Non-life — Assignment NL4

Niels Keizer[*] and Robert Jan Sopers[†]

October 9, 2016

1 De Vijlder's least squares method

First we read in the data of De Vylder.

```
rm(list=ls(all=TRUE)) ## Discard old garbage
Xij <- scan(n=60)
     0      0      0      0      0  4627
     0      0      0      0  15140 13343
     0      0      0 43465  19018 12476
     0      0 116531 42390  23505 14371
     0 346807 118035 43784  12750 12284
308580 407117 132247 37086  27744     0
358211 426329 157415 68219      0     0
327996 436744 147154     0      0     0
377369 561699      0      0      0     0
333827      0      0      0      0     0
```

Q1

We are asked to fill in the dots to create the covariates `i` and `j`. We use the folllowing statements in `R`.

```
i <- rep(1:10, each=6) ## the row nrs are (1,1,1,1,1,1,2,2,2,2,2,2,...)
j <- rep(1:6,10)       ## the col nrs are (1,2,3,4,5,6,1,2,3,4,5,6,...)
k <- i+j-1             ## the calendar year of the payments
future <- which(k>10)  ## TRUE for obs with calendar year after now
valid <- which(Xij!=0) ## 1 for the non-zero obs, 0 for zero obs
```

Q2

We then run the code from the assignment to find the `alpha` and `beta` estimates.

```
fi <- as.factor(i); fj <- as.factor(j); fk <- as.factor(k)
```

[*]Student number: 10910492
[†]Student number: 0629049

```
xtabs(Xij~i+j)

start <- Xij+0.5
gg <- glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)

cc <- exp(coef(gg)); round(cc, 3)
alpha <- cc[1] * c(1,cc[2:10]); names(alpha)[1] <- "fi1"
beta <- c(1,cc[11:15]); names(beta)[1] <- "fj1"
alpha <- alpha * sum(beta); beta <- beta / sum(beta)
round(alpha); round(beta, 3)
```

Then we check the obtained values against the values of x_i and p_i from De Vylder, Table 3, p. 253.

```
> options(digits=2)
> beta
  fj1   fj2   fj3   fj4   fj5   fj6
0.323 0.434 0.147 0.054 0.025 0.017
> alpha
   fi1     fi2     fi3     fi4     fi5     fi6     fi7     fi8     fi9    fi10
270638  664133  790749  796639  798643  939137 1032577 1009003 1249258 1033618
```

We conclude that the values are almost exactly equal, with exception of `fi10`, which has a value of 1033618. where x_9 is 1033617.

Q3

We execute the code from the assignment and get the following:

```
> start <- rep(1,length(Xij))
> glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)$iter
Error: no valid set of coefficients has been found: please supply starting values
> start <- rep(10000,length(Xij))
> glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)$iter
Error: inner loop 1; cannot correct step size
In addition: Warning message:
step size truncated due to divergence
> start <- rep(100000,length(Xij))
> glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)$iter
[1] 9
> start <- rep(mean(Xij),length(Xij))
> glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)$iter
[1] 10
> start <- rep(mean(Xij[Xij>0]), length(Xij))
> glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)$iter
[1] 7
> start <- fitted.values(glm(Xij~fi+fj,poisson,weights=valid))
> glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)$iter
[1] 4
```

```
> start <- Xij+0.5
> glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)$iter
[1] 5
> start <- Xij; start[Xij==0] <- 0.01
> glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)$iter
[1] 5
> start <- pmax(Xij, 0.01)
> glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)$iter
[1] 5
```

We see that starting with ones as values produces the same error as not supplying start values at all. Starting with values of 10000 gives a divergence in the algorithm. Starting with values of 100000 does not give divergence, but it does take 9 iterations. Starting with the mean of `Xij` gives even more iterations, 10, which is the highest for all glm's here. Initial values of the mean of the non zero values is slightly faster, but 7 is still not as low as it can get. The lowest amount of iterations is obtained by starting with the coefficients of a glm that does allow for starting values 0. The final three glm's all need 5 iterations. The final two even give exactly the same starting values and the one before that is comparable, as it adds 0.5 to everything, thus preventing values equal to 0.

Q4

We generate a glm both without and with an inflation term and compare the results.

```
> gg <- glm(Xij~fi+fj,gaussian(link=log),weights=valid,mustart=start)
> ggg <- glm(Xij~fi+fj+k,gaussian(link=log),weights=valid,mustart=fitted(gg))
> round(exp(coef(gg)),3); round(exp(coef(ggg)),3)
(Intercept)         fi2         fi3         fi4         fi5         fi6         fi7          fi
  87407.917       2.454       2.922       2.944       2.951       3.470       3.815        3.72
        fi9        fi10         fj2         fj3         fj4         fj5         fj6
      4.616       3.819       1.344       0.455       0.168       0.077       0.053
(Intercept)         fi2         fi3         fi4         fi5         fi6         fi7          fi
  87407.896       2.454       2.922       2.944       2.951       3.470       3.815        3.72
        fi9        fi10         fj2         fj3         fj4         fj5         fj6
      4.616       3.819       1.344       0.455       0.168       0.077       0.053           N
> gg$iter; ggg$iter
[1] 5
[1] 1
> (gg$deviance - ggg$deviance)/ggg$deviance
[1] 6.870721098e-14
```

We find that the only difference is that the second glm reports that the coefficient for `k` is `NA`, which means that it did not need the variate. The second glm needed only one iteration to converge, which does not surprise us, seeing that its starting values are equal to the end result. The deviances are also pretty much the same.

