Universiteit van Amsterdam

Amsterdam School of Economics

Computer class NLIST—Assignment 3B

GLMs and the Lee-Carter mortality model

This assignment is about the Lee-Carter mortality model and its relation to GLMs, as well as about methods to estimate its parameters.

1 The model of Lee-Carter

Mortality data as recorded in various databases count the number D_{xt} of people of a certain age x dying in a particular year t, as well as the number e_{xt} exposed to that risk. Lee-Carter (1992) used models of the form

$$\log\left(\frac{D_{xt}}{e_{xt}}\right) \approx \alpha_x + \beta_x \kappa_t; \quad x = 1, \ldots, X, \ t = 1, \ldots, T. \tag{1}$$

This model underlies practically every study on forecasting mortality trends. For this, one assumes that the α_x and β_x remain constant over time, and employs every trick in the book to expand the κ_t time series. In this assignment, we will not try to forecast the κ_t, but concentrate on finding 'good' parameters for (1).

The Lee-Carter algorithm looks for parameters of model (1) that minimize the sum of squared differences of rhs and lhs. For this to be statistically well-founded, the logs of the mortality rates in fact should be normally distributed. As outlined in Chapter 9 of MART, transforming Poisson variables by taking logs will ensure additive effects, but not homoskedasticity or symmetry in the sense of zero skewness. For example, from the delta method approximation MART (9.1) we can derive that if $X \sim \text{Poisson}(\lambda)$ with large λ and $Y = \log(X + 1/2)$, then

$$\text{E}[Y] \approx \log\lambda \text{ (log-linearity)}, \quad \text{Var}[Y] \approx \frac{1}{\lambda}, \quad \text{and} \quad \gamma_Y \approx \frac{-2}{\sqrt{\lambda}}. \tag{2}$$

Though they have skewness zero asymptotically, for small λ log-transformed Poisson random variables are far from symmetric, therefore not close to normal. Because of the negative skewness, positive errors $Y - \log\lambda$ tend to be smaller in absolute value than negative ones. Also, they are not homoskedastic. Logs of observations D_{xt} in (1) with large exposure have very small variance so they should be fitted perfectly, while for those corresponding to small λ's a large prediction error is not so bad.

In fact the D_{xt} are like binomial random variables, a 'success' in the e_{xt} independent trials being that someone dies. Since the probabilities of such a success are small in general and mortality probabilities are especially low when exposures are high, a Poisson approximation is a good choice because of its simplicity, its natural bond with the log-link between mean and covariates as in (2), and its good fit. So instead of transforming our variables so they

approach normality enabling us to use least squares reliably, we aim to maximize a Poisson likelihood with a log-link, using the logs of the e_{xt} as an offset.

The Poisson version of the Lee-Carter mortality model (1), see e.g. Brouhns et al. (2002), is

$$D_{xt} \overset{\text{ind.}}{\sim} \text{Poisson}(e_{xt}\mu_{xt}) \qquad \text{with} \qquad \mu_{xt} = \exp(\alpha_x + \beta_x\kappa_t). \tag{3}$$

This looks like an ordinary Poisson GLM with a log-link and an offset term, but the predictor $\log(e_{xt}\mu_{xt})$ is not quite linear in the parameters $\alpha_x, \beta_x, \kappa_t$; in fact it is bi-linear.

In (3) there are two parameters too many; to ensure identifiability it is customary to take $\sum_t \kappa_t = 0$ and $\sum_x \beta_x = 1$. There is no guarantee that $\sum_x \beta_x \neq 0$, so the latter condition should perhaps be replaced by e.g. $\sum_x \beta_x^2 = 1$, making the norm of $\vec{\beta}$ equal to 1.

Q_1 Show that if $\beta_\Sigma := \sum \beta_x \neq 0$, the parameter choices $\alpha'_x, \beta'_x, \kappa'_t$ below have the same values of μ_{xt} as in (3), but satisfy $\sum \beta'_x = 1$ and $\sum \kappa'_t = 0$:

$$\alpha'_x \leftarrow \alpha_x + \beta_x\overline{\kappa}; \quad \kappa''_t \leftarrow \kappa_t \times \beta_\Sigma; \quad \beta'_x \leftarrow \beta_x/\beta_\Sigma; \quad \kappa'_t \leftarrow \kappa''_t - \overline{\kappa''} \quad \forall x, t. \tag{4}$$

