Non-Life Insurance — Assignment 5

AJ*

Autumn 2021

Question 1

We first check that we have constructed the IBNR triangle correctly, having used the data of Taylor & Ashe (1983) :

```
> xtabs(Xij~i+j)
    j
i          1       2       3       4       5       6       7       8       9      10
  1   357848  766940  610542  482940  527326  574398  146342  139950  227229   67948
  2   352118  884021  933894 1183289  445745  320996  527804  266172  425046       0
  3   290507 1001799  926219 1016654  750816  146923  495992  280405       0       0
  4   310608 1108250  776189 1562400  272482  352053  206286       0       0       0
  5   443160  693190  991983  769488  504851  470639       0       0       0       0
  6   396132  937085  847498  805037  705960       0       0       0       0       0
  7   440832  847631 1131398 1063269       0       0       0       0       0       0
  8   359480 1061648 1443370       0       0       0       0       0       0       0
  9   376686  986608       0       0       0       0       0       0       0       0
  10  344014       0       0       0       0       0       0       0       0       0
```

We then compute the chain ladder estimates and verify that the model satisfies the *marginal totals property*:

```
## rows marginal totals
> (tapply(Xij, i, sum) - tapply(fitted(Orig.CL), i, sum))/tapply(Xij, i, sum)
            1             2             3             4             5             6
-4.559358e-11 -3.852987e-11 -4.286977e-11 -6.457354e-11 -4.379618e-11 -1.384984e-13
            7             8             9            10
-1.684500e-14 -1.398041e-14 -1.468754e-14 -1.235170e-14

## columns marginal totals
> (tapply(Xij, j, sum) - tapply(fitted(Orig.CL), j, sum))/tapply(Xij, j, sum)
            1             2             3             4             5             6
-7.105315e-13 -7.610457e-13 -8.601956e-13 -1.093003e-12 -2.736024e-12 -5.004009e-10
            7             8             9            10
-6.888954e-11 -4.344418e-13 -6.646440e-13 -1.349224e-14
```

*Student number: ∞

Question 2

`row(Orig.fits)` and `col(Orig.fits)` return matrices that have entries corresponding to the row or column index of that entry. `future` is a logical vector which identifies the entries that are below the diagonal of the matrix.

Question 3

The number of negative pseudo-observations is given by:

