Universiteit van Amsterdam

Amsterdam School of Economics

Computer class NLIST—Assignment 5

More GLM-related IBNR techniques

In this assignment we study the techniques described in the following papers, available through the Blackboard site:

- England, P. and Verrall, R. (1999). *Analytic and bootstrap estimates of prediction errors in claims reserving,* Insurance: Mathematics and Economics, 25(3), 281-293.
- England, P. (2002). *Addendum to 'Analytic and bootstrap estimates of prediction errors in claims reserving',* Insurance: Mathematics and Economics, 31(3), 461-466.

Also, we study methods in which the known exposures, or a good guess therefor like the premium total, replaces the unknown row parameters, including the method of Bornhuetter-Ferguson. We also look further at the Extended Chain Ladder model with parameters for three aspects of time: portfolio growth, development pattern as well as inflation.

1 Introduction

In England and Verrall (1999), an appropriate residual definition is considered for use in a bootstrap exercise (resampling the residuals) to provide a computationally simple method of obtaining reserve prediction errors for the chain ladder model. They also provide an analytical estimate based on the delta method, see MART, Example 9.1.1. This estimate can be calculated from values stored in the object generated by a `glm()`-call in `R`. Calculation of the first two moments of the predictive distribution only was considered.

In England (2002), the method was extended by using a two-stage process: bootstrapping to obtain the estimation error and simulation to obtain the process error. This has the advantage of providing realizations from the whole predictive distribution, rather than just its first two moments.

Both papers are summarized in Section 10.6 in MART.

Some additional issues encountered in this assignment:

- describing the predictive distribution:
  - graphically, by a histogram and a so-called kernel density estimate;
  - by using statistics such as skewness/kurtosis, quantiles;
- using a gamma error distribution instead of Poisson (that is, assuming a mean-variance relationship with constant coefficient of variation σ/μ rather than constant overdispersion σ^2/μ);
- computing the IBNR-reserve for each separate year of origin.

2 England/Verrall's (1999) bootstrap method

England, P. and Verrall, R. (1999) describe a bootstrap method to estimate the variability of the Chain Ladder estimate for the total reserve to be kept. It goes as follows.

- Start with a *run-off triangle* X_{ij}, $i+j-1 \leq t$.
- Use *Chain Ladder* (by a GLM with row- and column coeficients, a (quasi-)Poisson stochastic structure and a log-link), to find predictions $\widehat{\mu}_{ij} = \widehat{\alpha}_i \widehat{\beta}_j$ for all (i, j).
- Let $\widehat{R}$ be the resulting reserve (sum of future predictions).
- Compute unscaled Pearson *residuals* $R_{ij} = \frac{X_{ij} - \widehat{\mu}_{ij}}{\sqrt{V(\widehat{\mu}_{ij})}}$ with $V(\mu) = \mu$, and an estimate $\widehat{\phi}_{\mathrm{P}}$ of ϕ using the sum of squared residuals. Adjust for bias.
- *Resample* R_{ij}^b from the set of residuals (with replacement) and construct a *pseudo-history* $X_{ij}^b := R_{ij}^b \sqrt{\widehat{\mu}_{ij}} + \widehat{\mu}_{ij} \quad (i+j-1 \leq t)$. Do this for each $b = 1, \ldots,$ `nBoot`.
- Apply the *CL* method to X_{ij}^b to get a new reserve estimate $\widehat{R}^b$.
- The standard deviation of the $\widehat{R}^b$ values is a *bootstrap estimate* for the standard error in the reserve estimate $\widehat{R}$.
- Sample from the *estimated* process (CL-model with parameters $\widehat{\alpha}_i^b$, $\widehat{\beta}_j^b$ and $\widehat{\phi}_{\mathrm{P}}$) to generate a value for the predicted future payments $\widehat{P}^b$.
- The generated values $\widehat{P}^b$ give a *predictive distribution* for the future payments.

