Non-life — Assignment NL3

Niels Keizer* and Robert Jan Sopers†

October 2, 2016

1 GLMs and the Lee-Carter mortality model

Q1

Assuming $\beta_\Sigma := \sum \beta_x \neq 0$ and the transformation

$$\alpha'_x \leftarrow \alpha_x + \beta_x \bar{\kappa}; \quad \kappa''_t \leftarrow \kappa_t \beta_\Sigma; \quad \beta'_x \leftarrow \frac{\beta_x}{\beta_\Sigma}; \quad \kappa'_t \leftarrow \kappa''_t - \overline{\kappa''} \quad \forall x, t \tag{1}$$

a straightforward substitution of this transformation shows that

$$\begin{aligned} \mu'_{xt} &= \exp(\alpha'_x + \beta'_x \kappa'_t) \\ &= \exp\left(\alpha_x + \beta_x \bar{\kappa} + \frac{\beta_x}{\beta_\Sigma}(\kappa''_t - \overline{\kappa''})\right) \\ &= \exp\left(\alpha_x + \beta_x \bar{\kappa} + \frac{\beta_x}{\beta_\Sigma}(\kappa_t \beta_\Sigma - \overline{\kappa_t}\beta_\Sigma)\right) \\ &= \exp\left(\alpha_x + \beta_x \bar{\kappa} + \beta_x \kappa_t - \beta_x \bar{\kappa}\right) = \exp(\alpha_x + \beta_x \kappa_t) = \mu_{xt} \end{aligned}$$

and together with $\bar{\kappa} = \frac{1}{T}\sum_{t=1}^{T} \kappa_t$ we have

$$\sum_x \beta'_x = \sum_x \frac{\beta_x}{\beta_\Sigma} = \frac{1}{\beta_\Sigma} \sum_x \beta_x = \frac{1}{\beta_\Sigma} \beta_\Sigma = 1 \tag{2}$$

and

$$\sum_{t=1}^{T} \kappa'_t = \sum_{t=1}^{T} (\kappa''_t - \overline{\kappa''_t}) = \sum_{t=1}^{T} \kappa_t \beta_\Sigma - \sum_{t=1}^{T} \beta_\Sigma \bar{\kappa} = T\bar{\kappa}\beta_\Sigma - T\beta_\Sigma \bar{\kappa} = 0 \tag{3}$$

which are the desired properties for β'_x and κ'_t.

Q2

From (3) from the assignment we have that

$$\log(\mu_{xt}) = \alpha_x + \beta_x \kappa_t \Rightarrow \alpha_x = \log(\mu_{xt}) - \beta_x \kappa_t \tag{4}$$

assuming now the property $\sum_t \kappa_t = 0$ we see that summing over t gives

$$\sum_{t=1}^{T} \alpha_x = T\alpha_x = \sum_{t=1}^{T} \log \mu_{xt} - \beta_x \sum_{t=1}^{T} \kappa_t = \sum_{t=1}^{T} \log \mu_{xt} \tag{5}$$

*Student number: 10910492
†Student number: 0629049

which implies

$$\alpha_x = \frac{1}{T}\sum_{t=1}^{T} \log \mu_{xt} \quad \forall x \tag{6}$$

Defining the right hand side of this equation as $\overline{\log \mu_x}$ and using the propery $\sum_x \beta_x = 1$ we have by summing over x from (4) that

$$\sum_x \alpha_x = \sum_x \overline{\log \mu_x} = \sum_x \log \mu_{xt} - \kappa_t \sum_x \beta_x = \sum_x \log \mu_{xt} - \kappa_t \tag{7}$$

Solving for κ_t we find

$$\kappa_t = \sum_x (\overline{\log \mu_x} - \log \mu_{xt}) \quad \forall t \tag{8}$$

Q3

To find the α_x to minimize $\sum_{x,t}(\log m_{xt} - \alpha_x - \beta_x \kappa_t)^2$ we take the partial derivative with respect to α_x and equate to zero:

$$\frac{\partial}{\partial \alpha_x} \sum_{x',t} (\log m_{xt} - \alpha_x - \beta_x \kappa_t)^2 = -2 \sum_{x',t} (\log m_{x't} - \alpha_{x'} - \beta_{x'} \kappa_t) \frac{\partial \alpha_{x'}}{\partial \alpha_x} = -2 \sum_t (\log m_{xt} - \alpha_x - \beta_x \kappa_t) = 0 \tag{9}$$

where we used that $\frac{\partial \alpha_{x'}}{\partial \alpha_x} = 1$ if $x = x'$ and zero otherwise. Using $\sum_t \kappa_t = 0$ we have

$$-2 \sum_t \alpha_x = -2 \sum_t \log m_{xt} \Rightarrow \hat{\alpha}_x = \frac{1}{T} \sum_t \log m_{xt} = \overline{\log m_{xt}} \tag{10}$$

