SUPPLEMENT TO EFFICIENT USE OF DATA FOR LSTM MORTALITY FORECASTING

M. LINDHOLM AND L. PALMBORG

1. Introduction

This document contains technical details regarding the calibration procedures and implementation used in "Efficient use of data for LSTM mortality forecasting", [5]. Furthermore, it contains detailed numerical analyses and additional comparisons, to supplement the numerical illustrations in the main text. For notation and detailed explanations of figures we refer to the main text.

2. Calibration procedures

We structure data according to

$$\boldsymbol{K} = \begin{bmatrix} \widehat{\kappa}_1 & \widehat{\kappa}_2 & \dots & \widehat{\kappa}_p & \widehat{\kappa}_{p+1} \\ \widehat{\kappa}_2 & \widehat{\kappa}_3 & \dots & \widehat{\kappa}_{p+1} & \widehat{\kappa}_{p+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \widehat{\kappa}_{n-p} & \widehat{\kappa}_{n-p+1} & \dots & \widehat{\kappa}_{n-1} & \widehat{\kappa}_n \end{bmatrix}$$

where the first p columns correspond to the input sequences and the last column corresponds to the output that should be predicted by the model.

In Table 1 and 2 below we illustrate the three calibration procedures using a simple example, where $(\widehat{\kappa}_{t-1}, \widehat{\kappa}_t)$ are used to predict $\widehat{\kappa}_{t+1}$. Hence matrix $\boldsymbol{K}$ has three columns. Table 1 shows $\boldsymbol{K}$ for Calibration LO (left) and Calibration RT (right). The training set consists of the green coloured rows, and the validation set consists of the blue coloured rows. The test set in this simple example consists of four rows of data, coloured gray. Table 2 shows the training and validation set for Calibration SP, which is somewhat different compared two the first two approaches. Here we instead have two matrices: $\boldsymbol{K}^{(1)}$ containing $\widehat{\kappa}_t$s estimated for the first subpopulation, and $\boldsymbol{K}^{(2)}$ containing $\widehat{\kappa}_t$s estimated for the second subpopulation. Hence, all rows of matrix $\boldsymbol{K}^{(1)}$ on the left make up the training set and are thus coloured green, while all rows of matrix $\boldsymbol{K}^{(2)}$ on the right make up the validation set and are thus coloured blue. The test set is the same as for Calibration LO and RT, hence consisting of four rows of data containing $\widehat{\kappa}_t$s estimated for the total population.

Date: November 10, 2021.

Department of Mathematics, Stockholm University, Box 106 91, Stockholm, Sweden.

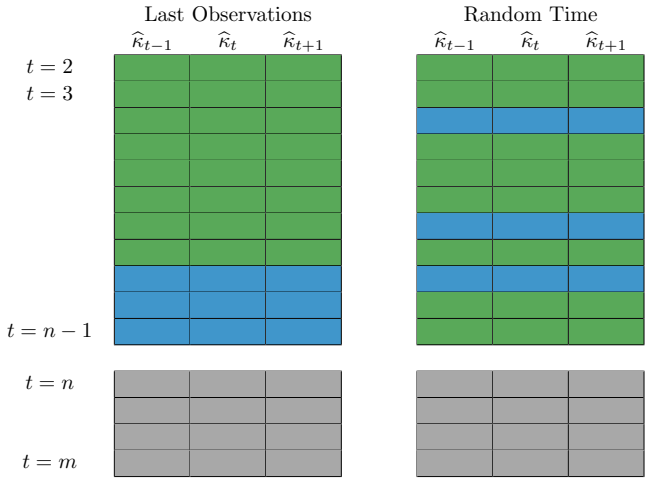

TABLE 1. Training and validation data, for Calibration LO (left) and Calibration RT (right).

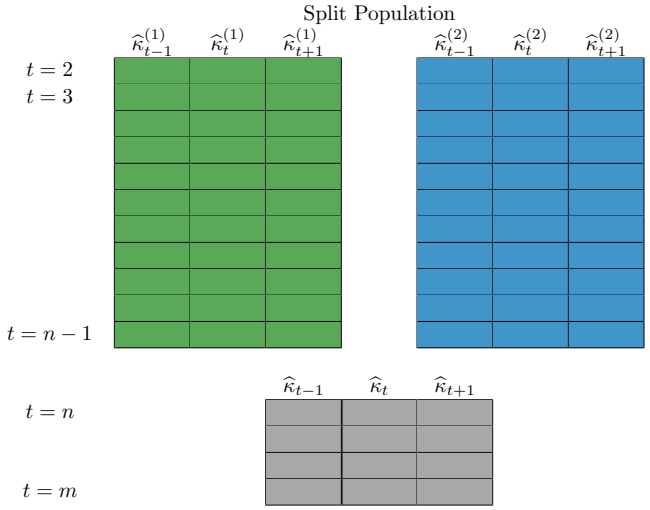

TABLE 2. Training and validation data, for Calibration SP.