3 Loading the data

We will study the bootstrap method using a dataset[1] used in many papers on IBNR. We read the IBNR-triangle (actual observations only, not padded with zeroes) and reconstruct row and column numbers. If there are t years, there are $n = t + (t-1) + \cdots + 1 = \frac{1}{2}t(t+1)$ observations, so obviously $t^2 < 2n < (t+1)^2$, therefore t can be found from n as $t = \lfloor \sqrt{2n} \rfloor$.

```
rm(list=ls(all=TRUE)); options(digits=6)   ##  housekeeping
Xij <- scan(n=55)   ##  Data Taylor & Ashe (1983)
357848 0766940 0610542 0482940 527326 574398 146342 139950 227229 067948
352118 0884021 0933894 1183289 445745 320996 527804 266172 425046
290507 1001799 0926219 1016654 750816 146923 495992 280405
310608 1108250 0776189 1562400 272482 352053 206286
443160 0693190 0991983 0769488 504851 470639
396132 0937085 0847498 0805037 705960
440832 0847631 1131398 1063269
359480 1061648 1443370
376686 0986608
344014
```

[1] see "Second moments of estimates of outstanding claims", G.C. Taylor, F.R. Ashe; Journal of Econometrics, Volume 23, Issue 1, September 1983, 37-61. The numbers are the products of columns 2 and 3 in the table on p. 59–60; bottom to top, left to right; $22 \times 15637 = 344014$, $40 \times 8946.2 = 357848$

```
n <- length(Xij); TT <- trunc(sqrt(2*n))
i <- rep(1:TT, TT:1); j <- sequence(TT:1)
i <- as.factor(i); j <- as.factor(j)
```

4 Chain ladder estimates

We assume *incremental* observations $X_{ij}, i, j = 1, \ldots, t$. In the Chain Ladder model, the means are $\mu_{ij} = \alpha_i \beta_j$ for positive parameters $\alpha_1, \ldots, \alpha_t$ and $\beta_1, \ldots, \beta_t$. The deviance between fitted values and data that is minimized is based on the likelihood ratio for $\phi \times \text{Poisson}(\mu_{ij}/\phi)$ random variables, where ϕ is the so-called overdispersion factor Var/E, generally > 1.

We compute the Chain Ladder estimates for the row and column parameters by a call to `glm`, with quasi-Poisson errors, log-link and covariates `i` and `j` as factors. Then we extract the α and β parameter estimates satisfying the corner restriction $\widehat{\beta}_1 = 1$:

```
Orig.CL <- glm(Xij~i+j, family=quasipoisson)
coefs <- exp(coef(Orig.CL)); round(coefs,4)
alpha <- c(1, coefs[2:TT]) * coefs[1]
beta <- c(1, coefs[(TT+1):(2*TT-1)])
names(alpha) <- paste0("row",1:10); round(alpha)
names(beta)  <- paste0("col",1:10); round(beta, 4)
```

Q_1 Explain the result of `sum((Xij-fitted(Orig.CL))[i==5])`. □

The fitted values, including those for future cells, are the outer product of the `alpha` and `beta` vectors. The lower right triangle of `Orig.fits` corresponds to predicted values.

```
Orig.fits <- outer(alpha, beta); round(Orig.fits)
future <- row(Orig.fits) + col(Orig.fits) - 1 > TT
(Orig.reserve <- sum(Orig.fits[future]))  ##  18680856
```

The sum 18 680 856 of the future predicted values, computed in `Orig.reserve`, represents the reserve to be kept.

Q_2 What do `row(Orig.fits)` and `matrix(as.numeric(future),10)` produce? □

Q_3 A different way of producing the fitted values uses the `predict` function of a `glm` object. What does the following script produce? Which numbers in `Xij` added up are equal to the result of the last line?

```
ij <- expand.grid(i=as.factor(1:TT),j=as.factor(1:TT))
ij[c(1,5,10,19,35,67),]
mm <- matrix(predict(Orig.CL, ij, type="response"), TT); round(mm)
sum(Xij); sum(mm[row(mm)+col(mm)-1<=TT])
sum(mm[row(mm)+col(mm)-1<=TT & row(mm)==TT-1])
```

□

Next we compute the unscaled Pearson residuals $(X - \widehat{\mu})/\sqrt{V(\widehat{\mu})}$. The variance for the Poisson family, as a function of the mean, is proportional to the mean: $\sigma^2(\mu) = \phi V(\mu) = \phi\mu$, with ϕ the dispersion parameter (overdispersion factor for the Poisson family). For other families, there is a different relation between the variance and the mean. We also compute the number of estimated parameters, the Pearson estimate $\widehat{\phi}_P$ of ϕ and the residuals, adjusted for bias:

```
Prs.resid <- (Xij - fitted(Orig.CL)) / sqrt(fitted(Orig.CL))
p <- 2*TT-1; phi.P <- sum(Prs.resid^2)/(n-p)
Adj.Prs.resid <- Prs.resid * sqrt(n/(n-p))
```

Now we are all set to start the bootstrapping process.