Q4

If we change the transformation to be

$$\alpha'_x \leftarrow \alpha_x + \beta_x \bar{\kappa}; \quad \kappa''_t \leftarrow \kappa_t \sqrt{\sum_x \beta_x^2}; \quad \beta'_x \leftarrow \frac{\beta_x}{\sqrt{\sum_x \beta_x^2}}; \quad \kappa'_t \leftarrow \kappa''_t - \overline{\kappa''} \quad \forall x, t \tag{11}$$

then we still have $\mu'_{xt} = \mu_{xt}$ (the proof remains the same if we change the definition of β_Σ to $\sqrt{\sum_x \beta_x^2}$) and we have the property that

$$\sum_x (\beta'_x)^2 = \sum_x \frac{\beta_x^2}{\sum_x \beta_x^2} = 1 \tag{12}$$

The script then becomes

```
kappa.LC <- kappa.LC * sqrt(sum(beta.LC^2))
beta.LC <- beta.LC/sqrt(sum(beta.LC^2))
alpha.LC <- alpha.LC + mean(kappa.LC)*beta.LC
kappa.LC <- kappa.LC - mean(kappa.LC)
```

By construction we have $\mathbf{Z} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}'$ and as remarked we also have $\mathbf{Z}\vec{1} = \vec{0}$. This implies that

$$\vec{0} = \mathbf{\Sigma^{-1}}\mathbf{U}'\vec{0} = \mathbf{\Sigma^{-1}}\mathbf{U}'\mathbf{Z}\vec{1} = \mathbf{\Sigma^{-1}}\mathbf{U}'\mathbf{U}\mathbf{\Sigma}\mathbf{V}'\vec{1} = \mathbf{V}'\vec{1} \tag{13}$$

using $\mathbf{U}'\mathbf{U}' = \mathbf{I}$ and $\mathbf{\Sigma^{-1}}\mathbf{\Sigma} = \mathbf{I}$. This shows that $\sum_{x,t} v_{xt} = 0$ so in particular $\sum_t v_{1t} = 0$. By definition $\hat{\kappa}_t = \sigma_1 v_{1t}$ so that

$$\sum_t \kappa_t = \sigma_1 \sum_t v_{1t} = 0 \tag{14}$$

Q5

The following relations hold for the SVD $\mathbf{Z} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}'$ of the matrix $\mathbf{Z}$

$$\mathbf{Z}'\mathbf{Z} = \mathbf{V}(\boldsymbol{\Sigma}'\boldsymbol{\Sigma})\mathbf{V}' \tag{15}$$

$$\mathbf{Z}\mathbf{Z}' = \mathbf{U}(\boldsymbol{\Sigma}\boldsymbol{\Sigma}')\mathbf{U}' \tag{16}$$

where the right hand sides are the eigenvalue decompositions of the left-hand sides. The columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{Z}'\mathbf{Z}$ and the columns of $\mathbf{U}$ are then eigenvectors of $\mathbf{Z}\mathbf{Z}'$. Executing the script gives the following output

```
> u1 <- eigen(Z%*%t(Z))$vectors[,1]
> v1 <- eigen(t(Z)%*%Z)$vectors[,1]
> d1 <- sqrt(eigen(t(Z)%*%Z)$values[1])
> beta.LC1 <- u1/sum(u1)
> kappa.LC1 <- v1*d1*sum(u1)
> range(beta.LC-beta.LC1); range(kappa.LC-kappa.LC1) # 'identical'
[1] -2.298509e-17  1.075529e-16
[1] -7.105427e-14  1.563194e-13
```

which implies that `beta.LC = beta.LC1` and `kappa.LC = kappa.LC1` apart from round-off errror. Given the relationship between the SVD and the eigendecomposition of $\mathbf{Z}'\mathbf{Z}$ respectively $\mathbf{Z}\mathbf{Z}'$ this output is expected. In `u1` the first eigenvector of $\mathbf{Z}\mathbf{Z}'$ is placed which corresponds to the first vector of $\mathbf{U}$, in `v1` the first eigenvector of $\mathbf{Z}'\mathbf{Z}$ which corresponds with $\mathbf{V}$ and in `d1` the eigenvalues of $\mathbf{Z}'\mathbf{Z}$. The assignment of `beta.LC1` and `kappa.LC1` is done using the transformation rules and by construction we then have that these assignments equal `beta.LC` respectively `kappa.LC`.

