`affine_mortality`: A Github repository for estimation, analysis and projection of affine mortality models

Francesco Ungolo[1,*], Michael Sherris[2,*], and Yuxin Zhou[2,*]

[*] *People sharing this report*
[1] *Chair of Mathematical Finance, Technical University of Munich, Garching bei München, Germany*
[2] *School of Risk & Actuarial Studies, Australian Research Council Centre of Excellence in Population Ageing Research (CEPAR), UNSW Sydney, Kensington, Australia*

August 12, 2021

This tutorial provides guidance in the use of the R files within the Github repository `affine_mortality` containing the code used to fit, examine and compare continuous time affine mortality models. The final implementation will be available as the R package **AffineMortality**. Throughout this tutorial we provide a discussion of how the code works in order to effectively estimate and project the models, and describe the functions available therein. Some illustration of the code for the analysis of the US dataset is provided.

Keywords: Longevity Risk, Kalman Filter, State-space models

1 Introduction

We address the parameter estimation problem of affine mortality models as introduced in Blackburn & Sherris (2013), Jevtić et al. (2013) and further analysed by Huang et al. (2020) and Ungolo et al. (2021).

Ungolo et al. (2021) and the references therein describe how this estimation task lends itself to several numerical issues due to the need of multiplying and inverting large dimensional matrices, and advocate the use of the maximum likelihood estimator with the univariate Kalman Filter of Koopman & Durbin (2000). Conversely, Jevtić et al. (2013) estimate the parameters of a two-factor model by means of the differential evolution algorithm which seeks to minimize a mean squared error measure based on the survival function. They find this algorithm to work effectively, at the expense of very large computational costs. On the other hand, Xu et al. (2020) find out how the classic Kalman Filter maximum likelihood for some parameters can reduce the computational costs and yield a reasonable fit.

The contribution of this tutorial is to provide the user with a toolkit which can be used to carry out an affine modelling based mortality analysis by using the R code available from the Github repository `https://github.com/ungolof/affine_mortality.git`. The available code implements the univariate Kalman Filter procedure to estimate and analyse the affine mortality models as in Ungolo et al. (2021).

To date, the code allows to fit the Blackburn-Sherris model with independent factors, the Blackburn-Sherris model with two and three dependent factors, the Arbitrage-Free Nelson-Siegel model with independent and dependent factors and the Cox-Ingersoll-Ross model. The plan is to further expand the library of models available for fit, such as the same models available so far with cohort or period specific factors, as well as other different models. Other extensions will encompass different formulations of the error term structure.

To the best of our knowledge, only the package StMoMo (Villegas et al. (2018)) is available for fitting mortality rates. However, they only focus on discrete time mortality models, such as Lee-Carter (Lee & Carter (1992)), CBD (Blake et al. (2006)) and the age-period-cohort model by Renshaw & Haberman (2006).

The paper develops as follows: Section 2 introduces the notation needed to describe the remainder of the paper, Section 3 describes the initial set up to prepare the R working environment and Section 4 describes the functions which can be called to estimate the parameters. Section 5 describes the functions which can be used to analyse and compare the affine models, and Section 6 contains a description of the functions which can be called to project future cohort survival curves. We provide two methods to analyse parameter uncertainty: the first estimates the covariance of the parameter estimates by using the bootstrap, while the other implements a multiple imputation based method (Section 7). Section 8 concludes.

2 Notation

Let $\mu_{x,t}$ denote the force of mortality for an individual born in calendar year t and aged x. This is approximated as the central death rate $m_{x,t}$, empirically estimated as the ratio between the observed number of deaths $d_{x,t}$ and the central exposure at risk years $E^c_{x,t}$.

Let $\bar{\mu}_{t,i}$ denote the average force of mortality for an individual born in calendar year

t and aged $x = \ell + i$, defined as follows[1]:

$$\bar{\mu}_{t,i} = \frac{1}{i}\sum_{j=1}^{i} \mu_{\ell+j-1,t} \approx \frac{1}{i}\sum_{j=1}^{i} m_{\ell+j-1,t} \tag{2.1}$$

where ℓ denotes the smallest age in the age-range of interest (equal to 50 in Ungolo et al. (2021)). Conversely, given the average force of mortality, we can obtain μ, by means of the following recursion, starting from $\mu_{\ell,t} = \bar{\mu}_{t,1}$:

$$\mu_{\ell+i,t} = i\bar{\mu}_{t,i} - (i-1)\,\bar{\mu}_{t,i-1} \tag{2.2}$$

for $i = 2, \ldots, N$.