Q5

We run the code from the assignment and get the following output in `R`.

```
> xtabs(round(fitted(gg))*future~i+j)[6:10,2:6]
    j
i         2      3      4      5      6
  6       0      0      0      0  16056
  7       0      0      0  25666  17654
  8       0      0  54669  25080  17251
  9       0 183413  67686  31052  21358
  10 448672 151753  56003  25692  17671
```

These results and those from Table 2 of De Vylder are equal.

Q6

The De Vijlder model is a multiplicative model $\alpha_i\beta_j$, which minimizes the least squares distance. Thus solving:

$$\min_{\alpha_i,\beta_j} \sum_{i,j} w_{ij} \left(x_{ij} - \alpha_i\beta_j\right)^2 \tag{1}$$

We take the derivative of the sum with respect to $\alpha_{i'}$ and set it equal to zero.

$$\begin{aligned}
\frac{\partial}{\partial \alpha_{i'}} \sum_{i,j} w_{ij} \left(x_{ij} - \alpha_i\beta_j\right)^2 &= 0 && (2)\\
\sum_{i,j} w_{ij} \frac{\partial}{\partial \alpha_{i'}} \left(x_{ij} - \alpha_i\beta_j\right)^2 &= 0 && (3)\\
\sum_{i,j} w_{ij} \left(-2\beta_j\right) \left(x_{ij} - \alpha_i\beta_j\right) \delta_{ii'} &= 0 && (4)\\
\sum_{j} w_{i'j} \left(-2\beta_j\right) \left(x_{i'j} - \alpha_{i'}\beta_j\right) &= 0 && (5)\\
\alpha_i \sum_j w_{ij}\beta_j^2 &= \sum_j w_{ij}x_{ij}\beta_j && (6)\\
\alpha_i &= \frac{\sum_j w_{ij}x_{ij}\beta_j}{\sum_j w_{ij}\beta_j^2} && (7)
\end{aligned}$$

In the derivation above, we used that $\frac{\partial \alpha_i}{\partial \alpha_{i'}} = \delta_{ii'}$, which is 1 when $i = i'$, 0 otherwise. Afterwards, we replaced i' by i, for easier reading. Because equation 1 is symmetric in α_i and β_j, we can exchange them in equation 7 to obtain:

$$\beta_j = \frac{\sum_i w_{ij}x_{ij}\alpha_i}{\sum_j w_{ij}\alpha_i^2} \tag{8}$$

Using equation 7, we fill the dots and run the following code, including results:

```
> beta <- rep(1, 6)
> repeat
+ { beta.old <- beta
+ alpha <- tapply(valid*Xij*beta[j],i,sum)/tapply(valid*beta[j]^2,i,sum)
+ beta <- tapply(valid*Xij*alpha[i],j,sum)/tapply(valid*alpha[i]^2,j,sum)
+ if (sum(abs((beta.old-beta)/beta)) < 1e-7) break ## out of the loop
+ # cat(beta,"\n") ## to monitor the iteration process
+ }
> round(xtabs(alpha[i]*beta[j]*future~i+j)[6:10,2:6])
    j
i         2      3      4      5      6
  6       0      0      0      0  16056
  7       0      0      0  25666  17654
  8       0      0  54669  25080  17251
  9       0 183413  67686  31052  21358
 10  448672 151753  56003  25692  17671
```

These results are exactly the same as the one obtained from Q6.

2 Using Hoerl growth curves

Q7

To find for which x the Hoerl-curve has a minimum, we differentiate $\beta(x)$ w.r.t. x and set equal to zero.

$$\begin{aligned}
\frac{d}{dx}\exp(\gamma x+\delta\log(x)) &= 0 && (9)\\
(\gamma+\frac{\delta}{x})\exp(\gamma x+\delta\log(x)) &= 0 && (10)\\
(\gamma+\frac{\delta}{x}) &= 0 && (11)\\
x &= -\frac{\delta}{\gamma} && (12)
\end{aligned}$$

Because the logarithm only operates only strictly positive values, this means that there can only be a maximum when $\delta > 0$ and $\gamma < 0$. This result could also give a minimum, so we check if the second derivative is negative for this value of x. We get

$$\begin{aligned}
\frac{d^2}{dx^2}\exp(\gamma x+\delta\log(x)) &= \frac{d}{dx}(\gamma+\frac{\delta}{x})\exp(\gamma x+\delta\log(x)) && (13)\\
&= ((\gamma+\frac{\delta}{x})^2-\frac{\delta}{x^2})\exp(\gamma x+\delta\log(x)) && (14)\\
&= \frac{\delta}{(-\frac{\delta}{\gamma})^2}\exp(\gamma(-\frac{\delta}{\gamma})+\delta\log(-\frac{\delta}{\gamma})) && (15)\\
&= \frac{\gamma}{\delta}\exp(\delta(\log(-\frac{\delta}{\gamma})-1)) && (16)
\end{aligned}$$

which is indeed a negative under the restrictions for δ and γ.

$$\begin{aligned}
\lim_{x\to\infty}\frac{\beta(x+1)}{\beta(x)} &= \lim_{x\to\infty}\frac{\exp(\gamma(x+1)+\delta\log(x+1))}{\exp(\gamma x+\delta\log(x))} && (17)\\
&= \lim_{x\to\infty}\exp(\gamma+\delta(\log(x+1)-\log(x))) && (18)\\
&= \lim_{x\to\infty}\exp(\gamma+\delta\log\left(\frac{x+1}{x}\right)) && (19)\\
&= \exp(\gamma+\delta\log(1)) && (20)\\
&= \exp(\gamma)=d && (21)
\end{aligned}$$

Q8

We fill the dots in the code and then check that `beta[1]==1` holds.

```
> rm(list=ls(all=TRUE)) ## Discard old garbage
> TT <- 10; x.top <- 2; d <- .5
> gamma <- log(d); delta <- -x.top*gamma
> beta <- exp(gamma*(1:TT)+delta*log(1:TT))/exp(gamma)
> beta[1]==1
[1] TRUE
```

Q9

The `Xij` values are all even numbers because the `rpois` function only returns integers, after which they are multiplied by `phi`, which is equal to two. An number is even when it is equal to an integer times two, therefore the `Xij` values are all even.