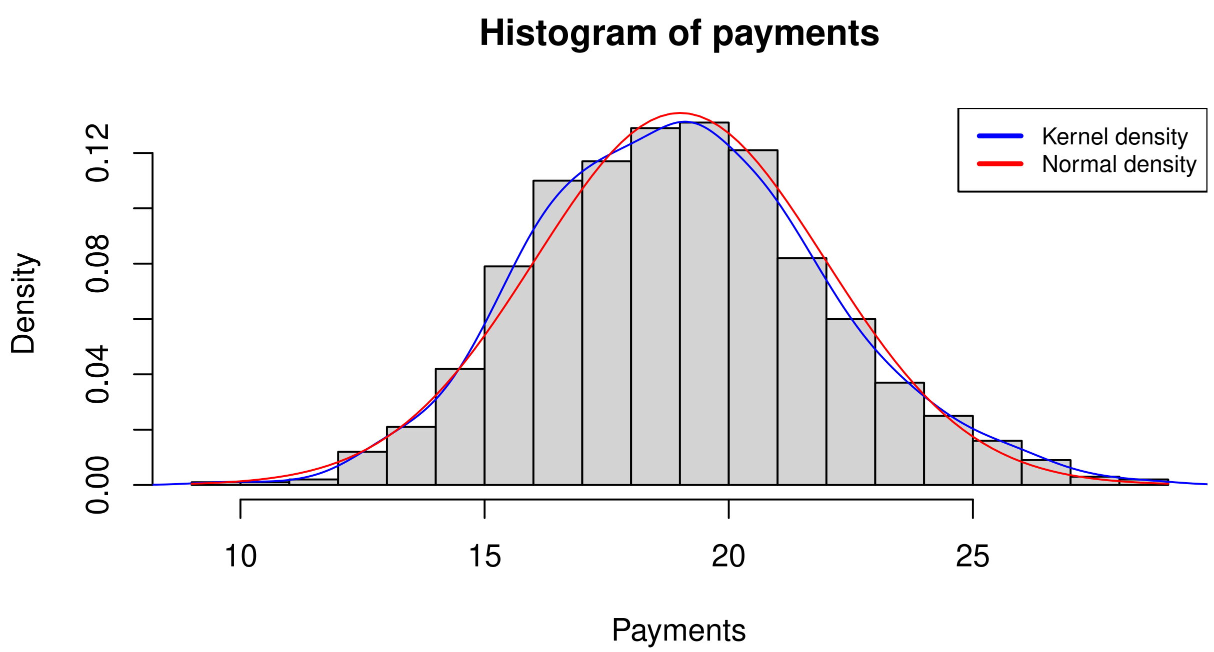
```
> sum(n.neg)
[1] 117
```

from a total of `55*1000` $= 55\,000$ generated pseudo-observations.

Question 4

Before computing the empirical quantiles, the following visualisation is useful:

```
## kernel density estimate
hist(payments, breaks=21, prob=TRUE, xlab = "Payments")
lines(density(payments), col="blue")
curve(dnorm(x, mean(payments), sd(payments)), add=TRUE, col="red")

## add
legend("topright", c("Kernel density","Normal density"),
       lty=c(1,1), lwd=c(2.5,2.5), col=c("blue" ,"red"),
       cex = .75)
```



Now, the empirical quantiles are given by:

```
> quantile(payments, c(.5, .75, .9, .95, .99))
     50%      75%      90%      95%      99%
18.93649 20.93534 22.82899 24.09668 26.30068
```

We see that the generated quantiles are close to those reported in the first paragraph of Remark 10.6.1 in MART.

Question 5

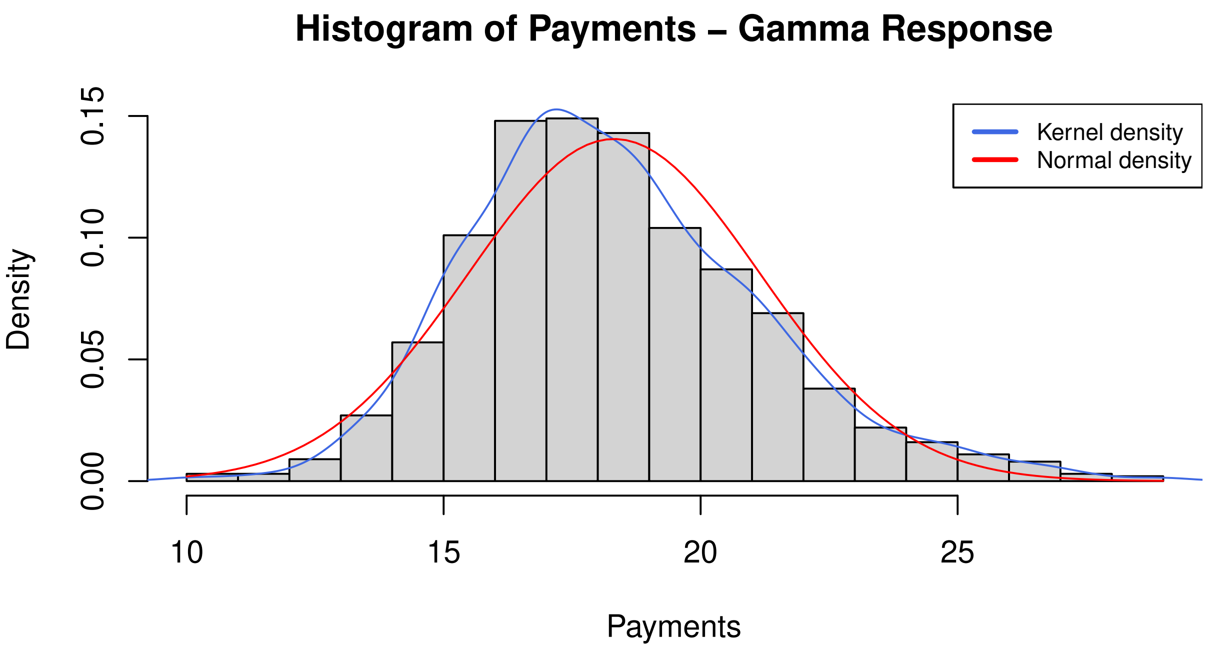
```
## Gamma Response Model

## (a) Fit:
Orig.gam <- glm(Xij~i+j, family = Gamma(link=log))

## (b) Extract coefficients:
coefs <- exp(coef(Orig.gam))
alpha <- c(1, coefs[2:TT]) * coefs[1]
beta <- c(1, coefs[(TT+1):(2*TT-1)])
names(alpha) <- paste0("row",1:10)
names(beta)  <- paste0("col",1:10)

## Compute reserves:
Orig.fits <- alpha %o% beta; round(Orig.fits)
future <- row(Orig.fits) + col(Orig.fits) - 1 > TT
Orig.reserve <- sum(Orig.fits[future])

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

continued on next page.