2.1. **Calibration SP for aggregate level data.** For Calibration SP, the idea is to create validation data by sampling individuals and randomly assigning them to subpopulations. This is done by first creating a bootstrapped population of the same size as the original population (or smaller size if using subsampling), and then randomly splitting this bootstrapped population into two subpopulations. However, this is only possible if one has access to individual level data, which is not the case in the numerical illustrations in Section 5 in the main text [5], where we rely on aggregate data from the HMD [4]. To solve this, we create synthetic subpopulations in the following way: Let the aggregate data be given

by $\{(r_{x,t}, d_{x,t}), x \in \mathcal{X}, t \in \mathcal{I}\}$, where $d_{x,t}$ is the total number of deaths and $r_{x,t}$ is the total exposure-to-risk for individuals being x years old during calendar year t. Let $n_{x,t}$ denote the total number of individuals in the population consisting of x year old individuals at the start of calendar year t. We use the approximation $n_{x,t} \approx r_{x,t} + \frac{1}{2} d_{x,t}$ to convert from $r_{x,t}$ to $n_{x,t}$. The number of individuals at the start of each calendar year in the aggregate population changes due to deaths, but also due to e.g. immigration/emigration. Hence we define the net immigration $\delta_{x,t}$ of x years old individuals during calendar year t as

$$\delta_{x,t} := n_{x+1,t+1} - (n_{x,t} - d_{x,t}).$$

Furthermore, the one-year death probability $q_{x,t}$ is estimated by

$$\widehat{q}_{x,t} = \frac{d_{x,t}}{n_{x,t}}.$$

Let $N_{x,t}^{(i)}$ denote the number of individuals who are x years old at the start of calendar year t, belonging to synthetic subpopulation i, for $i = 1, 2$; let $D_{x,t}^{(i)}$ denote the corresponding number of deaths; and let $\Delta_{x,t}^{(i)}$ denote the corresponding net immigration, during calendar year t. Similarly, let $N_{x,t}^{b}$, $D_{x,t}^{b}$, and $\Delta_{x,t}^{b}$, denote the corresponding number of individuals, number of deaths, and net immigration in the bootstrapped population. Let $t_{\min} := \min \mathcal{I}$, $t_{\max} := \max \mathcal{I}$, and $x_{\max} := \max \mathcal{X}$. Further, let π denote the fraction of the number of individuals in each age group that is assigned to subpopulation 1, and let ξ denote the subsampling parameter ($\xi = 1$ if subsampling is not used). We simulate $N_{x,t}^{(i)}$, $D_{x,t}^{(i)}$, and $\Delta_{x,t}^{(i)}$ according to Algorithm 1. Note that quantities that need to be integer valued should be rounded to the nearest integer.

Remark 1 (Algorithm 1)**.**

(a) Individuals born during or before calendar year $t_{\min}$ are bootstrapped or assigned to a subpopulation at that time point according to π and ξ. Individuals born after $t_{\min}$ are bootstrapped or assigned to a subpopulation at the time point corresponding to their year of birth.

(b) The number of deaths per calendar year and age in the bootstrapped population are simulated according to conditional binomial probabilities. Hence the number of deaths assigned to each subpopulation follow a hypergeometric distribution, i.e.

$$P(D_{x,t}^{(1)} = k) = \frac{\binom{N_{x,t}^{(1)}}{k}\binom{N_{x,t}^{b} - N_{x,t}^{(1)}}{D_{x,t}^{b} - k}}{\binom{N_{x,t}^{b}}{D_{x,t}^{b}}}.$$

Moreover, note that since the total sampled population is constructed using a parametric bootstrap using binomial probabilities, this can be thought of as sampling individuals from an infinite population, since no individuals can be sampled more than once. If one would have access to complete life histories and these were sampled with replacement, an individual might be sampled more than once, particularly when ξ is close to one. As an alternative to Algorithm 1 one could of course first create a synthetic *labelled* total population, satisfying

Algorithm 1 Creating synthetic subpopulations

Algorithm parameters: $t_{\min}$, $t_{\max}$, $x_{\max}$, π, ξ, and $n_{x,t}, \widehat{q}_{x,t}$ for $x \in \mathcal{X}$, $t \in \mathcal{I}$