Q10

After running the code from the assignment several times, we find that the Hoerl model gets both accepted and rejected. We run this code:

```
Xij <- phi * rpois(length(mu.ij), mu.ij/phi)

xtabs(round(mu.ij)~i+j)
round(xtabs(Xij~i+j))

CL <- glm(Xij~fi+fj-1, quasipoisson)
exp(coef(CL))

Hoerl <- glm(Xij~fi+I(j-1)+log(j)-1, quasipoisson)

round(coef(CL),3); round(coef(Hoerl),3)
beta.CL <- exp(c(0,coef(CL)[(TT+1):(2*TT-1)]))
beta.Hoerl <- exp(coef(Hoerl)[TT+1]*(0:(TT-1))) * (1:TT)^coef(Hoerl)[TT+2]
round(rbind(beta.CL, beta.Hoerl), 4)
plot(beta.CL); points
```

```
scale <- CL$deviance/CL$df.residual
Delta.Dev.Sc <- (Hoerl$deviance - CL$deviance)/scale
Delta.df <- Hoerl$df.residual - CL$df.residual
reject <- Delta.Dev.Sc > qchisq(0.95, Delta.df)
cat("The Hoerl model", ifelse(reject, "is", "is not"), "rejected",
"since the scaled deviance gained by CL is", round(Delta.Dev.Sc,1),
"\nwith", Delta.df, "extra parameters.\n")
```

And we get the following output:

```
> cat("The Hoerl model", ifelse(reject, "is", "is not"), "rejected",
+     "since the scaled deviance gained by CL is", round(Delta.Dev.Sc,1),
+     "\nwith", Delta.df, "extra parameters.\n")
The Hoerl model is not rejected since the scaled deviance gained by CL is 8.3
with 7 extra parameters.

> cat("The Hoerl model", ifelse(reject, "is", "is not"), "rejected",
+     "since the scaled deviance gained by CL is", round(Delta.Dev.Sc,1),
+     "\nwith", Delta.df, "extra parameters.\n")
The Hoerl model is rejected since the scaled deviance gained by CL is 16.3
with 7 extra parameters.

> cat("The Hoerl model", ifelse(reject, "is", "is not"), "rejected",
+     "since the scaled deviance gained by CL is", round(Delta.Dev.Sc,1),
+     "\nwith", Delta.df, "extra parameters.\n")
The Hoerl model is not rejected since the scaled deviance gained by CL is 9.4
with 7 extra parameters.

> cat("The Hoerl model", ifelse(reject, "is", "is not"), "rejected",
+     "since the scaled deviance gained by CL is", round(Delta.Dev.Sc,1),
+     "\nwith", Delta.df, "extra parameters.\n")
The Hoerl model is not rejected since the scaled deviance gained by CL is 5.8
with 7 extra parameters.
```

Out of a sample of 4 iterations, the Hoerl model is rejected once.

Q11

The CL-method is the fullest model available, therefore it is used to determine the scale factor. This is equal to the deviance divided by the residuals of the full CL-model. Because the Hoerl model uses less parameters, its deviance is likely to be larger, so the gain in deviance is that of the Hoerl model minus the CL-model. This is the divided by the scale factor to obtain the gain in scaled deviance. The difference in residuals is determined by subtracting the residuals of the models. The Hoerl model is then rejected when the gain in scaled deviance is larger than the 95th percentile of a chi square distribution with the gain in residuals as degrees of freedom. This is exactly what the `R` code does.

Q12

We run similar code to the previous exercises:

```
> rm(list=ls(all=TRUE)) ## Discard old garbage
> Xij <- c(232,106,35,16,2, 258,115,56,27, 221,82,4, 359,71, 349)
>   i <- c( 1, 1, 1, 1,1, 2, 2, 2, 2, 3, 3,3, 4, 4, 5)
>   j <- c( 1, 2, 3, 4,5, 1, 2, 3, 4, 1, 2,3, 1, 2, 1)
> fi <- as.factor(i); fj <- as.factor(j)
> xtabs(Xij~i+j)
    j
i      1   2   3   4   5
  1 232 106  35  16   2
  2 258 115  56  27   0
  3 221  82   4   0   0
  4 359  71   0   0   0
  5 349   0   0   0   0
> CL <- glm(Xij~fi+fj-1, quasipoisson)
> Hoerl <- glm(Xij~fi+I(j-1)+log(j)-1, quasipoisson)
> # Then we do an anlysis of deviance.
> scale <- CL$deviance/CL$df.residual
> Delta.Dev.Sc <- (Hoerl$deviance - CL$deviance)/scale
> Delta.df <- Hoerl$df.residual - CL$df.residual
> reject <- Delta.Dev.Sc > qchisq(0.95, Delta.df)
> cat("The Hoerl model", ifelse(reject, "is", "is not"), "rejected",
+     "since the scaled deviance gained by CL is", round(Delta.Dev.Sc,1),
+     "\nwith", Delta.df, "extra parameters.\n")
The Hoerl model is not rejected since the scaled deviance gained by CL is 0.7
with 2 extra parameters.
```

The output is that the Hoerl model is not rejected, so it is a good fit to the β's, compared to CL.