Q6

Running the code gives the following output

```
> library(gnm) ## install it the first time you use it
> set.seed(1)
> start <- exp(lnExt.vec + alpha.LC[x] + beta.LC[x]*kappa.LC[t])
> system.time(
+   gg <- gnm(Dxt.vec ~ 0 + offset(lnExt.vec) + x + Mult(x,t), family=poisson,
+             mustart=start, trace=TRUE)
+ ) ## ~ 13 sec
Initialising
Initial Deviance = 100722.194038
Running main iterations
Iteration 1. Deviance = 39749.368321
Iteration 2. Deviance = 25917.880223
Iteration 3. Deviance = 23870.090768
Iteration 4. Deviance = 23505.680116
Iteration 5. Deviance = 23431.747520
Iteration 6. Deviance = 23413.988074
Iteration 7. Deviance = 23408.876090
Iteration 8. Deviance = 23407.375544
```

```
Iteration 9. Deviance = 23406.912470
Iteration 10. Deviance = 23406.770225
Iteration 11. Deviance = 23406.726007
Iteration 12. Deviance = 23406.712305
Iteration 13. Deviance = 23406.708046
Iteration 14. Deviance = 23406.706724
Iteration 15. Deviance = 23406.706313
Iteration 16. Deviance = 23406.706185
Iteration 17. Deviance = 23406.706146
Iteration 18. Deviance = 23406.706133
Iteration 19. Deviance = 23406.706129
Iteration 20. Deviance = 23406.706128
Iteration 21. Deviance = 23406.706128
Iteration 22. Deviance = 23406.706128
Iteration 23. Deviance = 23406.706128
Iteration 24. Deviance = 23406.706128
Iteration 25. Deviance = 23406.706128
Iteration 26. Deviance = 23406.706128
Iteration 27. Deviance = 23406.706128
Iteration 28. Deviance = 23406.706128
Iteration 29. Deviance = 23406.706128
Iteration 30. Deviance = 23406.706128
Done
user  system elapsed
11.09    0.33   11.56
> gg$deviance; gg$iter ## 23406.706128 30
[1] 23406.71
[1] 30
```

To find the optimal parameter estimates we run `gg$coefficients` which outputs all parameter estimates. By inspection we see that the α_x paramters correspond to `gg$coefficients[1:101]`, the β_x parameters are `gg$coefficients[102:202]` (in the ouput `Mult(., t).x1` - `Mult(., t).x101`) and the κ_t parameters are `gg$coefficients[203:260]` (in the ouput `Mult(x, .).t1` - `Mult(x, .).t58`). The optimal parameter estimates are therefore given by

```
alpha.gnm <- gg$coefficients[1:101]
beta.gnm <- gg$coefficients[102:202]
kappa.gnm <- gg$coefficients[203:260]

kappa.gnm <- kappa.gnm * sum(beta.gnm)
beta.gnm <- beta.gnm/sum(beta.gnm)
alpha.gnm <- alpha.gnm + mean(kappa.gnm)*beta.gnm
kappa.gnm <- kappa.gnm - mean(kappa.gnm)
```

Q7

We plot the $\alpha_x, \beta_x, \kappa_t$ coefficients produced by the `gnm` and the LC in `R` by

```
par(mfrow=c(1,3))
```

```
plot(alpha.LC,ylim=range(alpha.LC), ylab="alpha", xlab="x", col="blue", type="l")
lines(alpha.gnm,col="red", type="l")
plot(beta.LC,ylim=range(beta.LC), ylab="beta", xlab="x", col="blue", type="l")
lines(beta.gnm,col="red", type="l")
plot(kappa.LC,ylim=range(kappa.LC), ylab="kappa", xlab="t", col="blue", type="l")
lines(kappa.gnm,col="red", type="l")
```

which gives Figure 1.