(i) Creating a bootstrapped population
Initialise $N^b_{x,t_{\min}} = \xi n_{x,t_{\min}}$, for $x \in \mathcal{X}$
for $t = t_{\min}, t_{\min}+1, \ldots, t_{\max}$ **do**
  if $t < t_{\max}$ **then**
    $N^b_{0,t+1} = \xi n_{0,t+1}$
  end if
  for all $x \in \mathcal{X}$ **do**
    $D^b_{x,t} \sim \mathsf{Bin}(N^b_{x,t}, \widehat{q}_{x,t})$
    $\Delta^b_{x,t} = \frac{N^b_{x,t} - D^b_{x,t}}{n_{x,t} - d_{x,t}} \delta_{x,t}$
    $N^b_{x+1,t+1} = N^b_{x,t} - D^b_{x,t} + \Delta^b_{x,t}$
  end for
end for

(ii) Splitting the bootstrapped population into two subpopulations
Initialise $N^{(1)}_{x,t_{\min}} = \pi N^b_{x,t_{\min}}$ and $N^{(2)}_{x,t_{\min}} = N^b_{x,t_{\min}} - N^{(1)}_{x,t_{\min}}$, for $x \in \mathcal{X}$
for $t = t_{\min}, t_{\min}+1, \ldots, t_{\max}$ **do**
  if $t < t_{\max}$ **then**
    $N^{(1)}_{0,t+1} = \pi N^b_{0,t+1}$
    $N^{(2)}_{0,t+1} = N^b_{0,t+1} - N^{(1)}_{0,t+1}$
  end if
  for all $x \in \mathcal{X}$ **do**
    $D^{(1)}_{x,t} \sim \mathsf{Hyp}(N^b_{x,t}, D^b_{x,t}, N^{(1)}_{x,t})$
    $D^{(2)}_{x,t} = D^b_{x,t} - D^{(1)}_{x,t}$
    $\Delta^{(1)}_{x,t} = \frac{N^{(1)}_{x,t} - D^{(1)}_{x,t}}{N^b_{x,t} - D^b_{x,t}} \Delta^b_{x,t}$
    $\Delta^{(2)}_{x,t} = \Delta^b_{x,t} - \Delta^{(1)}_{x,t}$
    $N^{(1)}_{x+1,t+1} = N^{(1)}_{x,t} - D^{(1)}_{x,t} + \Delta^{(1)}_{x,t}$
    $N^{(2)}_{x+1,t+1} = N^{(2)}_{x,t} - D^{(2)}_{x,t} + \Delta^{(2)}_{x,t}$
  end for
end for

the marginal characteristics w.r.t. number of deaths in particular ages and calendar time periods of the observed population, and then sample from this synthetic population, but his would become very computationally intensive.

(c) We only have access to net immigration and not the actual number of arrivals to and departures from the population. Since we do not have any view on an appropriate model for net immigration, we have chosen to simply distribute the net immigration for each age group and calendar year according to the relative population size after deaths of each subpopulation. Similarly, for the bootstrapped population, we add net immigration according to the population

size after deaths of the bootstrapped population relative to the population size after deaths in the original population.

(d) Note that if you only have access to aggregated death count and survivor data, given that these observations are from birth cohorts that consist of approximately i.i.d. individuals, then Algorithm 1(ii) can be used to allocate these individuals into two subgroups by replacing all bootstrapped values from Algorithm 1(i) with true observations. Further, note that if you are using subsampling with a sufficiently small ξ, an alternative could be to sample *without* replacement, which would correspond to using Algorithm 1(ii) twice: First using ξ, and then in a second step using π and replacing all bootstrapped values from Algorithm 1(i) with the ξ-sampled values.

(e) Algorithm 1 describes how a population can be bootstrapped and split into groups that have fixed birth cohort relations in all sampled populations compared with the original population. One could also consider to use random weights per birth cohort in the training and validation data set, but also in the bootstrapped overall population compared with the original population.

3. Comments on architecture and implementation

There are many different hyperparameters used to define the LSTM architecture. For all other hyperparameters than those discussed below, the default values of `R` package `keras` are being used and we refer to [1] for more details. However, it is important to note that we have not tried to optimise performance by tuning these parameters. Instead, our focus has been to choose parameters that work reasonably well for all populations that are being considered, in order to illustrate the performance of the different calibration methods. It is possible that other choices of these parameters could improve the performance of the model, and is something that should be investigated further before using these models in practice.