We also want to know if the portfolio growth can be described by a Hoerl curve. We then also try to fit a Hoerl curve to the rows.

```
> # Now try with a Hoerlcurve for the portfolio growth.
> Hoerl <- glm(Xij~I(i-1)+log(i)+fj-1, quasipoisson)
> # Then we do an anlysis of deviance.
> scale <- CL$deviance/CL$df.residual
> Delta.Dev.Sc <- (Hoerl$deviance - CL$deviance)/scale
> Delta.df <- Hoerl$df.residual - CL$df.residual
> reject <- Delta.Dev.Sc > qchisq(0.95, Delta.df)
> cat("The Hoerl model", ifelse(reject, "is", "is not"), "rejected",
+     "since the scaled deviance gained by CL is", round(Delta.Dev.Sc,1),
+     "\nwith", Delta.df, "extra parameters.\n")
The Hoerl model is not rejected since the scaled deviance gained by CL is 2.8
with 2 extra parameters.
```

We conclude that the CL model is not an improvement over the Hoerl model for the portfolio growth.

3 Separation methods; calendar year

Q13

First we run the code from the assignment to generate the chain ladder and threeway models.

```
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 <- trunc(sqrt(2*length(Xij)))
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)

CL <- glm(Xij~fi+fj, poisson)
Threeway <- glm(Xij~fi+fj+fk, poisson)
anova(CL, Threeway)
round(qchisq(0.95, c(21,15,6)),1)
```

We then extract `alpha` and `beta` from the vector of coefficients of `CL` and print the square of the fitted values.

```
> cc <- exp(coef(CL))
> alpha <- cc[1] * c(1,cc[2:TT]); names(alpha)[1] <- "fi1"
> beta <- c(1,cc[(TT+1):(2*TT-1)]); names(beta)[1] <- "fj1"
> alpha <- alpha * sum(beta); beta <- beta / sum(beta)
> round(alpha); round(beta, 3)
fi1 fi2 fi3 fi4 fi5 fi6 fi7 fi8
210 218 265 283 276 382 328 311
  fj1   fj2   fj3   fj4   fj5   fj6   fj7   fj8
0.711 0.192 0.048 0.025 0.014 0.009 0.002 0.000
> round(alpha%o%beta,3)
        fj1    fj2    fj3   fj4   fj5   fj6   fj7 fj8
fi1 149.206 40.245 10.025 5.201 3.013 1.819 0.491   0
fi2 154.890 41.778 10.407 5.399 3.128 1.888 0.509   0
fi3 188.013 50.712 12.633 6.553 3.797 2.292 0.618   0
fi4 201.154 54.257 13.516 7.012 4.062 2.453 0.661   0
fi5 196.096 52.893 13.176 6.835 3.960 2.391 0.645   0
fi6 271.520 73.236 18.244 9.464 5.483 3.310 0.893   0
fi7 233.121 62.879 15.664 8.126 4.708 2.842 0.767   0
fi8 221.000 59.610 14.849 7.703 4.463 2.695 0.727   0
```

The last column is equal to zero due to the marginal totals property, which states that the row and column sums of the observed and the fitted values must be equal. The observed values in column 8 are zero, therefore the fitted values are 0 as well.

Q14

We generate the Arithmetic Separation method with the following R-code:

```
> AS <- glm(Xij~fj+fk, poisson)
> exp(coef(AS)["(Intercept)"])
(Intercept)
156
```

For the Arithmetic Separation method, the marginal totals property also holds, in this case for the columns sums and the diagonal sums with equal `k`. Because the top left observation is the only cell in the diagonal with `k` value 1, the fitted value (which for the top left observation is the exponent of the intercept) must be equal to the observed value.

Q15

We construct the both the fitted and extrapolated values and then generate the fitted values for the full IBNR-square. We use the code from the assignment as base and then fill the dots.

```
> gamma.extrapolated <- exp(ab[1]+(1:15)*ab[2])
> gammas <- gamma.extrapolated; gammas[1:8] <- gamma.AS
> jjj <- rep(1:8,8); kkk <- jjj + rep(1:8,each=8) - 1
> mm <- beta.AS[jjj]*gammas[kkk]
> round(matrix(mm,8,byrow=TRUE),3)
        [,1]   [,2]   [,3]  [,4]  [,5]  [,6]  [,7] [,8]
[1,] 156.000 39.137 10.237 5.563 3.138 1.972 0.547    0
[2,] 151.863 44.211 12.159 5.702 3.569 2.205 0.453    0
[3,] 171.552 52.513 12.464 6.485 3.992 1.823 0.570    0
[4,] 203.766 53.828 14.177 7.253 3.301 2.294 0.613    0
[5,] 208.869 61.225 15.854 5.997 4.154 2.470 0.660    0
[6,] 237.572 68.469 13.110 7.547 4.471 2.659 0.711    0
[7,] 265.681 56.618 16.497 8.124 4.814 2.863 0.765    0
[8,] 219.697 71.245 17.759 8.746 5.182 3.082 0.824    0
```

Q16

We use the `AIC()` function in `R` to calculate the AIC.

```
> cbind("AS"=AIC(AS), "CL"=AIC(CL), "Threeway"=AIC(Threeway))
           AS       CL Threeway
[1,] 233.5406 227.4894 226.5827
```

The AIC for the Threeway model is the lowest, so with regards to the AIC, the Threeway model is the best fit for this data.