```
## (d) Adjusted bootstrap:
set.seed(174297)
nBoot <- 1000; payments.gam <- reserves.gam <- n.neg <- numeric(nBoot)
for (boots in 1:nBoot){
  Ps.Xij <- sample(Adj.Prs.resid, n, replace=TRUE)
  Ps.Xij <- Ps.Xij * fitted(Orig.gam) + fitted(Orig.gam)
  number.neg <- sum(Ps.Xij<0)
  Ps.Xij <- pmax(Ps.Xij, 0.01)
  Ps.gam <- glm(Ps.Xij~i+j, family = Gamma(link=log))
  coefs <- exp(as.numeric(coef(Ps.gam)))
  Ps.alpha <- c(1, coefs[2:TT]) * coefs[1]
  Ps.beta <- c(1, coefs[(TT+1):(2*TT-1)])
  Ps.fits <- Ps.alpha %o% Ps.beta
  Ps.reserve <- sum(Ps.fits[future])
  h <- length(Ps.fits[future])
  Ps.payments <- rgamma(h, shape=(1/phi.P), rate=(1/(Ps.fits[future]*phi.P)))
  Ps.totpayment <- sum(Ps.payments)
  reserves.gam[boots] <- Ps.reserve
  payments.gam[boots] <- Ps.totpayment
  n.neg[boots] <- number.neg
}

## (e) Statistics (in millions)
pp <- payments-mean(payments)
skew <- mean(pp^3)/sd(payments)^3
kurt <- mean(pp^4)/mean(pp^2)^2 - 3

payments.gam <- payments.gam/1e6
pp.gam <- payments.gam - mean(payments.gam)
skew.gam <- mean(pp.gam^3)/sd(payments.gam)^3
kurt.gam <- mean(pp.gam^4)/mean(pp.gam^2)^2 - 3

## comparison
> rbind(c(mean(payments), sd(payments), skew, kurt),
+       c(mean(payments.gam), sd(payments.gam), skew.gam, kurt.gam))
         [,1]     [,2]      [,3]       [,4]
[1,] 19.00545 2.966485 0.2107881 0.02943967
[2,] 18.31776 2.838490 0.5545964 0.54901357
```

continued on next page.

```
## 5e. continued:
## kernel density estimate
hist(payments.gam, breaks=21, prob=TRUE, xlab = "Payments",
     main = "Histogram of Payments - Gamma Response")
lines(density(payments.gam), col="royalblue")
curve(dnorm(x, mean(payments.gam), sd(payments.gam)), add=TRUE, col="red")

## add
legend("topright", c("Kernel density","Normal density"),
       lty=c(1,1), lwd=c(2.5,2.5), col=c("royalblue" ,"red"),
       cex = .75)
```



Now, the new empirical quantiles are given by:

```
> quantile(payments.gam, c(.5, .75, .9, .95, .99))
     50%      75%      90%      95%      99%
18.01098 20.04397 21.99663 23.41159 26.27565
```

For ease of reference, recall our earlier quantiles:

```
> quantile(payments, c(.5, .75, .9, .95, .99))
     50%      75%      90%      95%      99%
18.93649 20.93534 22.82899 24.09668 26.30068
```

The notable differences are that the mean of the Gamma response is lower and this model exhibits far more skewness and kurtosis.

Question 6

Note: to simplify indexing, this implementation creates a run-off triangle from the data with zeros for future observations, stored as a `matrix` object.