3.1. **Lags.** Lag 5 is used for all methods and time periods. In [6] lag 1 is used, but this choice of lag does not fully exploit the properties of an LSTM model, as discussed in Section 2 in the main text [5]. Furthermore, we have compared results with lag 1 and 5, and this limited analyses indicate that a lower MSE on the validation set is obtained when using lag 5. This also holds true on the test set. One can also mention that others consider different lags, see e.g. [8, 7].

3.2. **Activation function.** As a starting point the ReLu activation function is used, which is in accordance with [6]. This seems to work reasonably well on "raw", unscaled data. This choice will, however, turn out to be problematic for long-term predictions. Due to this, we also consider the tanh activation, which turn out to work well given that data is scaled / pre-processed.

3.3. **Number of neurons.** We set the number of neurons to equal the number of observations used for the initial parameter estimation in the Poisson Lee-Carter model. This gives us a simple rule for determining the number of neurons when varying the time-window for training data. The heuristic underlying this simple rule, is that when having fewer data points it will become increasingly harder

to avoid overfitting. The choice of 50 neurons for the longer time-window and 20 neurons for the shorter time-window is based on this simple rule, hence it is possible that other number of neurons might achieve better performance.

3.4. **Number of hidden layers.** Our analysis is restricted to a shallow model with one hidden LSTM layer. We have done some experimenting with two layers, without seeing any improved performance.

3.5. **Batch size.** In order to use SGD, a batch size smaller than the number of observations in the training set is needed. We have simply set the batch size equal to one. This is based on that we have a very limited number of observations in the training set, hence the batch size needs to be small for it to have any effect. Another option would be to set the batch size to the number of observations in the training set, in this way using standard gradient descent with random weight initialisation. Since we achieved a lower MSE on the validation set averaged over the model calibrations in each ensemble with batch size one, we decided to focus on this batch size without investigating if there are other batch sizes between one and the number of observations in the training set that produces a lower validation error.

3.6. **Number of model calibrations in each ensemble.** The number of calibrations used in each ensemble is a parameter that needs to be decided upon. By including more calibrations one expects that the predictive performance should stabilise, but at the cost of having a computationally more expensive model. As with the other hyperparameters we want to use a number of calibrations that work reasonably for all populations and calibration methods. Not surprisingly the effect of including more calibrations into the ensemble will diminish, and we have settled on using 20 calibrations in each ensemble. This choice may be suboptimal in some cases, but in general works well for most populations and calibration methods. In practice, this is a parameter that could be tuned as well. For an illustration of model performance as a function of the number of model calibrations included in each ensemble, see Figure 34, which shows the out-of-sample MSE for an ensemble consisting of $5, 10, \ldots, 80$ calibrations for Italian and Swedish males. In a non-life insurance setting, [9] saw stability after about 20 calibrations. Within the context of mortality modelling, [7] settled on 10 calibrations for their ensemble model.

3.7. **Data pre-processing.** As described in Section 3.2, in some examples we use data pre-processing. One form of data pre-processing is to use boosting, which attempts to remove the trend from data. When this procedure is used, we also pre-process data by using min-max-scaling, thus scaling data so that all values are in the range $[-1, 1]$. Transformations of this type is generally recommended to improve the training of neural networks, see e.g. [3, Ch. 11.5.3]. However, we consider this form of scaling of data as inappropriate to use unless the trend has been removed from data first, since otherwise the predicted values will tend to consistently lie outside of the interval $[-1, 1]$.

3.8. **Subsampling.** For Calibration SP it is not recommended to merely split the population in two, if the population size is too large. For the Swedish population, which is approximately 4-5 million females and males over the analysed time period, our analyses indicate that it works reasonably well to split the population in two parts, where 80 % is used for training and 20 % is used for validation. Due to this, when using Calibration SP for the other countries, subsampling producing an effective sample of size approximately 4-5 million is used. This motivates that we have used a 20 % subsampling for Italy and a 5 % subsampling for the USA.

Concerning optimisation routines, we use the Nesterov Adam optimiser, i.e. stochastic gradient descent with Nesterov momentum, see [2, Ch. 8.3] for an overview, and see [1] for the implementation in the `R` package `keras`. The method of early stopping is used throughout to prevent overfitting. We have not investigated the use of other regularisation methods, e.g. dropout, see e.g. [2, Ch. 7.12], hence it is possible that the network performance could be improved further.