4 Forecasting IBNR-claims

Q17

We start by running the code from the assignment. As seed, we use the birthday of Niels Keizer.

```
> rm(list=ls(all=TRUE)) ## Discard old garbage
> set.seed(841109) ## replace by your birthday in format yymmdd
> top <- 1+1.5*runif(1); decay <- .5 + runif(1)/5
> gamma <- log(decay); delta <- -top*gamma
> beta <- exp(gamma*(0:(10-1)) + delta*log(1:10))
> alpha <- 1.03^(1:10) * (.80+runif(10)/5)
> alpha <- 100 * alpha / alpha[1] / beta[1]
> i <- rep(1:10,10:1); j <- sequence(10:1)
> fi <- as.factor(i); fj <- as.factor(j)
> phi <- 1.1+runif(1)/3
> Xij <- round(phi * rpois(55, alpha[i] * beta[j]/phi))
> rm(phi,alpha,beta,gamma,delta,top,decay)
> Xij <- pmax(Xij,1)
> xtabs(Xij~i+j)
    j
i      1   2   3   4   5   6   7   8   9  10
  1   95 114 101  67  67  36  31  13  11  10
  2   88 116 115  78  59  34  18  19   5   0
  3   93 130  94  74  53  34  17  24   0   0
  4   94 120 107  98  52  40  26   0   0   0
  5   69 102  90  75  52  36   0   0   0   0
  6   91 138 106  77  66   0   0   0   0   0
  7   97 118 108  80   0   0   0   0   0   0
  8  109 117 110   0   0   0   0   0   0   0
  9  102 130   0   0   0   0   0   0   0   0
 10  133   0   0   0   0   0   0   0   0   0
> anova(glm(Xij ~ i+j+log(i)+log(j)+fi+fj, quasipoisson)) ## for Q16
Analysis of Deviance Table

Model: quasipoisson, link: log

Response: Xij

Terms added sequentially (first to last)


       Df Deviance Resid. Df Resid. Dev
NULL                      54    1226.48
i       1   275.65        53     950.83
j       1   671.86        52     278.97
log(i)  1     0.14        51     278.83
log(j)  1   224.53        50      54.30
fi      7    17.29        43      37.01
```

```
fj       7     6.22         36      30.79
> rbind(1:10,round(qchisq(.95,1:10),1))
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]  1.0    2  3.0  4.0  5.0  6.0  7.0  8.0  9.0  10.0
[2,]  3.8    6  7.8  9.5 11.1 12.6 14.1 15.5 16.9  18.3
```

From the `anova()` call above, we can already see which covariates give the largest reductions in deviance. This gives us an idea of what covariates to use in the 'best' model. For this, we need to do an analysis of deviance. The estimate $\hat{\phi}$ for the dispersion parameter ϕ is chosen as the residual deviance divided by the residual degrees of freedom of the 'fullest' model. We calculate this in `R`.

```
> # Fullest model
> CL <- glm(Xij ~ fi+fj, quasipoisson)
> phi <- CL$deviance/CL$df.residual
> phi
[1] 0.8552915
```

Thus $\hat{\phi} = 0,8553$. We then examine whether we can improve the CL model by replacing either `fi` or `fj` with a Hoerl pattern (or maybe both). From the anova call, we strongly suspect that introducing a Hoerl pattern in the columns might give an improvement. To check this, we perform an analysis of deviance.

```
> Hoerl_j <- glm(Xij ~ fi+I(j-1)+log(j), quasipoisson)
> anova(Hoerl_j,CL, test = "Chisq")
Analysis of Deviance Table

Model 1: Xij ~ fi + I(j - 1) + log(j)
Model 2: Xij ~ fi + fj
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        43     37.009
2        36     30.790  7   6.2182   0.4041
> delta.dev.sc <- (Hoerl_j$deviance - CL$deviance)/phi
> delta.df <- Hoerl_j$df.residual - CL$df.residual
> reject <- delta.dev.sc > qchisq(0.95,delta.df)
> reject;delta.dev.sc;qchisq(0.95,delta.df)
[1] FALSE
[1] 7.270253
[1] 14.06714
```

The analysis above shows that adding the factors of `j` as covariates over the Hoerl model, should give a scaled deviance improvement larger than 14.1 to be significant. This is not the case, thus this Hoerl model is better than the chain ladder model.

Then we check if we can describe the portfolio growth with a Hoerl curve over the chain ladder method.

```
> Hoerl_i <- glm(Xij ~ I(i-1)+log(i)+fj, quasipoisson)
> anova(Hoerl_i,CL, test = "Chisq")
Analysis of Deviance Table
```

```
Model 1: Xij ~ I(i - 1) + log(i) + fj
Model 2: Xij ~ fi + fj
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        43     48.687
2        36     30.790  7   17.897 0.004006 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> delta.dev.sc <- (Hoerl_i$deviance - CL$deviance)/phi
> delta.df <- Hoerl_i$df.residual - CL$df.residual
> reject <- delta.dev.sc > qchisq(0.95,delta.df)
> reject;delta.dev.sc;qchisq(0.95,delta.df)
[1] TRUE
[1] 20.92474
[1] 14.06714
```

This analysis shows that the `fi` covariates can not be replaced by a Hoerl curve. But maybe we can replace the covariates by a variate.

```
> Variate_i <- glm(Xij ~ i+fj, quasipoisson)
> anova(Variate_i,CL, test = "Chisq")
Analysis of Deviance Table

Model 1: Xij ~ i + fj
Model 2: Xij ~ fi + fj
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        44      52.95
2        36      30.79  8   22.159 0.001132 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> delta.dev.sc <- (Variate_i$deviance - CL$deviance)/phi
> delta.df <- Variate_i$df.residual - CL$df.residual
> reject <- delta.dev.sc > qchisq(0.95,delta.df)
> reject;delta.dev.sc;qchisq(0.95,delta.df)
[1] TRUE
[1] 25.9086
[1] 15.50731
```

This analysis above shows that using the row factors over the variate is a significant improvement, thus using the row variate over its factors is rejected.