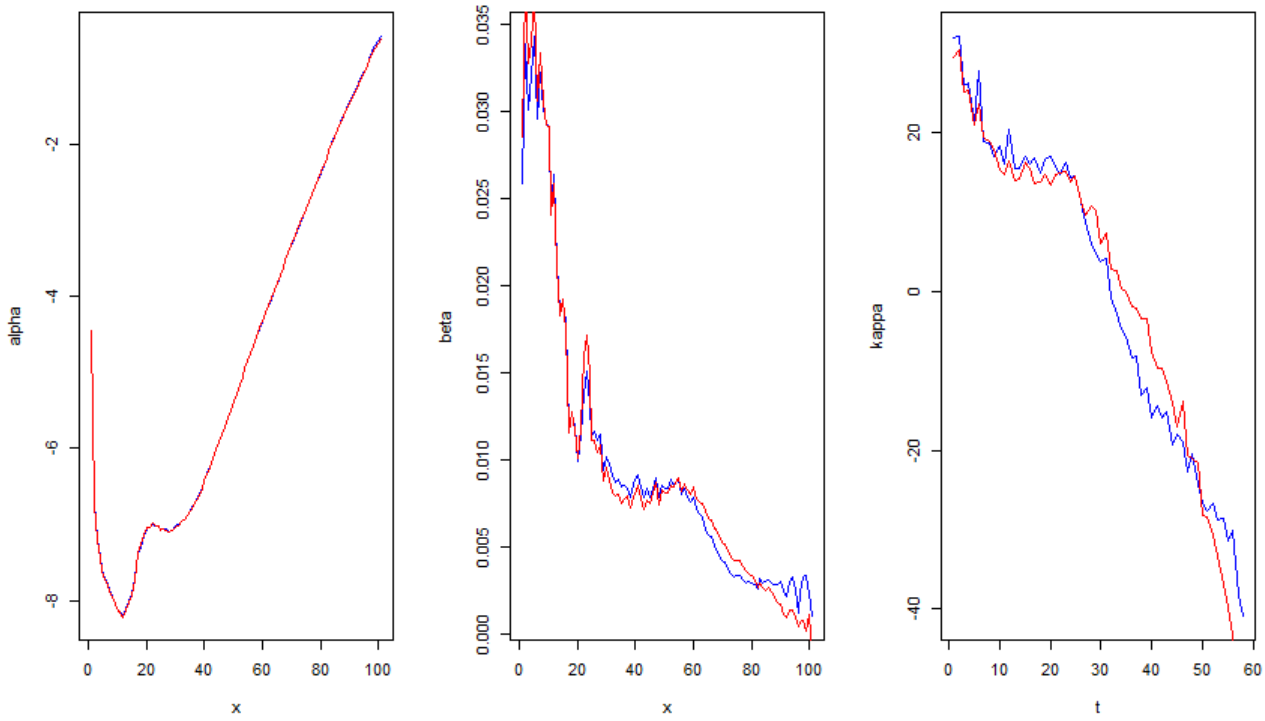

Figure 1: Coefficients $\alpha_x, \beta_x, \kappa_t$ produced by gnm (red lines) and LC (blues lines)

Q8

From equation (3) from the assignment we know that the relationship between α_x and μ_{xt} is given by $\mu_{xt} = \exp(\alpha_x + \beta_x \kappa_t)$. The increase from the log-mortality α_x (keeping all other parameters fixed) from -8 to -7 from age 16 to 18 implies that μ_{xt} increases by $\exp(1) \approx 2.72$. In the assignment about Gompertz and Makeham we already have seen this phenomenon and it is named the 'accident hump'.