5 Bootstrapping

Before the bootstrapping (1000 times), we set the random seed and initialize a few arrays that we need. For the Chain Ladder results on the original triangle, see Section 4.

```
birthday <- yymmdd; set.seed(birthday) ## do adjust this line
nBoot <- 1000; payments <- reserves <- n.neg <- numeric(nBoot)
for (boots in 1:nBoot){       ## running this will take 5--10 seconds
   Ps.Xij <- sample(Adj.Prs.resid, n, replace=TRUE)           ##  1
   Ps.Xij <- Ps.Xij * sqrt(fitted(Orig.CL)) + fitted(Orig.CL) ##  2
   number.neg <- sum(Ps.Xij<0)
   Ps.Xij <- pmax(Ps.Xij, 0)  ##  Set obs < 0 to 0
   Ps.CL <- glm(Ps.Xij~i+j, family=quasipoisson)              ##  5
   coefs <- exp(as.numeric(coef(Ps.CL)))
   Ps.alpha <- c(1, coefs[2:TT]) * coefs[1]
   Ps.beta <- c(1, coefs[(TT+1):(2*TT-1)])
   Ps.fits <- outer(Ps.alpha, Ps.beta)
   Ps.reserve <- sum(Ps.fits[future])
   Ps.totpayments <- phi.P * rpois(1, Ps.reserve/phi.P)       ## 11
   reserves[boots] <- Ps.reserve                              ## 12
   payments[boots] <- Ps.totpayments; n.neg[boots] <- number.neg}
```

A line-by-line description of the statements in the for-loop (see also Section 2):

1 Draw a sample of size `n` *with replacement* from the adjusted residuals.

2–4 Next, reconstruct the payment that would have led to this residual. This is done by multiplying the residual by the s.d. and adding the mean. Sometimes negative pseudo-payments result. We resolve that problem by simply setting negative values to zero. This causes a slight bias.

5–10 On the set of pseudo-data, run a Chain Ladder `glm`, producing a set of `alpha` and `beta` values as well as to a bootstrap estimate of the reserve.

11 The extension given in England (2002) is that he proposes to construct a predictive distribution of the future payments by sampling from the estimated process (characterized by the $\vec{\alpha}$, $\vec{\beta}$ and $\phi =$ `phi.P` parameters). Because the payments are the sum of independent random variables with a quasi-Poisson(`...`, `phi.P`) distribution, their total is a drawing from a quasi-Poisson(`sum(...)`, `phi.P`) distribution. Otherwise, we would have to draw a value in every cell (i, j) and add them up.

12–13 Return the results of the bootstrap simulation.

Q_4 How many negative pseudo-observations resulted, out of how many generated? □

The prediction error is the square root of the process variance plus the estimation variance, as follows:

```
(PEbs <- sqrt(phi.P*Orig.reserve + sd(reserves)^2))  ##  ~ 3000000
```

The ratio of estimation variance to process variance is:

```
sd(reserves)^2 / (phi.P*Orig.reserve)       ##  ~ 8 for these data
```

Some relevant quantiles, expressed in millions for readability, are produced by

```
payments <- payments/1e6
round(quantile(payments, c(0.5,0.75,0.9,0.95,0.99)), 1)
## output should resemble:
##  50%  75%  90%  95%  99%
##  19   21   23   24   27
```

Other statistics: mean, s.d. and s.d. / mean = coefficient of variation:

```
mean(payments)                          ##  ~ 19 million
sd(payments)                            ##  ~ 3 million
100 * sd(payments) / mean(payments)     ##  c.v. ~ 15%
```

Estimates of the skewness and the (excess) kurtosis, of the future payments:

```
pp <- (payments-mean(payments))/sd(payments)
sum(pp^3)/(nBoot-1)       ##  ~ .4    estimates the skewness
sum(pp^4)/(nBoot-1) - 3   ##  ~~ .4   estimates the kurtosis
```

A histogram of these payments, together with a smooth density estimate (kernel density estimate; see `?density` as well as MART, p. 334), and a simple normal density plot with the same mean and s.d. is produced by:

```
hist(payments,breaks=21,prob=T)
lines(density(payments), lty="dashed", col="blue")
curve(dnorm(x, mean(payments), sd(payments)), lty="dotted", add=T, col="red")
```

Q_5 Use `mean(payments)` and `quantile(payments, c(....))` to verify the statements in Remark 10.6.1. □

In view of the enormous volatility, it is hard to substantiate a claim that any reserving method gives 'superior' predictions. Other things to be considered are:

- easy access to software;
- the possibility to generate convincing and revealing plots easily;
- control over what one is doing;
- adaptability to new rules and situations.