```
Xij <- as.matrix(xtabs(Xij~i+j))

alpha.dm <- numeric(TT)
for (k in 1:TT){
  alpha.dm[k] <- sum(Xij[k,]/beta) / rowSums(!future)[k]
}

> round(alpha - alpha.dm, 3)
  row1   row2   row3   row4   row5   row6   row7   row8   row9  row10
-0.915  0.721  1.009  0.004 -0.316 -0.146 -0.023  0.221  0.018  0.000

beta.dm <- numeric(TT)
for (z in 1:TT){
  beta.dm[z] <- sum(Xij[,z]/alpha) / colSums(!future)[z]
}

> round(beta - beta.dm, 3)
 col1  col2  col3  col4  col5  col6  col7  col8  col9 col10
    0     0     0     0     0     0     0     0     0     0
```

Therefore, it is indeed the case that the parameters `alpha` and `beta` extracted from `Orig.gam` satisfy the *Direct-Method* equations.

Question 7

```
## (a)
> Xij[i==TT]
[1] 344014

> Xij.1[ii==TT]
 [1] 344014      0      0      0      0      0      0      0      0      0
```

Yes, the output looks as we'd expect as our goal was to pad the final row with nine zeros which is what we have achieved.

```
## (b)
> coef(CL); coef(Orig.CL)
 (Intercept)          ii2          ii3          ii4          ii5          ii6
12.506404677  0.331272153  0.321118578  0.305960003  0.219316314  0.270077015
         ii7          ii8          ii9         ii10          jj2          jj3
 0.372208424  0.553333059  0.368934194  0.242032956  0.912526274  0.958830628
         jj4          jj5          jj6          jj7          jj8          jj9
 1.025997003  0.435276183  0.080056547 -0.006381469 -0.394452205  0.009378211
        jj10
-1.379906692

 (Intercept)           i2           i3           i4           i5           i6
12.506404677  0.331272153  0.321118578  0.305960003  0.219316314  0.270077015
          i7           i8           i9          i10           j2           j3
 0.372208424  0.553333059  0.368934194  0.242032956  0.912526274  0.958830628
          j4           j5           j6           j7           j8           j9
 1.025997003  0.435276183  0.080056547 -0.006381469 -0.394452205  0.009378211
         j10
-1.379906692

> CL$deviance; Orig.CL$deviance
[1] 1903014
[1] 1903014
```

Clearly, weighting out unobserved cells produces the same fitting results.

Question 8

The equivalent methods can be separated as follows:

Mean-deviance estimate $\{1, 3, 5\}$

Pearson estimate $\{2, 4, 6\}$.

Question 9

```
## Chain Ladder (CL) vs. Exposure Model(EE)

CL <- glm(Xij ~ fi + fj, quasipoisson)
EE <- glm(Xij ~ offset(log(Expo)) + fj, quasipoisson)
phi <- CL$deviance / CL$df.residual
Delta.Dev.Sc <- (EE$deviance - CL$deviance)/phi
Delta.df <- EE$df.residual - CL$df.residual
reject <- Delta.Dev.Sc > qchisq(.95, Delta.df)

> 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")

The exposure model is rejected since the scaled deviance gained by CL is
30 with 7 extra parameters.
```

Question 10

```
## consider a new model
CL.off <- glm(Xij~offset(log(Expo))+fi+fj, quasipoisson)

## (a) Fitted value cell (1,1)
> exp(coef(CL))[1]
    149.206
> exp(coef(CL.off))[1] * ee[1]
    149.206

## (b) Fitted value cell (2,1)
> exp(coef(CL))[1] * exp(coef(CL))[2]
   154.8901
> exp(coef(CL.off))[1] * exp(coef(CL.off))[2] * ee[2]
   154.8901
```

Clearly, the fitted values are the same for both models.

Question 11

Please note: in this question, all χ^2_k critical values are taken at the $\mathbf{95^{th}}$ **percentile**.

```
## (a)
> anova(EE, EE.adj)
Analysis of Deviance Table

Model 1: Xij ~ offset(log(Expo)) + fj
Model 2: Xij ~ offset(log(Expo)) + i.is.3 + fj
  Resid. Df Resid. Dev Df Deviance
1        28     82.885
2        27     35.011  1   47.874
```

Recall out estimate of $\phi = 1.626585$ from question 9.

The decrease in *scaled* deviance is:

$$\frac{47.874}{1.626585} = 29.43221$$

Comparing this with the critical value of $\chi^2_1 = 3.841$, we reject the null and go with the finer model `EE.adj`.