4. Base case, 1950–1999, "raw data"

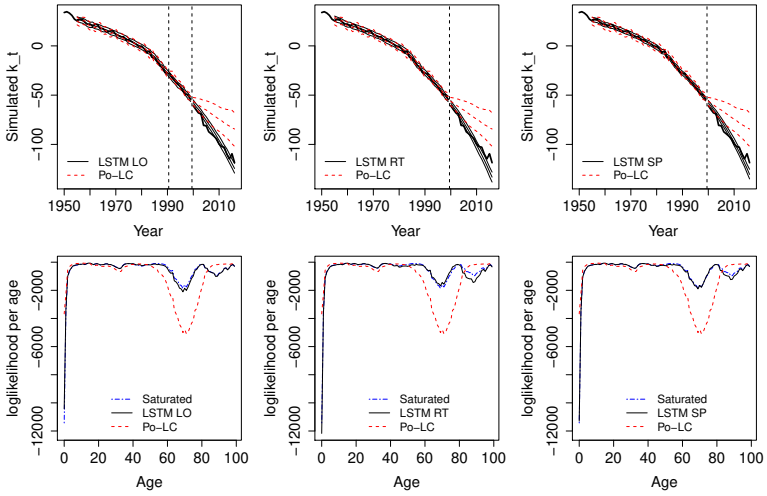

Figure 1. Italian males, LSTM ensembles trained on "raw data" with activation function ReLu: Top: κ_t in-sample and out-of-sample. Bottom: log-likelihood per age, out-of-sample.

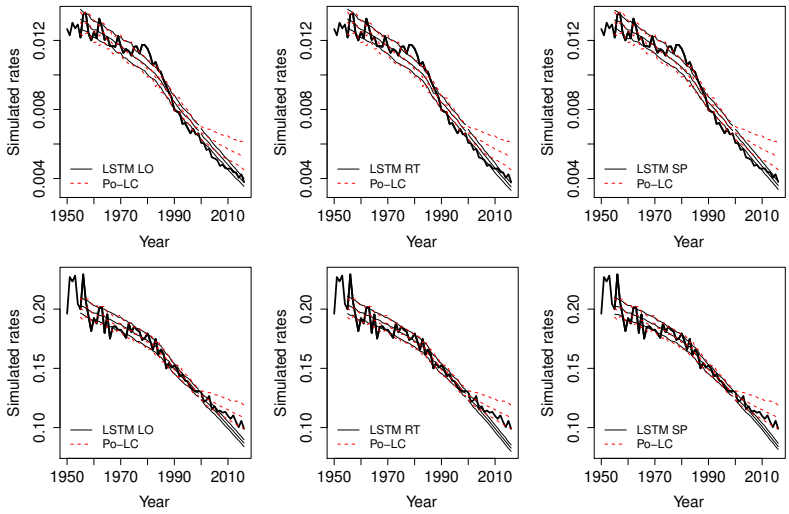

FIGURE 2. Italian males, LSTM ensembles trained on "raw data" with activation function ReLu: Observed mortality rates 1950–2016, and predicted simulated mortality rates. Top: age 55. Bottom: age 85.

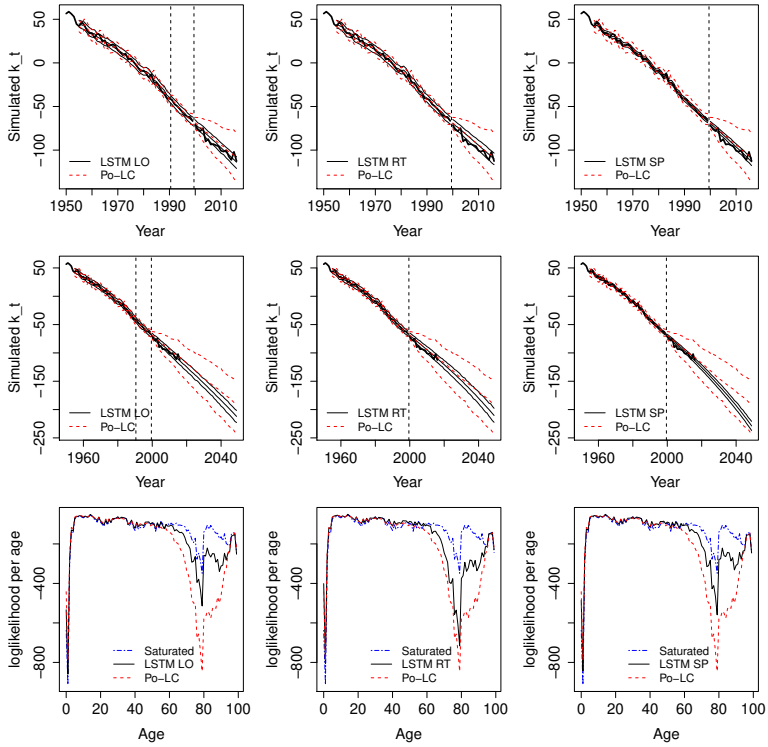

FIGURE 3. Italian females, LSTM ensembles trained on "raw data" with activation function ReLu: Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