All these are not present in an ill-documented black box method.

We have used `R` here, because it is very powerful and it follows closely the formulas of the theory, making checking the code very easy. England and Verrall have implemented their method in Excel and Visual Basic.

Q_6 = **10.6.1** Adapt the method above to construct a predictive distribution of the IBNR-reserve to be held using a *gamma* error distribution instead of Poisson. Compare the resulting histograms, as well as the first three estimated cumulants of the resulting distributions.

First do a `glm`-fit explaining `Xij` by `i+j` using a Gamma family with `fam=Gamma(link=log)`. Recall the upper case G needed to specify the Gamma family rather than the gamma-function (factorials)!

Hints for changes to the bootstrapping for-loop:

- Line 2: account for $V(\mu) = \mu^2$ rather than $V(\mu) = \mu$ such as with Poisson (see also England, P. and Verrall, R. (1999).
- Line 4: since the Gamma-family does not tolerate zero observations, change values less than zero to a small positive number like 0.01 .
- In line 5, use the Gamma family rather than (quasi-)Poisson.
- Line 11: you have to do the following:
  First compute the vector `vec.Ps.fits` of the future fitted values $\widehat{\mu}$ and store its length in `h`. Then draw the payments for each of the `h` cells from a gamma density (see `?rgamma`), and add them all up. Shape α and rate β used for the drawing must be specified in such a way that the means are the elements of the vector `vec.Ps.fits`, and also that the variances satisfy the relation $\sigma^2 = \phi\mu^2$. Because mean and variance of a gamma(α, β) r.v. are α/β and α/β^2 respectively, this leads to the reparameterization $\alpha = 1/\phi$, $\beta = 1/(\mu\phi)$ in Section 9.2; see also Table D as well as (2.8) on p. 283 in England, P. and Verrall, R. (1999). Note that in the Poisson case, we were able to take one drawing with parameter $\sum \widehat{\mu}_{ij}/\widehat{\phi}$ to simulate the entire future, but here, since the β's are not constant, the total prediction does not have a simple distribution. A gamma random variable has to be generated for each cell, rather than for all cells combined such as was possible in the quasi-Poisson case.

The final results of the procedure should not deviate too much from the Poisson results, cf. England, P. and Verrall, R. (1999), Tables 1 and 2. □

Q_7 Verify if the parameters α_i found in the Gamma-glm satisfy the DM-equations MART (9.23). Here $w_{ij} = 1$ for 'past' (i, j), 0 otherwise. So $w_{i\Sigma}$ is just the number of elements in `Xij` with row number i. Compute the relative differences ($\times 10^6$) between the lhs and the rhs of (9.23) $\alpha_i = \sum_j w_{ij} \frac{y_{ij}}{\beta_j} / w_{i\Sigma}$. □

6 England/Verrall's analytical estimate of the MSPE

For Exercise 10.6.3, we turn to the analytical estimate of the MSPE in England and Verrall (1999). In MART Sec. 10.6.2, see also slides 10.39–41, it was derived that the MSE of the prediction $\widehat{R} = \sum \widehat{X}_{ij}$ for future totals $R = \sum X_{ij}$ (with (i, j) 'future') can be approximated as:

$$\mathrm{E}[(R - \widehat{R})^2] \approx \widehat{\phi} \sum \widehat{\mu}_{ij} + \widehat{\mu}\,' \, \widehat{\mathrm{Var}}[\widehat{\eta}] \, \widehat{\mu}.$$

This expression can be computed using only by-products of a `glm` call in `R`:

1. for $\widehat{\phi}$, take deviance / df,
2. $\widehat{\mu}$ is the vector of fitted values $\widehat{\mu}_{ij}$,
3. an estimate $\widehat{\mathrm{Var}}[\widehat{\beta}]$ of the (co-)variance matrix of the $\widehat{\beta}$'s is found using `vcov()`, and
4. $\widehat{\mathrm{Var}}[\widehat{\eta}] = \mathbf{X}\widehat{\mathrm{Var}}[\widehat{\beta}]\mathbf{X}'$, since $\widehat{\eta} = \mathbf{X}\widehat{\beta}$ with $\mathbf{X}$ the model matrix.