Furthermore, let $\bar{\mu}_t = [\bar{\mu}(t, t+1), \ldots, \bar{\mu}(t, t+N)]'$, where $\bar{\mu}(t, t+k)$ is the average force of mortality between age ℓ and age $\ell + k$ for a population born in year t. N is the number of ages in the age-range of interest.

Due to the affine structure of the average force of mortality, we have $\bar{\mu}_t = A + BX(t)$, where the time-independent A and B depend on the stochastic differential equation which drives dynamics of the latent random vector $X(t)$, and specifies the mortality model under analysis.

3 Set up

The file `Example.R` will show an example of how to utilize the function described in this tutorial, together with some plotting examples.

The function `source` allows to load the files which are needed within the workflow:

```
source('Est_fun.R')
source('Full_opt.R')
source('Coordinate_Ascent.R')
source('GoodnessofFit.R')
source('FilterSmoother.R')
source('StErr_Bootstrap.R')
source('StErr_MultImp.R')
source('Projection.R')
```

Some functions will be `model` specific: for example, when describing the function for projecting the cohort survival curves (Section 6), in this paper we write `S_t_model_proj()`, where `model` can be:

- `BSi` for the Blackburn-Sherris model with independent factors, regardless the number of factors (e.g. `S_t_BSi_proj()`);
- `BSd_2F` and `BSd_3F` for the Blackburn-Sherris model with two and three dependent factors respectively (e.g. `S_t_BSd_2F_proj()`);

[1] We reversed the subscript with respect to μ for notational convenience.

- `AFNSi` for the Arbitrage-Free Nelson-Siegel model with independent level (L), slope (S) and curvature (C);
- `AFNSd` for the Arbitrage-Free Nelson-Siegel model with dependent level, slope and curvature;
- `CIR` for the Cox-Ingersoll-Ross model, regardless the number of factors.

3.1 Loading the data

We use the package `HMDHFDplus` (Riffe (2015)) to directly load the data in the R enviroment. For example, for the US dataset we can load the number of deaths and of the central exposures at risk for each age and calendar year:

```
deaths <- readHMDweb(CNTRY = "USA", item = "Deaths_1x1", username,
password, fixup = TRUE)
exposures <- readHMDweb(CNTRY = "USA", item = "Exposures_1x1",
username, password, fixup = TRUE)
```

Once the data are loaded in the R environment, then we produce the mortality dataset by age (N rows) and cohort (T columns). This code is illustrated in the file `Example.R`.

4 Parameter estimation

As described in Ungolo et al. (2021) we carried out the parameter estimation process using both the optimization over all parameter simultaneously, as well as by coordinate ascent by groups of parameters. After the optimization process, we seek to improve the quality of the fit by local search. The functions needed to estimate the parameters are illustrated below by type of optimization.

4.1 Full optimization (function `it_f_opt_model()`, file 'Full_opt.R')

The full optimization process is carried out as follows: after setting the starting values, the log-likelihood function is iteratively optimized over all parameters simultaneously, and at each iteration we use as starting values the parameter estimates obtained at the previous step. We note that the optimizer continues to search for further optimum.

For the Blackburn-Sherris with independent factors and the CIR models, the function recognizes the number of factors by looking at the length of the parameter vector κ.

In case the starting values are not specified, the code will run with its default values (based on the parameter estimates for the Danish dataset in Ungolo et al. (2021)). In any case, we would like to warn the user from utilizing the default starting values, as these may produce an error for the particular dataset on hand.