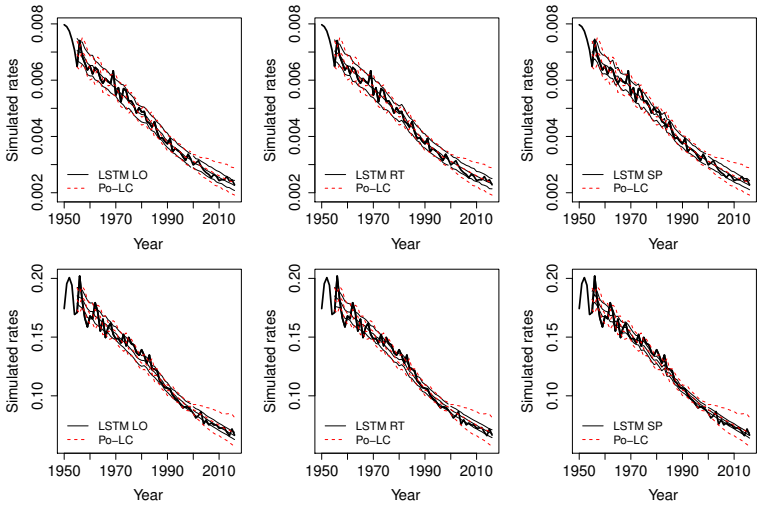

FIGURE 4. Italian females, LSTM ensembles trained on "raw data" with activation function ReLu: Top: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55. Bottom: age 85.

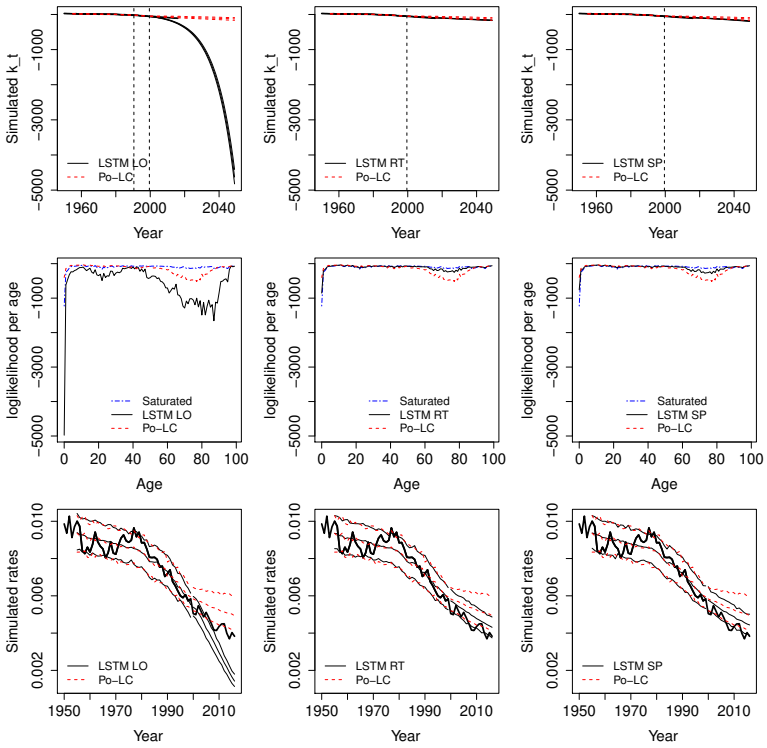

FIGURE 5. Swedish males, LSTM ensembles trained on "raw data" with activation function ReLu: Top: κ_t long-term prediction. Middle: log-likelihood per age, out-of-sample. Bottom: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55.

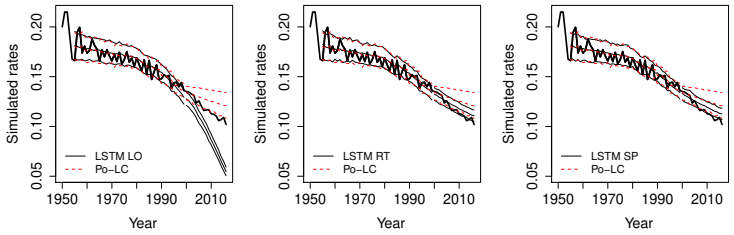

FIGURE 6. Swedish males, LSTM ensembles trained on "raw data" with activation function ReLu: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 85.

