_full, reference = cl_ref)
bt$tables$total[, c("calendar", "truth", "predicted", "rel_error", "reference_rel_error")]

#>    calendar    truth predicted rel_error reference_rel_error
#> 1        13 22671996  23874685   0.05305             0.07509
#> 2        14 15595321  16244211   0.04161             0.11901
#> 3        15 12917181  10288320  -0.20352            -0.18432
#> 4        16  4858019   7669556   0.57874             0.17996
#> 5        17  3233876   5287332   0.63498            -0.06710
#> 6        18  1170043   2708536   1.31490             0.17943
#> 7        19   168161   1335003   6.93884             1.87724
#> 8        20   -64028    265307  -5.14360            -2.66417
#> 9        21    75863   -156406  -3.06169            -1.84926
#> 10       22    73066   -263648  -4.60835            -2.00542
#> 11       23     7192   -246227 -35.23622           -15.73724
#> 12    total 60706690  67006669   0.10378             0.03510

bt_draw <- backtest(res_draw, cl_full, pay_full)
bt_draw$tables$total[c(1, 12), c("calendar", "truth", "predicted", "rel_error")]

#>    calendar    truth predicted rel_error
#> 1        13 22671996  23713642   0.04594
#> 12    total 60706690  56221054  -0.07389
```

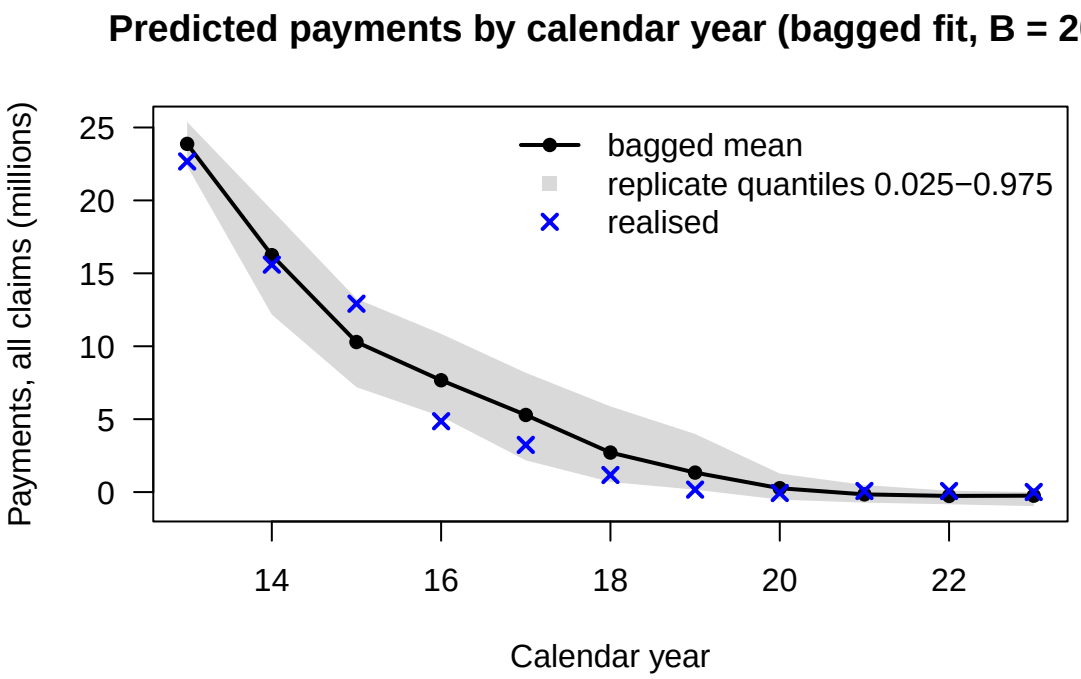


Figure 1: Predicted payments by calendar year with the realised payments.

```
plot(res, "calendar", truth = setNames(bt$tables$total$truth, bt$tables$total$calendar))
```

A back-test at one valuation date is one draw of the error. Valuing the data at several earlier dates (`claims_panel(..., future_payments = "drop")`, see the manuscript) gives an empirical distribution of the errors of the claim-level method and of the chain ladder on one's own data. These errors are a descriptive diagnostic, not independent validation samples: the valuation dates share claims and payments, and the horizon usually changes with the date, so they are dependent and do not give a calibrated predictive distribution.