As an illustration, for the Blackburn-Sherris model with three independent factors (Blackburn & Sherris (2013)), we use the function `it_f_opt_BSi()`, which has the following arguments:

- `mu_bar`: $N \times T$ dataset of average force of mortality;

- `x0`: starting value for the initial state, $X(0)$;
- `delta`: starting value for the parameter vector $\delta = (\delta_{11}, \delta_{22}, \delta_{33})$;
- `kappa`: starting value for the parameter vector $\kappa = (\kappa_1, \kappa_2, \kappa_3)$;
- `sigma`: starting value for the parameter vector $\sigma = (\sigma_{11}, \sigma_{22}, \sigma_{33})$;
- `r`: vector of starting values for r_1, r_2 and r_c;
- `max_iter`: maximum number of times the parameter estimation process should be repeated (default 10);
- `tol_lik`: the estimation process is stopped if the log-likelihood increase between iterations is lower than `tol_lik` (default 10);
- `opt_met`: the optimization method within the R built-in routine `optim` (default Nelder-Mead).

For example, `it_f_opt_BSi(mu_bar=mu_bar_USA, max_iter=3, tol_lik=10)` produces the following output:

```
$par_est
$par_est$x0
[1]  0.05878113 -0.07851862  0.03341285

$par_est$delta
[1] -0.002326806 -0.020335907 -0.066058875

$par_est$kappa
[1] 0.038615416 0.030284845 0.006777906

$par_est$sigma
[1] 0.0040856819 0.0074436718 0.0005671597

$par_est$r1
[1] 4.236575e-16

$par_est$r2
[1] 0.5913345

$par_est$rc
[1] 9.048733e-08

$log_lik
[1] 10600.5
```

In addition, the function `it_f_opt_BSi` returns also a table (not shown here) containing the value of the parameters, the value of the log-likelihood function and the convergence code from the `optim` function at each iteration.

For models with dependent factors, such as the Blackburn-Sherris analysed by Huang et al. (2020) and Ungolo et al. (2021), we directly code the model with two and three factors. The function `it_f_opt_BSd_3F()` has the following arguments:

- `mu_bar`;
- `x0`;
- `delta`: starting value for the parameter vector $\delta = (\delta_{11}, \delta_{21}, \delta_{22}, \delta_{31}, \delta_{32}, \delta_{33})$;
- `kappa`;
- `sigma_dg`: starting value for the parameter vector $\sigma = (\sigma_{11}, \sigma_{22}, \sigma_{33})$ (that is the standard deviations);
- `Sigma_cov`: starting value for the covariances σ_{21}, σ_{31} and σ_{32} (in this order);
- `r`;
- `max_iter`;
- `tol_lik`;
- `opt_met`;

The parameter estimation for the two AFNS and the CIR models follows along the same lines.

4.2 Coordinate Ascent (function `co_asc_model()`, file 'Coordinate_ascent.R')

This method optimizes the log-likelihood function by groups of parameters iteratively until convergence. For illustration, the AFNS model with independent factors can be estimated by using the function `co_asc_AFNSi()` with the following arguments:

- `mu_bar`;
- `x0`;
- `delta`: starting value for the parameter δ;
- `kappa`: starting value for the parameter vector $\kappa = (\kappa_L, \kappa_S, \kappa_C)$;
- `sigma`: starting value for the parameter vector $\sigma = (\sigma_L, \sigma_S, \sigma_C)$;
- `r`: vector of starting values for r_1, r_2 and r_c;
- `max_iter`: maximum number of times the log-likelihood function is iteratively optimized by groups of parameters until convergence (default 200);
- `tol_lik`: the estimation process is stopped if the log-likelihood increase is lower than `tol_lik` compared to the previous iteration (default 0.1).

In this case, the gradient-free Nelder-Mead optimization method is always used. The function `co_asc_model()` yields a similar output compared to the function `it_f_opt_model()`.

4.3 Local search (function `f_opt_model`, file Full_opt.R)

In a further attempt to optimize the log-likelihood function, we use the `BFGS` gradient-based optimization method by means of the functions `f_opt_BSi_LS()`, `f_opt_BSd_3F_LS()`, `f_opt_AFNSi_LS()`, `f_opt_AFNSd_LS()` and `f_opt_CIR_LS()`, depending on the fitted model. These functions take as arguments matrix `mu_bar` and the starting values for the parameters of the models. Also in this case, default starting values are provided.