The IBNR-triangle in Section 3 was not padded with zeroes for future years, but we use a simple trick to convert the IBNR-triangle to a square matrix. It applies the `xtabs` function to compute for each (i, j) value the total of the `Xij` values corresponding to these values of `i` and `j`; of course there is only one such entry in the dataset. Row numbers of this square matrix are stored in factors `ii` and `jj` below; `future` is `TRUE` for those entries in `Xij.1` corresponding to a future value. At the end we estimate (again) a Chain ladder model. To eliminate the future zeroes from the estimation, we give them weight zero.

```
Xij.1 <- as.vector(t(xtabs(Xij~i+j))) ## stored row-wise as usual
ii <- rep(1:TT, each=TT); jj <- rep(1:TT, TT); future <- ii+jj-1 > TT
ii <- as.factor(ii); jj <- as.factor(jj)
Orig.CL <- glm(Xij~i+j, family=quasipoisson, epsilon = 1e-12)
CL <- glm(Xij.1~ii+jj, fam=quasipoisson, wei=as.numeric(!future))
```

Q_8 To see if 'weighting out' unobserved cells produces the same results as not including them, compare `CL` and `Orig.CL`. □

Q_9 Several ways to find an estimate $\widehat{\phi}$ are given below. Which ones are the mean deviance estimate and the Pearson estimate?

```
p <- 2*TT-1
phi.P <- sum((Xij - fitted(Orig.CL))^2 / fitted(Orig.CL))/(n-p) ## 1
phi <- CL$deviance/CL$df.residual ## 2
c(phi, phi.P) ## 52861.5  52601.4
sum(resid(CL)^2)/(n-p) ## 3; "type=devi" is default
sum(resid(CL,type="pear")^2)/(n-p) ## 4
Prs.resid <- (Xij.1 - fitted(CL)) / sqrt(fitted(CL))
sum(as.numeric(!future)*Prs.resid^2)/(n-p) ## 5
dev.resid2 <- 2 * (Xij.1*log(ifelse(Xij.1==0, 1, Xij.1/fitted(CL))) -
  (Xij.1 - fitted(CL))) ##  cf. (9.29)
sum(as.numeric(!future)*dev.resid2)/(n-p) ## 6
summary(CL)$dispersion ## 7 (the warning is reassuring)
summary(Orig.CL)$dispersion ## 8
```
□

Now construct the vector of predictions, with zeroes for past observations. Then, to find the estimated variance matrix of the linear predictors $\eta = \mathbf{X}\beta$, first retrieve the estimated covariance matrix of the coefficients. Ignore the warning about zero weight observations not being used for computing dispersion, that is just as intended. Store the design matrix of regressors in `X`.

```
mu.hat <- fitted(CL)*future
Cov.beta <- vcov(CL)
X <- model.matrix(CL)
```

This is how an estimate for the covariance matrix of the $\widehat{\eta}$'s is obtained:

```
Cov.eta <- X %*% Cov.beta %*% t(X)
```

Finally construct the MSPE by adding process error and parameter error:

```
MSPE <- phi * sum(mu.hat) + t(mu.hat) %*% Cov.eta %*% mu.hat
cat("Total reserve =", round(sum(mu.hat)), "p.e. =", round(sqrt(MSPE)), "\n")
```

This results in: `Total reserve = 18680856 p.e. = 2946484`

Now all objects are available for you to make the following exercise.

Q_{10} **= 10.6.3** England and Verrall (1999) in their formula (3.4) also compute the contribution to the total prediction error of each separate year of origin `i`. Essentially, (3.4) equals (3.5) with the summations restricted to the corresponding row of the IBNR predicted triangle. This can be easily achieved by using the same code used for implementing formula (3.5), but now with the $\widehat{\mu}_{kj}$ replaced by zero for predictions of origin years other than `k=i`. Reproduce the 3rd column of Tables 1 and 2 in England and Verrall (1999), p. 288.

Hint: use a construction like below.

```
for (r in 2:TT){
  mu.r <- ....  ## replace the elements of mu.hat not having rownr==r by 0
  MSPE <- ....; res <- ....  ## see above
  cat("Year =", r, "\treserve =", round(res),
   "\tp.e./res. =", round(100*sqrt(MSPE)/res), "%\n") }
```
□

7 IBNR-problems with known exposures

In an IBNR triangle, often some measure, n_i, say, for the *exposure* is known for the years of origin i. One can use the number of policies, or e.g. the premium income. Or, the actuary might have a good prior idea about what is known as the *schedule* or *budget ultimate losses*. In that case, instead of the parameters α_i in the CL method, one may use these numbers n_i, or rather, $e_i = R_1 n_i/n_1$, with R_1 the fully observed total paid for accident year 1. In the Bornhuetter-Ferguson method, one then predicts future claims using $e_i\widehat{\beta}_j$, with $\widehat{\beta}_j$ obtained by the CL method.