Q_2 Prove that the restrictions $\sum_x \beta_x = 1$ and $\sum_t \kappa_t = 0$ on the parameters in (3) imply the following relations:

$$\alpha_x = \frac{1}{T}\sum_t \log \mu_{xt} =: \overline{\log \mu_x} \text{ for all } x; \qquad \kappa_t = \sum_x (\log \mu_{xt} - \overline{\log \mu_x}) \text{ for all } t. \tag{5}$$

□

So assuming without loss of generality that these restrictions apply, we have the following interpretation for the parameters in (3):

- α_x is the mean log-mortality over time at age x;
- κ_t is the sum over all ages of the excess of log-mortality over its mean at time t;
- β_x describes the amount of log-mortality change at a given age x for a unit of yearly total log-mortality change.

The predictor on the rhs of (3) being not quite linear in the parameters means that maximizing its likelihood cannot be done directly by invoking the `glm`-function. Model (3) does, however, fit directly in the framework of Generalized Non-linear Models, so it can be estimated using R's `gnm` function from the library of that same name. See for example Turner and Firth (2012), especially Sec 7.7. A solution to the least-squares problem (1) that has optimal α_x and near optimal β_x and κ_t can be found very quickly using a singular value decomposition (SVD). See Section 3. From that starting solution, parameters maximizing the Poisson likelihood can be found by calling `gnm`. Other starting solutions may work as well, but using the initial solution provided by the SVD we start from a point that is hopefully close enough to the global optimum to avoid ending up in a local optimum. See Section 4.

Another approach, similar to the successive substitution method to minimize the Bailey-Simon distance in MART, Ch. 9, is to use `glm` calls iteratively. Successively one set of

parameters is kept constant and the remainder optimized, then the other way around. If either the β_x or the κ_t are fixed, an ordinary GLM remains from (3); see Section 5.

In the final Section 6, for each period t, parameters for the Gompertz/Makeham life-time distribution are estimated and analyzed.

2 The data

In this assignment we will consider Dutch mortality data about calendar years $t = 1, \ldots, 58$ concerning numbers of people of age $x = 0, \ldots, 100$ alive and dying in year t. Such data can be downloaded from the site the CBS or from the Human Mortality Database.There are many ways to get these data in a format readable by `R`. For your convenience, we fed them to Excel and stored them in the cloud as a csv-file, with comma separated values. In D_{xt}, the number of people of age x that died in year t is recorded, while e_{xt} is the initial exposure, that is, the number of people of age x alive at the start of year t. Both matrices D_{xt} and e_{xt} will be stored as vectors of length $58 \times 101 = 5858$, and the row and column numbers corresponding to the entries will also be stored in vectors. As is the custom in e.g. the Netherlands and Belgium, the numbers have a decimal comma rather than a decimal point to separate integer and fractional parts; see also Wikipedia. Therefore they are separated by semi-colons. Sometimes non-integer estimates rather than actual numbers of deaths were inserted. We work with rounded numbers so as to be able to use Poisson regression rather than just quasi-Poisson.

```
path <- "http://www1.fee.uva.nl/ke/act/people/kaas/"
nages <- 101; nyears <- 58
Dxt.vec <- scan(paste(path,"deaths.csv",sep=""), sep=";", dec=",")
Ext.vec <- scan(paste(path,"exposures.csv",sep=""), sep=";", dec=",")
Dxt.vec <- round(Dxt.vec)
x <- gl(nages,nyears,nages*nyears); t <- gl(nyears,1,nages*nyears)
lnExt.vec <- log(Ext.vec)
```

The vectors `x` and `t`, generated by calls to `gl`, are of type factor, for later use in calls of `glm` and `gnm`. They contain row and column number of the matrices D_{xt} and e_{xt} as stored into the vectors `Dxt.vec` and `Ext.vec`.

3 Using LC to find a good initial solution

Often a crude initial guess for the parameters $\alpha_x, \beta_x, \kappa_t$, followed by either a call of `gnm` or repeated `glm`-calls will do the job. One might take $\alpha_x = 0, \beta_x = 1, \kappa_t = T - t$. A more sophisticated way to find starting values is to look for the parameters minimizing the sum of squared errors in (1), providing a distance function similar to the deviance of (3):

$$\min_{\alpha_x,\beta_x,\kappa_t} \sum_{x,t} \left(\log m_{xt} - \alpha_x - \beta_x\kappa_t\right)^2, \quad \text{with} \quad m_{xt} = \frac{D_{xt}}{e_{x,t}}. \tag{6}$$

As noted, to ensure identifiability we take $\sum_t \kappa_t = 0$ and $\sum_x \beta_x = 1$.