Q9

We execute the code and add one line to view the coefficients of the object `g1`

```
> kappa.glm <- kappa.LC
> g1 <- glm(Dxt.vec ~ x*kappa.glm[t] + offset(lnExt.vec), poisson)
```

```
> c1 <- coef(g1)
> g1$deviance; g1$iter ## 27603.07 4
[1] 27603.07
[1] 4
> c1
```

the output given from the call `c1` is an intercept, `x2 - x101` and `kappa.glm[t]`,`x2:kappa.glm[t] - x101:kappa.glm[t]`. We can then retrieve the optimal parameter estimates by (using the intercept resp. `kappa.glm[t]` as base contribution)

```
alpha.glm <- c(c1[1],c1[2:101]+c1[1])
beta.glm <- c(c1[102],c1[103:202]+c1[102])
```

Q10

We run the code as given in the assignment and obtain the following results

```
> g2 <- glm(Dxt.vec ~ 0 + x + t:beta.glm[x] + offset(lnExt.vec), poisson,
+           mustart=fitted(g1))
> c2 <- coef(g2)
> g2$deviance; g2$iter ## 23594.62 4
[1] 23594.62
[1] 4
> c2[c(1,nages,nages+1,nages+nyears-1,nages+nyears)] ## t58:beta.glm[x] is NA
x1            x101  t1:beta.glm[x] t57:beta.glm[x] t58:beta.glm[x]
-6.0994748      -0.5598663      88.0965362       5.6195332              NA
```

We see that the last coefficient `t58:beta.glm[x]` is `NA`. Using the command `summary(g2)` we see that 1 coefficient is not defined because of singularities. Therefore the last coefficient is `NA`. Since there is no base contribution (no constant in the regression) we can use the following to assign α_x and κ_t (adding a zero for the parameter κ_{58} because of the `NA` result)

```
alpha.glm <- c2[1:101]
kappa.glm <- c(c2[102:158],0)
kappa.glm <- kappa.glm * sum(beta.glm)
beta.glm <- beta.glm/sum(beta.glm)
alpha.glm <- alpha.glm + mean(kappa.glm)*beta.glm
kappa.glm <- kappa.glm - mean(kappa.glm)
```

where the transformations ensure we have the properties of Question 1.

Q11

Using the inverse link function we can calculate `fitted(g2)[532]` in terms of `Ext.vec[532]`, `alpha.glm[x[532]]`, `beta.glm[x[532]]` and `\verb|kappa.glm[t[532]])| as`

```
exp(log(Ext.vec[532])+alpha.glm[x[532]]+beta.glm[x[532]]*kappa.glm[t[532]])
```

We check in R if the two are equal by determining if the absolute value of the difference is small and obtain the following output

```
> abs(exp(log(Ext.vec[532])+alpha.glm[x[532]]+
beta.glm[x[532]]*kappa.glm[t[532]])-fitted(g2)[532])<0.0001
x10
TRUE
```

which implies that the reconstruction and the value of `fitted(g2)[532]` are equal (checked for a maximum difference of 0.0001).

Q12

We execute the code from the assignment and obtain

```
> beta.glm <- beta.glm/sum(beta.glm)
> alpha.glm <- alpha.glm + mean(kappa.glm)*beta.glm
> kappa.glm <- kappa.glm - mean(kappa.glm)
> (d1 <- sum(dpois(Dxt.vec,Dxt.vec,log=TRUE))) ## -21643.76
[1] -21643.76
> (d2 <- sum(dpois(Dxt.vec,
+                  Ext.vec*exp(alpha.glm[x] + beta.glm[x] * kappa.glm[t]),
+                  log=TRUE))) ## -33441.07, same as d3
[1] -33441.07
> (d3 <- sum(dpois(Dxt.vec,fitted(g2),log=TRUE)))
[1] -33441.07
> (d4 <- log(prod(dpois(Dxt.vec,fitted(g2)))))## -Inf
[1] -Inf
> (d5 <- 2*sum(Dxt.vec*log(Dxt.vec/fitted(g2)) - (Dxt.vec-fitted(g2))))
[1] 23594.62
> (d1-d2)*2 ## 23594.62, same as d5
[1] 23594.62
```

We see that the result `d2` equals `d3` and that `d5` equals `(d1-d2)*2` . The results `d2` and `d3` are equal by construction (see Question 11). The result from `d4` should be equal to the result from `d3`. The difference is using the `log=TRUE` option in `d3` instead of taking the logarith from the products in `d4`. The result in `d4` is not equal because `R` has difficulties in taking the logarithm of the very small probabilities. The equality of `d5` and `(d1-d2)*2` follows from MART (9.29). The calculation `d5` is equal to right-hand side of MART (9.29) (with ϕ and w_i equal to zero) and `(d1-d2)*2` is equal to the middle expression of MART (9.29).