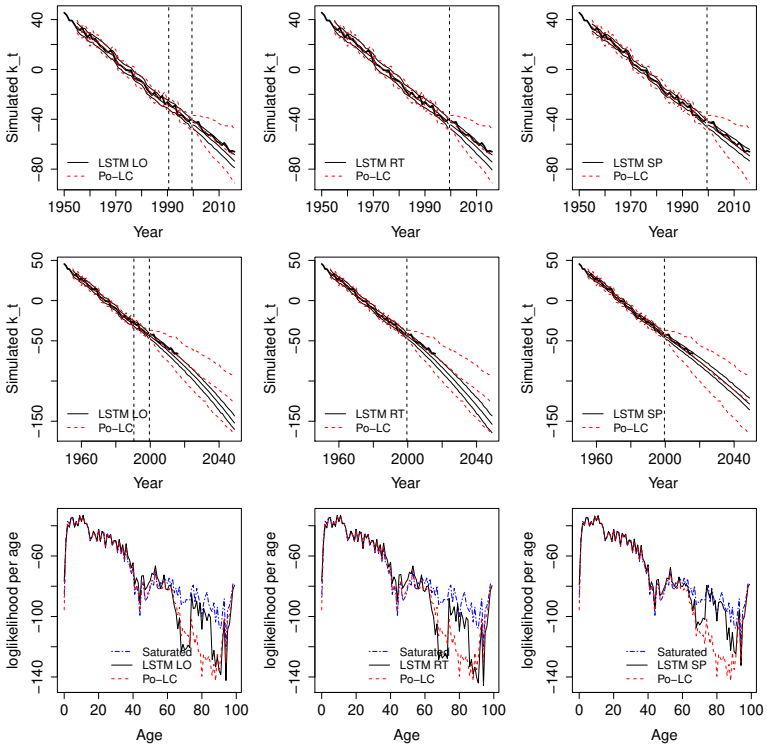

FIGURE 7. Swedish females, LSTM ensembles trained on "raw data" with activation function ReLu: Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