Q_3 By setting zero the partial derivative with respect to α_x, show that the $\widehat{\alpha}_x$ minimizing (6) must satisfy $\widehat{\alpha}_x = \overline{\log m_x}$, the average being taken over t. □

Because we showed in Q_2 that $\alpha_x = \overline{\log \mu_x}$, the LS-estimate turns out to be a method-of-moments estimate.

Now let $X \times T$ matrix $\mathbf{Z}$ have (x,t) element $Z_{xt} = \log m_{xt} - \widehat{\alpha}_x$. By construction, $\mathbf{Z}$ has row sums zero, therefore an eigenvalue zero. Matrix $\mathbf{Z}$ has the following SVD decomposition:

$$\mathbf{Z} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}'. \tag{7}$$

Here $\mathbf{U}$ and $\mathbf{V}$ are real orthogonal matrices, $\mathbf{U}$ is $X \times X$, $\mathbf{V}$ is $T \times T$, while $\mathbf{\Sigma}$ is an $X \times T$ matrix with non-negative real elements $\sigma_i := \sigma_{ii}$ on the diagonal and zeroes elsewhere. The diagonal entries σ_i are known as the singular values of $\mathbf{Z}$. A common convention is to list the singular values in descending order, so σ_1 is the largest, while the smallest $\sigma_{\min(X,T)} = 0$. It is not mentioned in the help-file for R's `svd()` routine, but this convention is also adhered to by the LAPACK function invoked by `svd`, see the Intel developer zone.

Now define $\vec{u}_i$ to be the i^{th} column of $\mathbf{U}$, and $\vec{v}_i$ to be the i^{th} columnn of $\mathbf{V}$. Then from (7),

$$\mathbf{Z} = \sum_i \vec{u}_i \sigma_i \vec{v}_i{}'. \tag{8}$$

A good approximation to $\mathbf{Z}$ is its principal component, that is, the first term in this sum. See also the section 'Low-rank matrix approximation', in this case rank 1, on the SVD Wikipedia page. So the optimal solution to (6) is

$$\widehat{\alpha}_x = \overline{\log m_x}; \quad \widehat{\beta}_x = u_{x1}; \quad \widehat{\kappa}_t = \sigma_1 v_{1t}. \tag{9}$$

In summary, we get a good starting solution for our likelihood maximization problem (3) by:

```
Z <- matrix(Dxt.vec, nrow=nages, ncol=nyears, byrow=TRUE)
Z[c(1,2,nages-1,nages), c(1,2,nyears-1,nyears)] # to check if the same
xtabs(Dxt.vec~x+t)[c(1,2,nages-1,nages), c(1,2,nyears-1,nyears)]
Z <- log((Z+.5)/matrix(Ext.vec, nrow=nages, ncol=nyears, byrow=TRUE))
alpha.LC <- rowMeans(Z)
Z <- Z - alpha.LC
s <- svd(Z)
beta.LC <- s$u[,1]/sum(s$u[,1])
kappa.LC <- s$v[,1]*s$d[1]*sum(s$u[,1])
```

Note that instead of taking observed log-mortalities $\log m_{xt} = \log \frac{D_{xt}}{e_{xt}}$, we use $\log \frac{D_{xt}+1/2}{e_{xt}}$. By adding a positive constant to D_{xt} we avoid the problem that with positive probability, the event $D_{xt} = 0$, so $\log D_{xt} = -\infty$, occurs. And in MART, Ch. 9, it is shown that when $X \sim \text{Poisson}(\lambda)$, $\text{E}[\log(X + 1/2)] = \log \text{E}[X]$ holds almost perfectly even for very small λ.

By (4), to make parameters satisfy $\beta_\Sigma = 1$ and $\kappa_\Sigma = 0$ without changing fitted values we must do

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

Q_4 A perhaps better way to standardize is to let $\sum(\beta_x)^2 = 1$, since it is not certain that $\sum \beta_x \neq 0$ holds. Change the above script accordingly. But run the old transformation after this, because we will assume $\sum \beta_x = 1$ later on.