For example, for the AFNS model with dependent factors, we have:

```
> f_opt_AFNSd_LS(mu_bar=mu_bar_USA)
$par_est
$par_est$x0
[1]  0.009569488  0.010913521 -0.001464853

$par_est$delta
[1] -0.07486799

$par_est$kappa
[1] 0.013893762 0.003525892 0.003004961

$par_est$Sigma
$par_est$Sigma$sigma_L
[1] 0.003215422

$par_est$Sigma$sigma_LS
[1] -8.670149e-06

$par_est$Sigma$sigma_S
[1] 0.002730213

$par_est$Sigma$sigma_LC
[1] -2.667579e-06

$par_est$Sigma$sigma_SC
[1] 2.272511e-06

$par_est$Sigma$sigma_C
[1] 0.0008445408


$par_est$r1
[1] 3.259345e-15

$par_est$r2
[1] 0.5451931

$par_est$rc
[1] 1.817543e-07

$log_lik
```

```
[1] 10367.95
```

For the CIR model, this method should be used with care, as in our experience it turned out to return an error in few cases.

In general, models with large number of parameters, such as the Blackburn-Sherris model with dependent factors and the CIR model are estimated more robustly by means of the Coordinate Ascent algorithm.

5 Goodness-of-fit (file `GoodnessofFit.R`)

The file `GoodnessofFit.R` contains the code with the functions which return the fitted rates (`mu_bar_hat_model()`, where `model` can be `BSi`, `BSd_2F` and so on) having `mu_bar` and the parameters as argument, the value of the AIC and the BIC information criteria (`AIC_BIC()`), the Root Mean Squared Error (`RMSE()`), and the mean absolute percentage error by age (`MAPE_row()`). The same file contains also the code to compute the residuals (`residuals_f()`), the 0-1 residuals which distinguish between negative and positive residuals (`residuals_01()`), the standardized residuals for each model (`residuals_std_model()`) and the Poisson Standardized Residuals (`poi_r_std()`).

5.1 Fitted rates

The function `mu_bar_hat_model()` takes `mu_bar` and the parameters (estimates in this case) as arguments and yields a matrix of fitted average mortality rates where the rows label the ages, and the columns indicate the cohort year. For the tth cohort we have:

$$\widehat{\bar{\mu}}_t = A + B\widehat{X}(t) \tag{5.1}$$

where the time-invarying yield adjustment term A and the factor loading B depend on the parameter estimates. The state vector $\widehat{X}(t)$ is estimated by using the Kalman Filter, and it also depend on `mu_bar` and the parameter estimates.

The function `avg2rates(mu_bar)` takes as argument the matrix of the average force of mortality $\bar{\mu}$ and yields a matrix of death rates μ, by using equation (2.2). In this way, we can obtain a matrix of fitted death rates.

5.2 Information Criteria

The function `AIC_BIC(log_likelihood, n_par, n_obs)` has three arguments, namely the value of the log-likelihood function, the number of model parameters, and the number of observations, this latter given by *number of ages*$\times$*number of cohort year*. It returns the value of the Akaike Information Criterion, defined as AIC$= -2$`log_likelihood`$+2$`n_par` and of the BIC, defined as BIC$= -2$`log_likelihood`$+$`n_par`$\times \log($`n_obs`$)$.

For example, for the Blackburn-Sherris model fitted to the USA dataset, we have[2]:

[2]These results are different from those shown in Ungolo et al. (2021), because they first fully optimize the models, then they improve the estimates by local search, and repeat the same process using coordinate ascent before obtaining the final parameter estimates.

```
fit_BSi_3F <- it_f_opt_BSi(n_iter=3, tol_lik=10)
> AIC_BIC(fit_BSi_3F$log_lik, 12, nrow(mu_bar)*ncol(mu_bar))
$AIC
[1] -21177

$BIC
[1] -21112.1
```

5.3 Root Mean Squared Error

The function `RMSE(observed, estimated)` takes as arguments the observed matrix of the average force of mortality (as approximated in Section 2) and of the estimated average of fitted rates, returning the value of the Root Mean Squared Error, calculated as follows:

$$\text{RMSE} = \frac{1}{TN} \sum_x \sum_t \left(\bar{\mu}_{t,x} - \widehat{\bar{\mu}}_{t,x}\right)^2 \tag{5.2}$$

It is also possible to supply the value of the mortality rates in lieu of the average rates.