One might also include these e_i in an offset term and *optimally* estimate development factors $\widehat{\beta}_j$. Then one gets a model restricted to have $\alpha_i/\alpha_1 = n_i/n_1$, so with about half as many parameters as CL itself. A test will show if the loss of deviance is significant.

Using portfolio size to replace row parameters

We look at the same run-off 8×8 triangle we used to illustrate the calendar year effect in the Arithmetic Separation method. As usual, we clean up, read the data and construct the covariates:

```
rm(list=ls(all=TRUE)); Xij <- scan(n=36)
156  37   6   5   3   2   1   0
154  42   8   5   6   3   0
178  63  14   5   3   1
198  56  13  11   2
206  49   9   5
250  85  28
252  44
221
TT <- 8; i <- rep(1:TT, TT:1); j <- sequence(TT:1); k <- i+j-1
fi <- as.factor(i); fj <- as.factor(j); fk <- as.factor(k)
```

The portfolio sizes for each year (row) are known. The first 8 entries in `Xij` correspond to year 1 and have exposure 28950, the next 7 arose from 29754 policies, and so on. So do:

```
ee <- c(28950,29754,26315,39442,38423,50268,44762,43541)
Expo <- rep(ee, TT:1)
```

Q_{11} Explain why `all(Expo == ee[i])` results in `TRUE`, providing an alternative way to find the exposure associated to each observation in `Xij`. □

We want to compare the Chain Ladder and the Threeway models we studied earlier with models in which the effect of portfolio growth is not estimated into factors α_i, but simply taken proportional to exposure e_i. Using a log-link, we can make the predicted means proportional to the exposures by adding the logs of these numbers to the linear predictors, using the *offset* mechanism.

In this assignment, we don't assume the observations to be Poisson (integer observations with variance equal to mean), but merely quasi-Poisson (variance equals mean times scale factor ϕ). To be able to analyze the scaled deviance rather than just the deviance, we use the mean deviance of the 'fullest' model available to estimate ϕ.

Q_{12} Estimate Chain Ladder (year of origin and development year) and Exposure model (exposure and development year). Perform an analysis of scaled deviance by filling in the dots in the script below. What restriction does `EE` impose on the coefficients of `CL`? Which model comes out 'better'?

```
CL <- glm(Xij~...., quasipoisson) ## the Chain ladder model
EE <- glm(....) ## the Exposure model
scale <- ....$deviance / ....$df.residual ## mean-deviance estimate for phi
Delta.Dev.Sc <- .... ## difference of scaled deviances for CL and EE
Delta.df <- .... ## difference of degrees of freedom for CL and EE
reject <- .... ## TRUE if Delta.Dev.Sc > the chi^2 critical value
cat("The exposure model", ifelse(reject, "is", "is not"), "rejected",
 "since the scaled deviance gained by CL is\n",
 round(Delta.Dev.Sc,1), "with", Delta.df, "extra parameters.\n")
```

□

Now inspect the percentage differences in fitted values produced by CL and EE and the resulting coefficients:

```
xtabs(round(100*(fitted(CL) - fitted(EE))/fitted(CL))~i+j)
round(coef(CL),2); round(coef(EE),2)
```

This shows that except for row 3, the models are rather similar as regards fitted values. The column coefficients are very similar, the row coefficients are not present for the offset model, and the intercept is very different.

Q_{13} To understand how the fitted values can still be similar, construct and inspect vectors `alpha`, `beta`, `M` and `delta` such that `alpha` and `M` represent row totals, while `beta` and `delta` can be identified as fractions paid in each development year, so

- the fitted values for `CL` are `alpha[ii] * beta[jj]` for cell `(ii,jj)`
- the fitted values for `EE` are `M[ii] * delta[jj]`
- `sum(beta) = sum(delta) = 1`