In fact, the values resulting from `svd` already satisfy $\beta_\Sigma = 1$ and $\kappa_\Sigma = 0$. By construction, `sum(beta.LC) == 1` holds before the transformations, but show that then `sum(kappa.LC) == 0` holds as well.

Hint: $\mathbf{Z}$ has row sums zero, so $\mathbf{Z}\vec{1} = \vec{0}$ if $\vec{1}$ and $\vec{0}$ are column vectors of ones and zeros respectively. Now consider $\mathbf{\Sigma}^{-1}\mathbf{U}'\mathbf{U}\mathbf{\Sigma}\mathbf{V}'\vec{1}$, with $\mathbf{\Sigma}^{-1}$ a $T \times X$ matrix with diagonal elements $1/\sigma_{ii}$, $i = 1, \ldots, T$, so $\mathbf{\Sigma}^{-1}\mathbf{\Sigma} = \mathbf{I}_T$.

□

Q_5 Find here the relation between SVD and eigenvalue decomposition. Now argue why the following script leads to the same solution found earlier (apart from round-off error):

```
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'
```

□

We do some checks on dimensions of matrices:

```
D <- diag(s$d)
dim(s$u); dim(D); dim(t(s$v)) # X = 101 rows x T = 58 columns;  58x58;  58x58
dim(s$u %*% D %*% t(s$v)) # 101x58
dim(t(s$u) %*% Z %*% s$v) # 58x58
range(s$u %*% D %*% t(s$v)-Z) #  Z = U D V'
range(t(s$u) %*% Z %*% s$v - D) #  D = U' Z V
```

With $X = n = 101$ and $T = p = 58$ here, the dimensions of the returned objects `s$u, s$v, s$d` are as stated in `?svd`. Matrix $\mathbf{U}$ on the Wikipedia page would be $n \times n$ and $\mathbf{\Sigma}$ would be $n \times p$. In fact, $\mathbf{\Sigma} = [\mathbf{D}, \mathbf{0}]$ holds, and (7) is valid in both cases.

4 Using `gnm` to find the optimal solution

Model (3) is a rather simple example of a Generalized Non-linear Model. So to solve it optimally is easy using `gnm`. As a starting solution, the fitted values resulting from the estimated coefficients found above are used.

```
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
gg$deviance; gg$iter  ## 23406.706128 30
```

Note that as opposed to `glm`, the function `gnm` in its initial phase uses a *randomized* solution from which to start; running it again gives a different path to hopefully the same solution. From the last few iterations it can be seen that the criterion for convergence involves more than just noting a very small change in the deviance. The `gnm`-call takes quite some time.

Q_6 From `gg`, derive the optimal parameter estimates that satisfy the identification constraints. □

Q_7 Draw plots of the $\alpha_x, \beta_x, \kappa_t$ coefficients produced by `gnm` and LC. □

Q_8 In the plot of α_x it can be seen that the log-mortality $\alpha_x = \log \overline{\mu_x}$ increases from, say, -8 to -7 from age 16 to 18. Ignoring the contribution of the $\beta_x \kappa_t$ term, which averages to zero over time, what does this mean for the ratio of the mortality rates at these ages? □

5 Finding the optimal solution using iterative GLMs

Recall the Bailey-Simon Method 9.3.5 in MART. It uses a fixed point algorithm (by successive substitution) to find a solution to the set of first order conditions that hopefully gives the optimal solution to the distance minimization problem. This involves calculating optimal row coefficients, treating the column coefficients as fixed and known, then keeping the row coefficients fixed and calculating the best corresponding column coefficients, and repeating these steps until the process has converged. Observe that if the κ_t (or the β_x) are fixed and known, (3) is just an ordinary GLM. So optimal coefficients in (3) can be found by a similar iteration, where optimal must be understood as 'cannot be improved by choosing better parameters when certain subsets of the parameters are kept fixed'. In the Bailey-Simon problem, the parameters kept fixed are either all row coefficients or all column coefficients, in (3), either all β_x or all κ_t.