Finally, we have not checked if using `j` as a variate over `fj` is an improvement. Considering the first `anova()` call, we do not expect an improvement at all. We check to be absolutely sure...

```
> Variate_j <- glm(Xij ~ fi+j, quasipoisson)
> anova(Variate_j,CL, test = "Chisq")
Analysis of Deviance Table

Model 1: Xij ~ fi + j
Model 2: Xij ~ fi + fj
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1        44     266.19
2        36      30.79  8    235.4 < 2.2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
> delta.dev.sc <- (Variate_j$deviance - CL$deviance)/phi
> delta.df <- Variate_j$df.residual - CL$df.residual
> reject <- delta.dev.sc > qchisq(0.95,delta.df)
> reject;delta.dev.sc;qchisq(0.95,delta.df)
[1] TRUE
[1] 275.2217
[1] 15.50731
```

Nope, definitely not an improvement. This means that our 'best' model uses factors for the years of origin and a Hoerl pattern for the development years.

Q18

Generalized Linear Models have three characteristics. We will describe each of these characteristics for our 'best' model.

1. The stochastic component. These are the `Xij` variables, which have a quasi-Poisson distribution.
2. The systematic component. These are the linear predictors. A sum of products of the regressors and the coefficients of those regressors, linear in the coefficients. In our 'best' model, the regressors are the factors of the rows (or row dummies), the column value (minus 1) and the logarithm of the column value. The coefficients are determined by the `glm()` call in `R`.
3. The link function. Because we consider this as a multiplicative model, we use the logarithm. This function also happens to be the canonical link function of a model with a quasi-Poisson stochastic component.

Q19

From the question it is unclear to me if we should estimate using our 'best' model, or the chain ladder model. The final sentence of the question seems to imply the chain ladder method, since it would weird to say that the difference calculated by a Hoerl method estimates the reserve of a CL method. To be sure, I will do both.

First is the `R` code for the Hoerl method:

```
> alpha.h <- exp(coef(Hoerl_j))[1]*c(1,exp(coef(Hoerl_j))[2:10])
> gamma <- coef(Hoerl_j)[11]; delta <- coef(Hoerl_j)[12]
> beta.h <- exp(gamma*(0:9) + delta*log(1:10))
> mu.past.h <- alpha.h[i]*beta.h[j]
> i.all <- rep(1:10,each=10)
> j.all <- rep(1:10,10)
> mu.all.h <- alpha.h[i.all]*beta.h[j.all]
> reserve.h <- sum(mu.all.h) - sum(mu.past.h)
>
> xtabs(round(mu.past.h,2)~i+j)
    j
i         1      2      3      4      5      6      7      8      9     10
  1   92.17 116.73 103.82  79.90  56.80  38.41  25.09  15.97   9.98   6.14
  2   91.00 115.24 102.50  78.88  56.08  37.92  24.77  15.77   9.85   0.00
  3   90.45 114.55 101.88  78.40  55.74  37.69  24.62  15.68   0.00   0.00
  4   96.50 122.21 108.69  83.65  59.47  40.21  26.26   0.00   0.00   0.00
  5   80.11 101.46  90.24  69.44  49.37  33.38   0.00   0.00   0.00   0.00
  6   98.03 124.15 110.42  84.98  60.41   0.00   0.00   0.00   0.00   0.00
  7   94.61 119.82 106.56  82.01   0.00   0.00   0.00   0.00   0.00   0.00
  8   99.03 125.42 111.55   0.00   0.00   0.00   0.00   0.00   0.00   0.00
  9  102.36 129.64   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
  10 133.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
> xtabs(round(mu.all.h,2)~i.all+j.all)
j.all
i.all      1      2      3      4      5      6      7      8      9     10
   1   92.17 116.73 103.82  79.90  56.80  38.41  25.09  15.97   9.98   6.14
   2   91.00 115.24 102.50  78.88  56.08  37.92  24.77  15.77   9.85   6.06
   3   90.45 114.55 101.88  78.40  55.74  37.69  24.62  15.68   9.79   6.02
   4   96.50 122.21 108.69  83.65  59.47  40.21  26.26  16.72  10.45   6.43
   5   80.11 101.46  90.24  69.44  49.37  33.38  21.80  13.88   8.67   5.33
   6   98.03 124.15 110.42  84.98  60.41  40.85  26.68  16.99  10.61   6.53
   7   94.61 119.82 106.56  82.01  58.30  39.42  25.75  16.40  10.24   6.30
   8   99.03 125.42 111.55  85.84  61.03  41.27  26.95  17.16  10.72   6.59
   9  102.36 129.64 115.30  88.73  63.08  42.65  27.86  17.74  11.08   6.82
   10 133.00 168.43 149.80 115.29  81.96  55.42  36.20  23.05  14.40   8.86
> reserve.h
[1] 1639.473
```