□

Three-way models

Next, we compare the three-way exposure model $\mu_{ij} = e_i\beta_j\gamma_k$ and the three-factor model $\mu_{ij} = \alpha_i\beta_j\gamma_k$, or equivalently, $\mu_{ij} = e_i\alpha_i\beta_j\gamma_k$:

```
Three.off <- glm(Xij~offset(log(Expo))+fj+fi+fk, quasipoisson)
anova(Three.off, test="Chisq")
#      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
```

```
# NULL                    35    3098.22
# fj    7   3041.5        28       56.7   <2e-16 ***
# fi    7     22.6        21       34.2   0.0315 *
# fk    6     12.9        15       21.3   0.1856
```

The string of models for the mean of observation (i, j) that we look at is $\mu e_i \to \mu e_i \beta_j \to \mu e_i \beta_j \alpha_i \to \mu e_i \beta_j \alpha_i \gamma_k$. Note that the equivalent models `CL` and `offset(log(Expo))+fj+fi` indeed have the same deviance and df.

Q_{14} Repeat the `anova` call above, but now without the last model term `+fk`. Explain how it can happen that in the deviance analysis of the smaller models, the exposure model is not rejected at 5% level in favor of `CL`, but when `fk` is included in the analysis, it is. □

It turns out that, as expected, coefficients for the development year cannot be missed. On top of development year and exposure, it proves that the significance of year of origin is doubtful, and calendar year is not a significant predictor.

We can inspect the parameter estimates:

```
options(digits=7); summary(Three.off)
#               Estimate Std. Error t value Pr(>|t|)
# (Intercept)   -5.22347    0.09700 -53.850  < 2e-16 ***
# fj2           -1.33452    0.07207 -18.518 9.60e-12 ***
# fj3           -2.76272    0.14682 -18.817 7.62e-12 ***
# fj4           -3.46343    0.23298 -14.866 2.20e-10 ***
# fj5           -4.05421    0.34344 -11.805 5.41e-09 ***
# fj6           -4.58245    0.51255  -8.941 2.13e-07 ***
# fj7           -5.92674    1.21829  -4.865 0.000206 ***
# fj8          -22.35244 4201.43685  -0.005 0.995825
# fi2            0.10724    0.16931   0.633 0.536004
# fi3            0.40860    0.17493   2.336 0.033800 *
# fi4           -0.08240    0.17121  -0.481 0.637259
# fi5           -0.12257    0.17055  -0.719 0.483385
# fi6           -0.00432    0.16387  -0.026 0.979318
# fi7           -0.21568    0.15830  -1.362 0.193160
# fi8           -0.05983    0.12669  -0.472 0.643568
# fk2           -0.13933    0.18226  -0.764 0.456456
# fk3           -0.17866    0.18643  -0.958 0.353091
# fk4            0.02194    0.17497   0.125 0.901896
# fk5            0.10201    0.16601   0.614 0.548098
# fk6           -0.04999    0.15420  -0.324 0.750267
# fk7            0.23054    0.13930   1.655 0.118689
# fk8                 NA         NA      NA       NA
```

Note that no coefficients for `fj1`, `fi1` and `fk1` are listed; they are zero because of corner restrictions. The coefficient with `fk8` was not defined because of singularity: `fk8` is expressible as a linear form of earlier columns in the design matrix (see MART ex. 10.4.2 and 10.4.7).

Q_{15} What happens in year of development $j = 8$? □

Q_{16} The three-way model $\mu_{ij} = \alpha_i\beta_j\gamma_{i+j-1}$ and the one with an offset-term $\mu_{ij} = e_i\alpha_i\beta_j\gamma_{i+j-1}$ generate the same fitted values. If you look at the coefficients, you'll see that they are the same for the levels of j and k. The coefficients for levels of i differ.

The intercepts are `-5.22347` and `+5.04986`, respectively. Do they indeed lead to the same fitted value for the top-left observation? Hint: what is `exp(+5.04986) / exp(-5.22347)`?

Also check if the fitted values coincide for year of origin 2 and development year 1, governed by $\alpha_2 = 0.13463$, $\beta_1 = 0$ and $\gamma_2 = -0.13933$ in the model without offset. □

Q_{17} We want to try models in which the effect of year of origin is captured by the known portfolio sizes except for year 3, where apparently some outside effect resulted in much higher claims than exposure suggests. One way to do this is to create a dummy variable denoting membership of year origin 3, e.g. by `i.is.3 <- as.numeric(i==3)`. Note the *double* equality sign. Analyze the deviances for model formula `offset(log(Expo))+fj+i.is.3+fi`.