Vector `kappa[t]` expands to `(kappa[t[1]], ..., kappa[t[5858]])`, containing the `kappa` values associated with each observation D_{xt} as stored in the vector `Dxt.vec`. So to find the optimal α_x and β_x with these fixed κ_t, we treat `kappa[t]` as a numeric covariate, and take its interaction with the factor `x`. Taking such an interaction as a model term gives a coefficient for intercept α_x and slope β_x for every x. As starting values for κ_t, we may take some decreasing sequence like $T, ..., 1$, or, in accordance with (5), the column sums of $\mathbf{Z}$. But we choose the values resulting from (9).

```
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
alpha.glm <- ...
beta.glm <- ...
```

Q_9 By inspecting the object `g1`, find out how to fill in the dots above to retrieve the optimal parameter estimates `alpha.glm` and `beta.glm` in model (3), assuming κ_t known. □

In model `g1`, the predictor for cell (x, t) is $\eta_{xt} = \alpha_x + \beta_x \kappa_t + 1 \times \log(e_{xt})$, where κ_t, $t = 1, 2, \ldots$, is fixed, α_x, β_x, $x = 0, 1, \ldots$ are parameters to be optimally estimated, and the offset $\log(e_{xt})$ is a covariate with coefficient fixed to be 1. This predictor is linear in the unknown parameters α_x, β_x.

In the next step, we estimate κ_t and α_x optimally, using the same linear predictor $\eta_{xt} = \alpha_x + \kappa_t \beta_x + 1 \times \log(e_{xt})$, again with an offset $\log(e_{xt})$ but this time with β_x fixed. Note that we now start at the optimum of `g1`.

```
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
c2[c(1,nages,nages+1,nages+nyears-1,nages+nyears)]  ## t58:beta.glm[x] is NA
alpha.glm <- ...
kappa.glm <- ...
```

As we started from the fitted values of `g1`, the deviance of `g2` is guaranteed to be lower than the one of `g1`. Note that the model term `t:beta.glm[x]` gives a term $\kappa_t \beta_x$ in the linear predictor, but not a coefficient for every t (interaction only, not the main effect). Using the term `0+x` (no constant in the regression) rather than just `x`, where `x` is a factor, does not affect the linear predictors, but makes it easier to extract the α_x coefficients.

Q_{10} Explain why the final element of `c2`, that is, the coefficient with `t58:beta.glm[x]`, is `NA`.

Fill in the dots to retrieve from the object `g2` the optimal parameters `alpha.glm` and `kappa.glm` in (3), assuming β_x known. □

Q_{11} Noting that observation 532 has `x[532] = t[532] = 10`, reconstruct `fitted(g2)[532]` using the (inverse) link function, exposure `Ext.vec[532]` and parameter estimates `alpha.glm[x[532]]`, `beta.glm[x[532]]` and `kappa.glm[t[532]]`.

□

As noted, estimating parameters $\alpha_x, \beta_x, \kappa_t$ for all x and t leads to identifiability problems, since many different parameter vectors give the same values for the linear predictors and hence for the deviance. The deviance thus has a lot of equivalent minima, which might upset the minimization. We may resolve the ambiguity by taking the vectors with the property

that $\sum \beta_x = 1$ as well as $\sum \kappa_t = 0$, facilitating interpretation, but `glm` just applies corner restrictions, dropping the first or last parameter of a set.

Q_{12} In the following R-output, explain why `d2` and `d3` are equal, why `d3` and `d4` are not equal, and why the last two results are equal (cf. (9.29) in MART).

```
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)

(d1 <- sum(dpois(Dxt.vec,Dxt.vec,log=TRUE))) ## -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
(d3 <- sum(dpois(Dxt.vec,fitted(g2),log=TRUE)))
(d4 <- log(prod(dpois(Dxt.vec,fitted(g2)))))## -Inf
(d5 <- 2*sum(Dxt.vec*log(Dxt.vec/fitted(g2)) - (Dxt.vec-fitted(g2))))
(d1-d2)*2 ##  23594.62, same as d5
```

□

Q_{13} Execute the following script, and comment on the results:

```
range(tapply(Dxt.vec-fitted(g2),x,sum)) ##  -3e-10  2e-10
range(tapply(Dxt.vec-fitted(g2),t,sum)) ##  -3300   2726
```

□

Iteration method

After the initial guess for κ_t, we found the optimal corresponding β_x (and α_x), and from these, the best κ_t (and α_x) values with these coefficients. Now we iterate this process, starting from the newly found κ_t. After a few iterations, the deviance does not change visibly anymore.