Second is the R code for the chain ladder method:

```
> alpha.cl <- exp(coef(CL))[1]*c(1,exp(coef(CL))[2:10])
> beta.cl <- c(1,exp(coef(CL))[11:19])
> alpha.cl <- alpha.cl*sum(beta.cl)
> beta.cl <- beta.cl/sum(beta.cl)
> mu.past.cl <- alpha.cl[i]*beta.cl[j]
> mu.all.cl <- alpha.cl[i.all]*beta.cl[j.all]
> reserve.cl <- sum(mu.all.cl) - sum(mu.past.cl)
>
> xtabs(round(mu.past.cl,2)~i+j)
    j
i         1      2      3      4      5      6      7      8      9     10
  1   91.15 118.02 102.79  78.35  58.41  36.66  22.80  18.80   8.02  10.00
  2   90.64 117.36 102.22  77.91  58.09  36.46  22.67  18.69   7.98   0.00
  3   89.77 116.23 101.24  77.16  57.53  36.11  22.45  18.51   0.00   0.00
  4   96.32 124.71 108.62  82.79  61.73  38.74  24.09   0.00   0.00   0.00
  5   79.62 103.09  89.79  68.44  51.03  32.03   0.00   0.00   0.00   0.00
  6   97.10 125.72 109.50  83.46  62.22   0.00   0.00   0.00   0.00   0.00
  7   94.12 121.86 106.13  80.89   0.00   0.00   0.00   0.00   0.00   0.00
  8   98.18 127.11 110.71   0.00   0.00   0.00   0.00   0.00   0.00   0.00
  9  101.10 130.90   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
  10 133.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
> xtabs(round(mu.all.cl,2)~i.all+j.all)
    j.all
i.all      1      2      3      4      5      6      7      8      9     10
    1  91.15 118.02 102.79  78.35  58.41  36.66  22.80  18.80   8.02  10.00
    2  90.64 117.36 102.22  77.91  58.09  36.46  22.67  18.69   7.98   9.94
    3  89.77 116.23 101.24  77.16  57.53  36.11  22.45  18.51   7.90   9.85
    4  96.32 124.71 108.62  82.79  61.73  38.74  24.09  19.86   8.48  10.57
    5  79.62 103.09  89.79  68.44  51.03  32.03  19.91  16.42   7.01   8.74
    6  97.10 125.72 109.50  83.46  62.22  39.06  24.28  20.02   8.55  10.65
    7  94.12 121.86 106.13  80.89  60.31  37.86  23.54  19.41   8.28  10.33
    8  98.18 127.11 110.71  84.38  62.91  39.49  24.55  20.24   8.64  10.77
    9 101.10 130.90 114.01  86.90  64.79  40.67  25.28  20.85   8.90  11.09
   10 133.00 172.20 149.98 114.32  85.23  53.50  33.26  27.43  11.71  14.59
> reserve.cl
[1] 1666.644
```

The difference between the estimates for both past and future and the estimates for the past are the estimates for the future. The estimates for the future are the expected claims for past years of origin and future development years, which are precisely the IBNR claims. The sum then gives the total expected IBNR claims, for which the insurer should hold a reserve.

Although it is not a part of this question, we see that in this case, the chain ladder method is more prudent because the estimated reserve is higher, 1666.64 vs. 1639.47. A difference of 27.17.

5 Zero adjusted geometric delays

Q20

Given that $\Pr[T = t] \propto (1-p)^t$, for $t > 0$, $t \in \mathbb{Z}$, we derive

$$\begin{aligned}
\Pr[T = t+1]/\Pr[T = t] &= \frac{(1-p)^{t+1}}{(1-p)^t} && (22)\\
&= 1-p && (23)\\
&= \Pr[T = 2]/\Pr[T = 1] && (24)
\end{aligned}$$

which is independent on t. Therefore it is equal for all $t > 0$, of which $t = 1$ is a special case.

We now consider an exponential(λ) random variable U, which determines when a claim is filed. $T = 0$ if $U \leq t_0$, but $T = \lceil U - t_0 \rceil$. Also $\Pr[T = 0] = P_0$. Then

$$\Pr[T = 0] = \Pr[U \leq t_0] = \int_0^{t_0} \lambda e^{-\lambda u} du = 1 - e^{-\lambda t_0} = p_0 \qquad (25)$$

gives a relation between λ, t_0 and p_0. Also, since $\Pr[T=t] = \Pr[\lceil U - t_0 \rceil = t] = \Pr[t_0 + t - 1 < U \leq t_0 + t]$ for $U > t_0$, we derive:

$$\begin{aligned} \Pr[T=t] &= \Pr[t_0 + t - 1 < U \leq t_0 + t] = \int_{t_0+t-1}^{t_0+t} \lambda e^{-\lambda u} du = e^{-\lambda(t_0+t-1)} - e^{-\lambda(t_0+t)} && (26) \\ &= e^{-\lambda(t_0+t)}(e^{\lambda} - 1) && (27) \end{aligned}$$

Then

$$\Pr[T = t+1] / \Pr[T=t] = \frac{e^{-\lambda(t_0+t+1)}(e^{\lambda}-1)}{e^{-\lambda(t_0+t)}(e^{\lambda}-1)} = e^{-\lambda} = 1 - p \qquad (28)$$

This means that $\lambda = -\log(1-p)$ and from equation 25 follows that $t_0 = \frac{\log(1-p_0)}{\log(1-p)}$ via the following:

$$\begin{aligned} 1 - e^{-\lambda t_0} &= p_0 && (29) \\ e^{-\lambda t_0} &= 1 - p_0 && (30) \\ -\lambda t_0 &= \log(1-p_0) && (31) \\ \log(1-p) t_0 &= \log(1-p_0) && (32) \\ t_0 &= \frac{\log(1-p_0)}{\log(1-p)} && (33) \end{aligned}$$

Since $\log(1-p)$ is monotonically decreasing and negative for $0 \leq p < 1$, this implies that $t_0 > 1$ if and only if $p > p_0$.