Somebody suggests that the exposure for that year was not `26315` but should have been `36315`. After this correction, is the adjustment $\widehat{\alpha}_3$ for year 3 still significant?
Hint: repeat the anova call with a new exposure constructed as
`Expo1 <- c(28950,29754,36315,39442,38423,50268,44762,43541)[i]` □

8 The Bornhuetter-Ferguson method

Use again the following example, where the `ee` vector is not filled with the exposures used above but rather with exposures adjusted by the actuary, with the same values as used in the book:

```
rm(list=ls(all=TRUE)) ## Discard old garbage
Xij <- scan(n=36)
156 37  6  5 3 2 1 0
154 42  8  5 6 3 0
178 63 14  5 3 1
198 56 13 11 2
206 49  9  5
250 85 28
252 44
221
TT <- 8; i <- rep(1:TT, TT:1); j <- sequence(TT:1); k <- i+j-1
fi <- as.factor(i); fj <- as.factor(j); fk <- as.factor(k)
ee <- c(28950,29754,31141,32443,34700,36268,37032,36637)
Expo <- rep(ee, TT:1)
CL <- glm(Xij~fi+fj, quasipoisson)
EE <- glm(Xij~offset(log(Expo))+fj, quasipoisson)
```

Q_{18} Compute the past and future fitted totals in the Chain Ladder model for the data of this example; you should get the values 2121 and 152.0.

Why does the total of the fitted values for past observations equal `sum(Xij)`? □

Q_{19} Run the following statement and explain the result:

```
round(tapply(fitted.values(EE)-Xij,j,sum),6)
```
□

Q_{20} Fill in the dots below to find out what happens with the reserves according to chain ladder if the number 221 in the last row is replaced by 171,181,...,271.

Plot the resulting reserves against the replacing value.

```
reserves <- numeric(); lasts <- c(171,181,191,201,211,271,261,251,241,231,221)
for (last in lasts)
{ Xij[36] <- last
  cc <- exp(coef(glm(Xij~fi+fj,quasipoisson)))
  alpha <- c(1,cc[2:TT])*cc[1]; beta <- c(1,cc[(TT+1):(2*TT-1)])
  fits <- (alpha %o% beta)
  reserve <- .... ## the sum of the 'future' fitted values
  reserves <- c(reserves, reserve)  }
rbind(lasts, reserves=round(reserves))
plot(lasts, reserves); lines(range(lasts),range(reserves))
```
□

Q_{21} From the plot just made you see that the relation between reserve and value of the last element is exactly linear. Can you explain that?
Hint: check what happens in the CL-algorithm (10.3) in MART. □

Instead of $\vec{\alpha}\vec{\beta}'$, the Bornhuetter-Ferguson method uses $\vec{M}\vec{\beta}'$ to compute the reserves, where $\vec{M}$ is the vector of relative weights (normed s.t. $M_1 = \alpha_1$) for the various years of origin. In this case, the actuary's prior belief is that the total losses for each year of origin will be proportional to the numbers stored in the vector `ee`. In practice, the coefficients β_j are computed using Verbeek's algorithm or by development factors.

Q_{22} How is the vector $\vec{M}$ constructed? Store it in `M`. □

Q_{23} Compare the reserves to be kept, by year of origin, for these two methods. Fill in the dots:

```
pred.CL <- alpha %*% t(beta); round(pred.CL, 4)
pred.BF <- ....; round(pred.BF, 4)
future <- ....; reserve.CL <- ....; reserve.BF <- ....
```
□

Q_{24} The total 'retrofitted' values are found by, e.g., `sum(pred.BF * (1-future))` for the BF predictions. Note that the BF-retrofitted values do not satisfy the property that their total is the observed total `sum(Xij)`. Why not? □

Q_{25} Taking coefficients proportional to the exposure for the rows, just as earlier in this assignment, we can choose optimal coefficients for the columns. Fill in the dots:

```
CLoff <- .... ## The model with log(exposures) as offset
```
□

To see what difference this makes for the reserves by year of origin, do

```
cc <- exp(coef(CLoff))
betaoff <- cc[1] * c(1,cc[2:TT])
pred.off <- ee %*% t(betaoff); round(pred.off, 4)
round(cbind(rowSums(pred.BF * future),
         rowSums(fits * future),
         rowSums(pred.off * future)), 1)
sum(pred.off * future)
```

Q_{26} Explain why the retrofitted values again sum to `sum(Xij)`:

```
sum(pred.off * (1-future)); sum(Xij)
```

□