Note that though the likelihood found cannot be improved by changing either the α_x, β_x parameters or the α_x, κ_t parameters, changing *all* parameters simultaneously might lead to higher values. And even if a parameter set gives a maximum, it could be just a local one.

The iteration procedure, including the initial steps shown earlier in this section, goes as follows:

```
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 <- ...; beta.glm <- ... ## See Q9

    g2 <- glm(Dxt.vec ~ 0+x + t:beta.glm[x] + offset(lnExt.vec), poisson,
              mustart=fitted(g1))
```

```
    c2 <- coef(g2)
    alpha.glm <- ...; kappa.glm <- ... ## See Q10

    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
TotnIter  ##  20
```

Iteration runs smoothly, and in total, the `glm` calls use `TotnIter = 20` Fisher scoring iteration steps. They do take some time, to be expected since the number of parameters estimated is 202 in `g1` and 158 in `g2`, with $101 \times 58 = 5858$ observations. The sequence of deviances printed is decreasing and converges quickly:

27603.07	23594.62
23416.47	23407.23
23406.74	23406.71
23406.71	23406.71

Q_{14} AIC is minus twice the maximized log-likelihood plus twice the effective number of parameters, that is, the number of parameters minus the number 'substituted away' through linear restrictions for identifiability. Calling `AIC(g1)` and `AIC(g2)` leads to `67098.22` and `67010.22`; the loglikelihood `logLik(g1)` equals `logLik(g2)` except for the parameters. But what is the actual AIC of model (3)?

```
AIC(g1); AIC(g2)  ##  67098.22  67010.22
logLik(g1); logLik(g2)  ## 'log Lik.' -33347.11 with df=202 and df=158
```
□

Q_{15} (♠) Ordinary weighted least squares problems must be solved in each iteration step in a `glm` call, requiring computation of $(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\vec{Z}$; see MART (11.59). What are the dimensions of the $\mathbf{X}$-matrix in `g1`? Check with what you find from `dim(model.matrix(g1))`. So what dimensions does $\mathbf{X}'\mathbf{W}\mathbf{X}$ have? Using the `object.size()` function, find out how much memory objects `g1` and `g2` occupy. From `sort(sapply(g1,object.size))`, find out which part of the `g1` object occupies most of the space. □

6 Comparison with Makeham/Gompertz

The problem in (3) has quite a lot of parameters: twice the number of ages considered plus the number of time periods. One might want to reduce this number for example by imposing

a functional relation on the parameter vectors. They might be restricted to be constant, linear, quadratic, of Hoerl-type, of some mixed type, and so on.

Simple distribution functions describing mortality probabilities over all ages are Makeham's and Gompertz' laws, having only three and two parameters, respectively. Compare the one-year mortality probabilities when life-times obey Makeham's law and the Lee-Carter model:

$$a + bc^x \qquad \text{and} \qquad \exp(\alpha_x + \beta_x \kappa_t).$$

To make them equal for all ages x and calendar years t, we can take $a = 0$ (reducing to Gompertz), $b = \mathrm{e}^{\alpha_x}$, $\beta_x = x$ and $c = \mathrm{e}^{\kappa_t}$, therefore (taking logarithms)

$$\log b + x \log c = \alpha + \kappa x.$$

This is a linear form in the parameters, meaning that we can find ML-estimates for the Gompertz parameters b and c by one call of `glm` with a log-link, taking $b = \mathrm{e}^{\alpha}$, $c = \mathrm{e}^{\kappa}$ with α and κ the intercept and slope of the regression trend line.

Q_{16} Find out which parameters $\widehat{b}$ and $\widehat{c}$ maximize the Gompertz(b, c) likelihood for this dataset.

Calling `lines(...)`, add a plot of the α_x found using $\alpha_x \equiv \log \widehat{b}$, $\kappa_t \equiv \log \widehat{c}$ and $\beta_x = x$, after the transformation making $\sum \beta_x = 1$ and $\sum \kappa_t = 0$.

Do the same for when the estimates are restricted to ages 30+, by adding the argument `subset = x1>=30` to the glm-call producing `g3`. Here `x1` is a variate containing the real ages $x = 0, 1, \ldots$ corresponding to indices `x = 1,2,...`, produced by `x1 <- as.numeric(x)-1`. □

Note that the α_x parameters in (5), and approximately also in the optimal solution to (3), represent the mean log-mortality over time at age x. When derived from Gompertz' law, they completely miss several important features: the infant mortality and the so-called 'accident hump' at age around 20. Also, the annual increase in mortality is somewhat steeper after age 30, being about 10.3% rather than the 8.7% resulting when considering all ages. And of course there is no evolution of mortality in time, the κ_t being constant.