Q13

We execute the code from the exercise and obtain the following result.

```
> range(tapply(Dxt.vec-fitted(g2),x,sum)) ## -3e-10 2e-10
[1] -3.885816e-10  2.038405e-10
```

```
> range(tapply(Dxt.vec-fitted(g2),t,sum)) ## -3300 2726
[1] -3299.962  2726.266
>
```

The first line shows the range of the differences of the data versus the fit with respect to the ages x and the second line shows the range of the differences of the data versus the fit with respect to the time t. The result shows that the residuals are very small with respect to the ages but can be significant with respect to time (residuals up between -3300 and 2726).

Q14

We run the code from the assignment and obtain the given results

```
> kappa.glm <- kappa.LC
> oldDeviance <- 0; TotnIter <- 0; start=NULL
> system.time(
+   repeat
+   { g1 <- glm(Dxt.vec~x*kappa.glm[t]+offset(lnExt.vec), poisson, mustart=start)
+   c1 <- coef(g1)
+   alpha.glm <- c(c1[1],c1[2:101]+c1[1])
+   beta.glm <- c(c1[102],c1[103:202]+c1[102])
+   g2 <- glm(Dxt.vec ~ 0+x + t:beta.glm[x] + offset(lnExt.vec), poisson,
+             mustart=fitted(g1))
+   8
+   c2 <- coef(g2)
+   alpha.glm <- c2[1:101]
+   kappa.glm <- c(c2[102:158],0)
+   kappa.glm <- kappa.glm*sum(beta.glm); beta.glm <- beta.glm/sum(beta.glm);
+   alpha.glm <- alpha.glm + mean(kappa.glm)*beta.glm
+   kappa.glm <- kappa.glm - mean(kappa.glm)
+   TotnIter <- TotnIter + g1$iter + g2$iter
+   newDeviance <- g2$deviance;
+   done <- abs((oldDeviance-newDeviance)/newDeviance)<1e-6
+   cat(g1$deviance, "\t", g2$deviance, "\n")
+   oldDeviance <- newDeviance; start <- fitted(g2)
+   if (done) break
+   }
+ ) ## ~ 6 sec
27603.07   23594.62
23416.47   23407.23
23406.74   23406.71
23406.71   23406.71
user  system elapsed
4.54    0.19    4.74
> TotnIter ## 20
[1] 20
>
> AIC(g1); AIC(g2) ## 67098.22 67010.22
```

```
[1] 67098.22
[1] 67010.22
> logLik(g1); logLik(g2) ## 'log Lik.' -33347.11 with df=202 and df=158
'log Lik.' -33347.11 (df=202)
'log Lik.' -33347.11 (df=158)
```

The calculated log-likelihoods are equal for both models. Because not all parameters are determined in `g1` and `g2` the calculated AIC is not correct with respect to the complete model. Since the total effective number of paramters is $101 + 101 + 58 - 2 = 258$ (101 α_x parameters, 101 β_x parameters, 58 κ_t parameters and a redundancy of 2) the actual AIC calculation is equal to $AIC = -2l + 2k = -2(-33347.11 - 258) = 67210.22$ which was obtain from the `R` output

```
> -2*(logLik(g1)-258)
'log Lik.' 67210.22 (df=202)
```

Q15

The $\mathbf{X}$ matrix has 5858 rows from the observations. For the `g1` model we estimated the α_x and β_x parameters so the matrix $\mathbf{X}$ has 202 columns. We check the dimensions of the matrix in `R` and obtain the following output

```
> dim(model.matrix(g1))
[1] 5858  202
```

which are the dimensions we expected. Using the command given in the question we determine the amount of memory for each of the objects `g1` and `g2` and obtain

```
> object.size(g1); object.size(g2);
13254648 bytes
11130864 bytes
```

To determine which part of `g1` occupies the most space we use the command given and obtain the following output

```
> sort(sapply(g1,object.size))
rank              deviance   aic            null.deviance iter   df.residual
48                48         48             48            48     48
df.null           converged  boundary       data          method contrasts
48                48         48             56            96     360
control           formula    call           xlevels       terms  coefficients
544               1720       1904           5952          6840   15408
family            offset     effects        R             residuals fitted.values
45880             46904      105952         354080        375104    375104
linear.predictors weights    prior.weights  y             model  qr
375104            375104     375104         375104        603456 9811960
```

which shows that the `qr` part of the object `g1` occupies the most space.

Q16

Q17

Q18