5.4 Mean Absolute Percentage Error (by age)

The function `MAPE_row(observed, estimated)` has the same argument as `RMSE()` and yields a vector of the Mean Absolute Percentage Error calculated for each age taken into consideration. A plotting example is available from the file `Example.R`

5.5 Residuals

The function `residuals_f(observed, estimated)` yields a matrix of the difference between the empirical average force of mortality (or the force of mortality itself) and its estimated counterpart. Similarly, the function `residuals_01(observed, estimated)` yields a matrix with cells equal to zero if the residuals are negative and equal to 1 otherwise.

The function `residuals_std_model(observed, par1, par2, ..., parK)` takes as arguments the matrix of the observed average mortality rates[3] (`observed`) and of the parameters of the `model` under analysis (`par1`, ..., `parK` by groups of parameters). The standardized residuals here denoted by the vector r_t for the tth cohort are computed as follows:

$$r_t = \left(\sqrt{\widehat{\mathbb{V}\left(\bar{\mu}_t\right)}}\right)^{-1} \left(\bar{\mu}_t - \widehat{\bar{\mu}}_t\right) \tag{5.3}$$

where $\sqrt{\widehat{\mathbb{V}\left(\bar{\mu}_t\right)}}$ depends on the parameter estimates.

The Poisson standardized residuals can be obtained by means of the function `poi_r_std(mu_bar_hat, deaths, exposures)` takes as arguments the matrix of fitted average mortality rates,

[3]The matrix for the force of mortality will be created in a second moment

the number of deaths and the central exposure at risk years (all matrices with same dimension) and yields the matrix of Poisson residuals $r^P_{x,t}$, calculated as:

$$r^P_{x,t} = \frac{d_{x,t} - \widehat{d}_{x,t}}{\sqrt{\widehat{d}_{x,t}}} \tag{5.4}$$

where the fitted number of deaths $\widehat{d}_{x,t}$ is obtained as $\widehat{d}_{x,t} = E^c_{x,t}\widehat{\mu}_{x,t}$.

Heatmap plotting examples are shown in the file `Example.R`

6 Projection

Once we have the fitted rates for the cohorts born between year Y_1 and year Y_2, we can obtain the projected cohort survival curves for cohorts born h years after Y_2 by using the function `S_t_model_proj(par1, par2, ..., parK, proj_years)` which takes as arguments the parameter of the `model` and the value of h (`proj_years`). The output of this function is a vector with the value of $S(Y_2 + h)$ for each age in the age-range of interest.

7 Parameter uncertainty

A further source of risk when projecting cohort survival curves is the uncertainty inherited from the parameter estimation process. In this Section we present and implement two methods to estimate the standard errors, namely the bootstrap, introduced by Stoffer & Wall (2009) and used in Blackburn & Sherris (2013) (7.1) and the multiple imputation method, described in Little & Rubin (2019) (7.2).

7.1 Bootstrap

The file `StErr_Bootstrap.R` contains the code which implements the bootstrap method for estimating the covariance of the parameter estimates.

In this way, we compute the standardized innovations from the measurement error, in order to obtain a bootstrapped dataset of average mortality rates. These are then used to re-estimate the model and obtain a new set of parameter estimates. This procedure allows to obtain a set of `n_BS` parameter estimates, which are used to estimate the covariance.

This can be estimated by means of the function `CovEst_BS_model(par1, par2, ..., parK, mu_bar, n_BS=500, t_ex = 4)`, which takes as arguments the parameter estimates, the matrix of the average force of mortality (`mu_bar`), the number of bootstrap samples (`n_BS`, default 500) and the number of the oldest cohort residuals which should be discarded from the resampling (`t_ex` default 4) due to the Kalman Filter start up irregularities (see Stoffer & Wall (2009) and Blackburn & Sherris (2013)). This function carries out a procedure where the log-likelihood function is optimized with respect all parameters simultaneously by using the `BFGS` optimization method. In this way, using

the parameter estimate as starting value of each optimization procedure, we are better able to explore the likelihood surface in the neighborhood of the parameters.

`CovEst_BS_model()` returns the bootstrap estimate of the covariance of the parameter estimates and the corresponding standard errors.