With log-mortality close to linear after age $x = 30$ or so, it appears that the adult Dutch population closely follows Gompertz' law. This means that the mortality, aggregated over time t, increases by a constant factor c each year, so $\mu_{x\Sigma} = b\, c^x$. The parameters can be estimated by a single `glm()` call.

In fact, by including as a model term `x1*t`, that is, the interaction of the variate `x1` and the factor `t`, we can estimate a slope $\log c_t$ and an intercept $\log b_t$ for every individual time t, with in fact $\log \mu_{xt} = \log b_t + x \log c_t$. The following script produces a plot of the evolution of the b_t, c_t parameters of the Gompertz distribution fitted to the number of deaths.

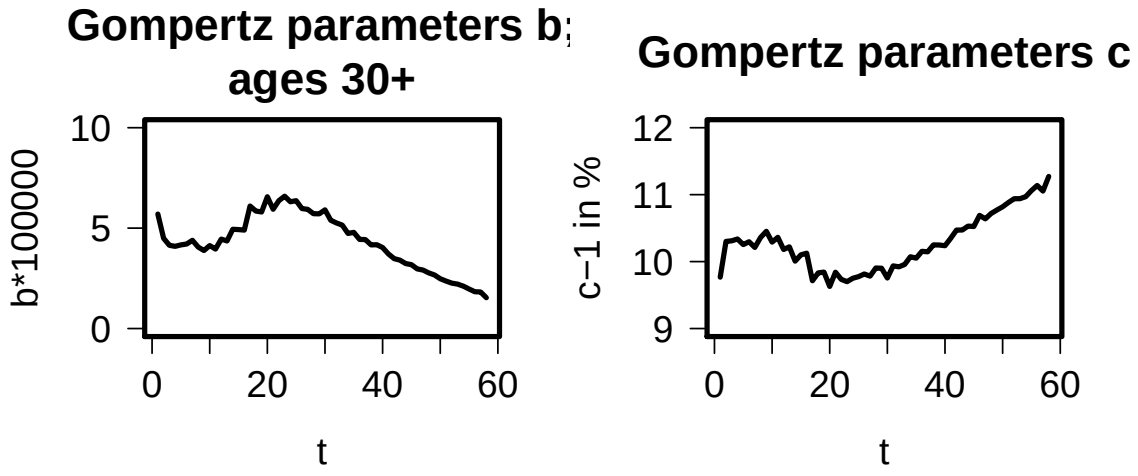
```
g4 <- glm(Dxt.vec~x1*t-x1-1+offset(lnExt.vec), poisson, subset=x1>=30)
b <- 1e5*exp(head(coef(g4),nyears)); c <- 100*(exp(tail(coef(g4),nyears))-1)
par(mfrow=c(1,2))
plot(b, xlab="t", ylab="b*100000", ylim=c(0,10), type="l", yaxp=c(0,10,2),
     main="Gompertz parameters b;\nages 30+")
plot(c, xlab="t", ylab="c-1 in %", ylim=c(9,12), type="l",
     yaxp=c(9,12,3), main="Gompertz parameters c")
```

Note that the b_t parameters (representing Gompertz mortality at age 0) were multiplied by 10^5 for readability, while for the c_t parameters giving the increase from one age to the next, we look at the percentage increase $100(c_t - 1)$.

Also note that the parameters were estimated on the subset with ages 30 and over. The Gompertz model should only be used for these ages. Main effect `as.numeric(x)` and constant `1` were deleted from the model formula in order to facilitate extraction of the parameters from the `glm`-object, as usual.

To forecast future mortality in the Lee-Carter model, one might extrapolate the κ_t vector using the standard time series techniques available. In the resulting plot below, we see that the pattern of the Gompertz parameters is very regular after, say, year $t = 20$.

Q_{17} Extrapolating the trend in the graph of the b_t parameters after $t = 20$, find out after what time $b_t < 0$ will happen. Discuss the consequences of a negative b_t for the one-year mortality probabilities. □



Q_{18} To visualize the trend in mortality, plot the evolution of the fitted log-mortalities at ages $x = 65, 75, 85$ against time t. □