```
## (b)
> anova(EE.adj, CL)
Analysis of Deviance Table

Model 1: Xij ~ offset(log(Expo)) + i.is.3 + fj
Model 2: Xij ~ fi + fj
  Resid. Df Resid. Dev Df Deviance
1        27     35.011
2        21     34.158  6  0.85293
```

Noting that the critical value of $\chi^2_6 = 12.59159$, we do not even need to compute the change in *scaled* deviance to conclude that the observed change is not significant. Therefore, the `EE.adj` model is preferred.

```
## (c) EE.adj.
```

Question 12

```
## models for adjusted exposure
EE.typo <- glm(Xij ~ offset(log(Expo1)) + fj, quasipoisson)
EE.adj.typo <- glm(Xij~offset(log(Expo1))+i.is.3+fj, quasipoisson)

> anova(EE.typo, EE.adj.typo)
Analysis of Deviance Table

Model 1: Xij ~ offset(log(Expo1)) + fj
Model 2: Xij ~ offset(log(Expo1)) + i.is.3 + fj
  Resid. Df Resid. Dev Df Deviance
1        28     37.932
2        27     35.011  1   2.9203
```

Changing the exposure for year 3, we now see that the change in deviance is not significant, so the change in *scaled* deviance certainly is not. Therefore, the preferred model is now `EE.typo`.

```
> anova(EE.typo, CL)
Analysis of Deviance Table

Model 1: Xij ~ offset(log(Expo1)) + fj
Model 2: Xij ~ fi + fj
  Resid. Df Resid. Dev Df Deviance
1        28     37.932
2        21     34.158  7   3.7733
```

We have $\chi^2_7 = 14.06714$, and by the above-mentioned logic we see that `EE.typo` is the preferred model *i.e.* the model with column parameters and offset by exposure.

Question 13

```
M <- ee * alpha[1] / ee[1]

> CL.fits <- alpha %o% beta; round(CL.fits, 2)
           fj2   fj3   fj4  fj5  fj6  fj7 fj8
    149.21 40.24 10.03 5.20 3.01 1.82 0.49   0
fi2 154.89 41.78 10.41 5.40 3.13 1.89 0.51   0
fi3 188.01 50.71 12.63 6.55 3.80 2.29 0.62   0
fi4 201.15 54.26 13.52 7.01 4.06 2.45 0.66   0
fi5 196.10 52.89 13.18 6.84 3.96 2.39 0.64   0
fi6 271.52 73.24 18.24 9.46 5.48 3.31 0.89   0
fi7 233.12 62.88 15.66 8.13 4.71 2.84 0.77   0
fi8 221.00 59.61 14.85 7.70 4.46 2.69 0.73   0

> BF.fits <- M %o% beta; round(BF.fits, 2)
            fj2   fj3   fj4  fj5  fj6  fj7 fj8
[1,] 149.21 40.24 10.03 5.20 3.01 1.82 0.49   0
[2,] 153.35 41.36 10.30 5.35 3.10 1.87 0.50   0
[3,] 160.50 43.29 10.78 5.59 3.24 1.96 0.53   0
[4,] 167.21 45.10 11.23 5.83 3.38 2.04 0.55   0
[5,] 178.84 48.24 12.02 6.23 3.61 2.18 0.59   0
[6,] 186.92 50.42 12.56 6.52 3.77 2.28 0.61   0
[7,] 190.86 51.48 12.82 6.65 3.85 2.33 0.63   0
[8,] 188.82 50.93 12.69 6.58 3.81 2.30 0.62   0
```

Question 14

```
## (a)
future <- row(CL.fits) + col(CL.fits) - 1 > TT
> CL.reserve <- sum(CL.fits[future]); CL.reserve
[1] 152.0312
> BF.reserve <- sum(BF.fits[future]); BF.reserve
[1] 125.9025

## (b)
> CL.retro <- sum(CL.fits[!future]); CL.retro
[1] 2121
> BF.retro <- sum(BF.fits[!future]); BF.retro
[1] 1810.341

## (c)
> sum(Xij)
[1] 2121
```

Indeed, `CL.retro` is equal to `sum(Xij)` but `BF.retro` is not. This is because the Chain Ladder method satisfies the *marginal totals property*, whereas no such condition applies to the Bornhuetter-Ferguson method.