We solve the following equation:

$$\begin{aligned} \Pr[T=0] &= \Pr[T=1] && (34) \\ 1 - e^{-\lambda t_0} &= e^{-\lambda(t_0+1)}(e^{\lambda} - 1) && (35) \\ 1 &= e^{-\lambda t_0} e^{-\lambda}(e^{\lambda} - 1) + e^{-\lambda t_0} && (36) \\ e^{\lambda t_0} &= e^{-\lambda}(e^{\lambda} - 1) + 1 && (37) \\ e^{\lambda t_0} &= 2 - e^{-\lambda} && (38) \\ t_0 &= \frac{\log(2 - e^{-\lambda})}{\lambda} = t_0^* && (39) \end{aligned}$$

For $t = 0$, $\Pr[T=t]$ is a monotonically increasing function in t_0. For $t = 1$, $\Pr[T = t]$ is a monotonically decreasing function in t_0. When $t_0 = t_0^*$, the two are equal. When $t_0 < t_0^*$, $\Pr[T=0] < \Pr[T=1]$, thus $\Pr[T=1]$ is maximal and when $t_0 > t_0^*$, $\Pr[T=0] > \Pr[T=1]$, thus $\Pr[T=0]$ is maximal.

Q21

We consider a CL-model with Poisson claim numbers with expected value $\alpha_i \beta_j$. The total number of total claims in a year of origin is Poisson(α_i) distributed. A claim on a contract in year i is filed after $j-1$ years. This is described by β_j, which follows a zero adjusted geometric delay. For the β_j's, we derive the following:

$$\begin{aligned}
\beta_1 &= \Pr[T=0] = p_0 && (40)\\
\beta_j &= \Pr[T=j-1] = e^{-\lambda(t_0+j-1)}(e^{\lambda}-1) && (41)\\
&= e^{--\log(1-p)(\frac{\log(1-p_0)}{\log(1-p)}+j-1)}(e^{-\log(1-p)}-1) && (42)\\
&= e^{\log(1-p_0)+(j-1)\log(1-p)}(\frac{1}{1-p}-1) && (43)\\
&= (1-p_0)(1-p)^{(j-1)}\frac{p}{1-p} && (44)\\
&= p(1-p)^{(j-2)}(1-p_0) && (45)
\end{aligned}$$

The we find expressions for p and p_0 in terms of β_1 and β_2. $p_0 = \beta_1$, as derived above in equation 40. To find p, we use equation 45 for $j = 2$.

$$\begin{aligned}
\beta_2 &= p(1-\beta_1) && (46)\\
p &= \frac{\beta_2}{1-\beta_1} && (47)
\end{aligned}$$

A GLM specification for this would be the following. We know the claims are Poisson(α_i) distributed, so we use a GLM with Poisson errors. From the exercises above we estimate the X_{ij}'s by the following:

$$X_{ij} \approx \alpha_i \beta_j \tag{48}$$

Because this is multiplicative, we use a logarithmic link to estimate the linear coefficients. This means that

$$\mathbb{E}[\log(X_{ij})] = \log(\alpha_i) + \log(\beta_j) \tag{49}$$

For the regressors of the α_i, we choose the row dummies. Because β_1 does not follow the geometric delay, we use a dummy for column 1. Further, if we take the logarithm of β_j for $j > 1$, we get

$$\log(\beta_j) = \log(p) + (j-2)\log(1-p) + \log(1-p_0), \tag{50}$$

which is linear in $j-2$. So as the second regressor for β_j, we take the numeric values of $j-2$. This gives us the following model description in R:

```
> zero <- glm(Xij~fi+I(j==1)+I(j-2), poisson)
> Hoerl <- glm(Xij~fi+I(j-1)+log(j), poisson)
> xtabs(round(zero$fitted.values,2)~i+j)
    j
i         1      2      3      4      5      6      7      8      9     10
  1   90.32 127.67  94.13  69.40  51.17  37.72  27.81  20.51  15.12  11.15
  2   90.01 127.23  93.80  69.16  50.99  37.59  27.72  20.43  15.07   0.00
  3   90.37 127.74  94.18  69.44  51.19  37.74  27.83  20.52   0.00   0.00
  4   97.35 137.61 101.46  74.80  55.15  40.66  29.98   0.00   0.00   0.00
  5   81.41 115.08  84.84  62.55  46.12  34.00   0.00   0.00   0.00   0.00
  6   99.78 141.04 103.99  76.67  56.52   0.00   0.00   0.00   0.00   0.00
  7   95.41 134.86  99.43  73.31   0.00   0.00   0.00   0.00   0.00   0.00
  8    97.23 137.44 101.33   0.00   0.00   0.00   0.00   0.00   0.00   0.00
  9    96.12 135.88   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
  10 133.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
>
> AIC(zero);AIC(Hoerl)
[1] 419.5824
[1] 388.3281
```

According to the AIC, the Hoerl model is a better fit.

From the coefficients of the `zero` model, it is possible to estimate α_i, p_0 and p. However, we are not asked to calculate this, so we dont.