7.2 Multiple imputation

The multiple imputation procedure consists of randomly imputing a value of the latent state variable $X(t)$ sampled from the smoothing distribution at the value of the parameter estimates. In this way, we obtain a set of "completed" datasets such that parameters are then re-estimated.

This methodology turns out useful, because from one side, it may not be possible to numerically compute the Information matrix coming from the optimization process of the likelihood function due to its very flat surface. On the other hand, the bootstrap method of Section 7.1 may be computationally expensive if carried out hundreds of times.

The downside of this methodology, is that unlike the bootstrap, it may tend to underestimate the standard error, since it is a delta method. From a computational perspective, a potential downside is given by the need to invert a Hessian matrix of larger dimensions, albeit this task is simpler than the case where we invert the Hessian matrix from the estimation procedure (whose likelihood is integrated out the latent states).

In our implementation it turned out that in some cases the resulting covariance matrix may not be positive definite, hence we recommend to check very carefully the resulting output.

8 Conclusion

This paper provides a short tutorial on the use of the code available in the Github repository `affine_mortality` for analysing mortality models by means of affine mortality models, as described in Huang et al. (2020) and Ungolo et al. (2021). These methodologies can be used also for the analysis of age-period mortality datasets, as well as for the analysis of interest rate models.

The authors plan to further expand the range of models which can be fitted, as well as to add specific other features, such as cohort specific factors. Further additions will encompass the possibility to account for incomplete cohort (or period) data, the inclusion of models whose mean-reversion parameter is non zero, and so on.

References

Blackburn, C. & Sherris, M. (2013), 'Consistent dynamic affine mortality models for longevity risk applications', *Insurance: Mathematics and Economics* **53**(1), 64 – 73.
URL: *http://www.sciencedirect.com/science/article/pii/S0167668713000632*

Blake, D., Cairns, A. & Dowd, K. (2006), 'A two-factor model for stochastic mortality with parameter uncertainty: Theory and calibration', *Journal of Risk Insurance* **73**, 687–718.

Huang, Z., Sherris, M., Villegas, A. & Ziveyi, J. (2020), 'The application of affine processes in cohort mortality risk models', **Research paper, UNSW Business School. Available at SSRN 3446924**.

Jevtić, P., Luciano, E. & Vigna, E. (2013), 'Mortality surface by means of continuous time cohort models', *Insurance: Mathematics and Economics* **53**(1), 122–133.
URL: *https://www.sciencedirect.com/science/article/pii/S0167668713000619*

Koopman, S. J. & Durbin, J. (2000), 'Fast filtering and smoothing for multivariate state space models', *Journal of Time Series Analysis* **21**(3), 281–296.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-9892.00186*

Lee, R. D. & Carter, L. R. (1992), 'Modeling and forecasting u. s. mortality', *Journal of the American Statistical Association* **87**(419), 659–671.
URL: *http://www.jstor.org/stable/2290201*

Little, R. J. A. & Rubin, D. B. (2019), *Statistical Analysis with Missing Data, 3rd ed.*, Wiley, USA.

Renshaw, A. & Haberman, S. (2006), 'A cohort-based extension to the lee-carter model for mortality reduction factors', *Insurance: Mathematics and Economics* **38**, 556–570.

Riffe, T. (2015), Hmdhfdplus: Reading human fertility database and human mortality database data into r, Technical Report TR-2015-004, MPIDR.
URL: *http://www.demogr.mpg.de/en/projects_publications/publications_1904/mpidr_technical_reports/re*

Stoffer, D. & Wall, K. (2009), 'Resampling in state space models', *State Space and Unobserved Component Models: Theory and Applications* .

Ungolo, F., Sherris, M. & Zhou, Y. (2021), 'Estimation and projection of affine mortality models: A multi-country comparison', *Working paper* .

Villegas, A. M., Kaishev, V. K. & Millossovich, P. (2018), 'StMoMo: An R package for stochastic mortality modeling', *Journal of Statistical Software* **84**(3), 1–38.

Xu, Y., Sherris, M. & Ziveyi, J. (2020), 'Continuous-time multi-cohort mortality modelling with affine processes', *Scandinavian Actuarial Journal* **2020**(6), 526–552.
URL: *https://doi.org/10.1080/03461238.2019.1696223*