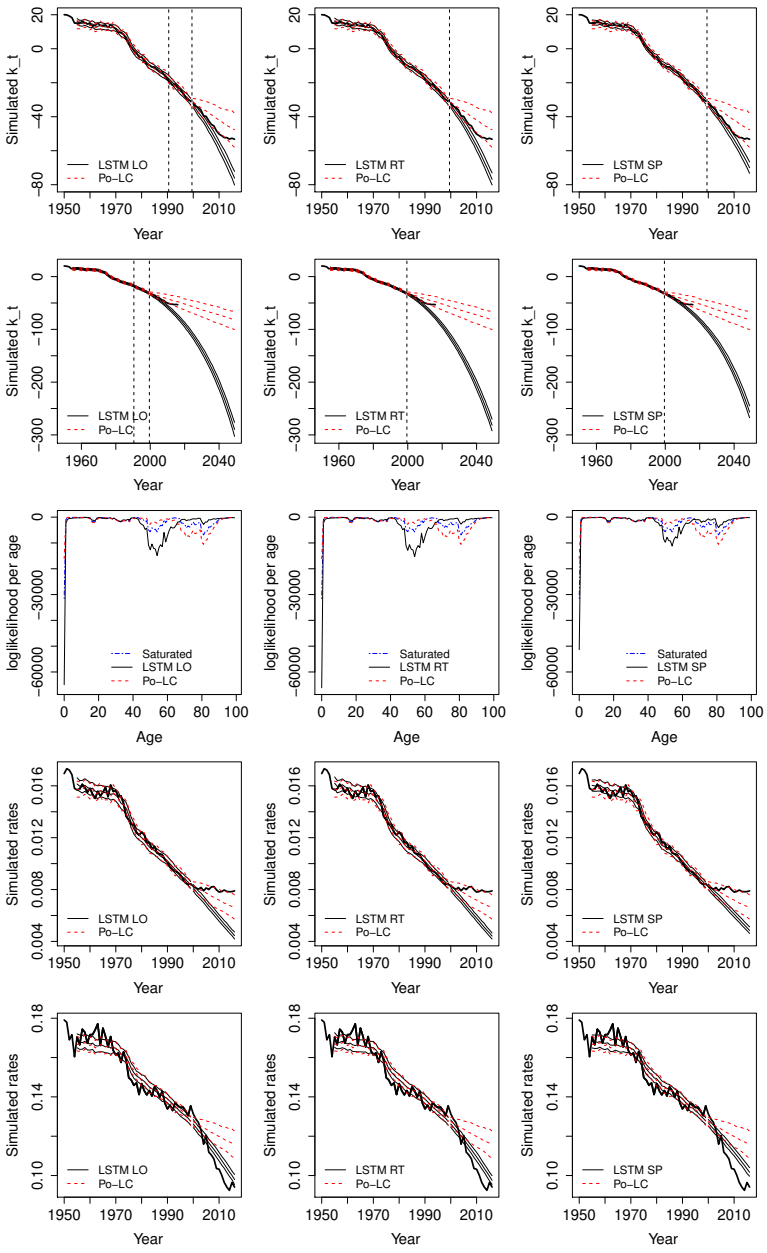

FIGURE 8. USA males, LSTM ensembles trained on "raw data" with activation function ReLu: 1st row: κ_t in-sample and out-of-sample. 2nd row: κ_t long-term prediction. 3rd row: log-likelihood per age, out-of-sample. 4th row: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55. 5th row: age 85.

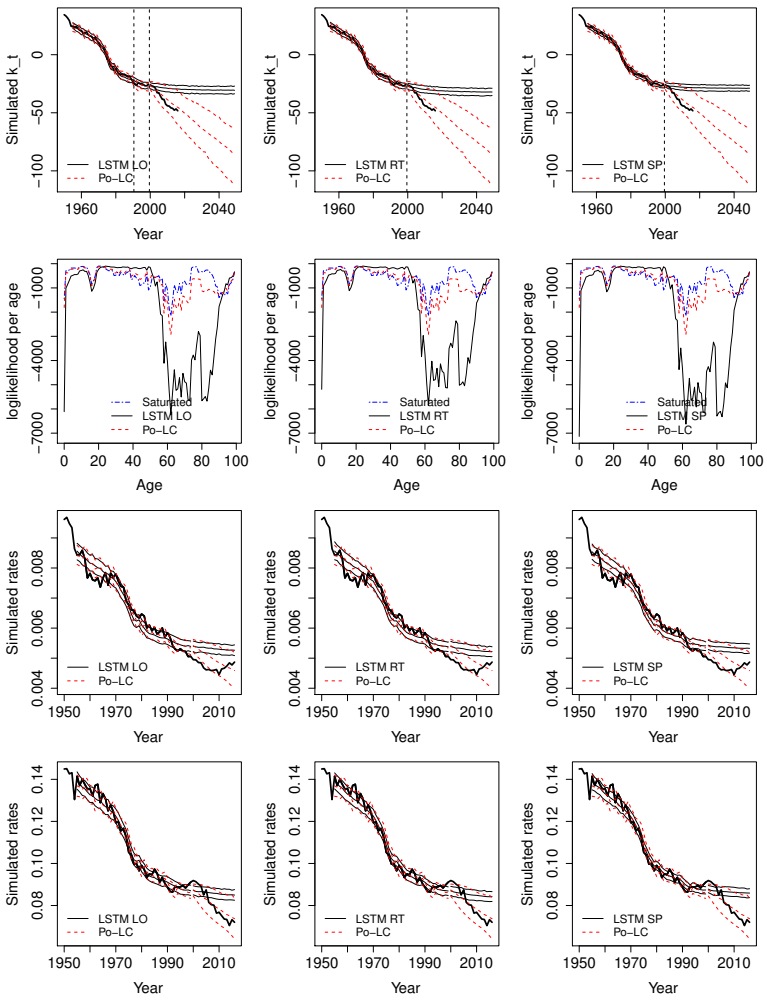

FIGURE 9. USA females, LSTM ensembles trained on "raw data" with activation function ReLu: 1st row: κ_t long-term prediction. 2nd row: log-likelihood per age, out-of-sample. 3rd row: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55. 4th row: age 85.

5. 1950–1999, BOOSTING & SCALING

	RWD	LO	RT	SP
ITAmale	489.55	315.74	**163.81**	192.18
ITAfemale	35.65	35.85	31.91	**20.77**
SWEmale	459.02	**250.39**	283.54	300.32
SWEfemale	3.31	81.09	15.58	**8.02**
USAmale	32.11	**23.71**	28.04	34.12
USAfemale	10.37	41.24	**23.20**	34.00

TABLE 3. Out-of-sample MSE (2000–2016) for LSTM ensembles trained on residual after boosting and scaling (1950–1999) with activation function tanh.

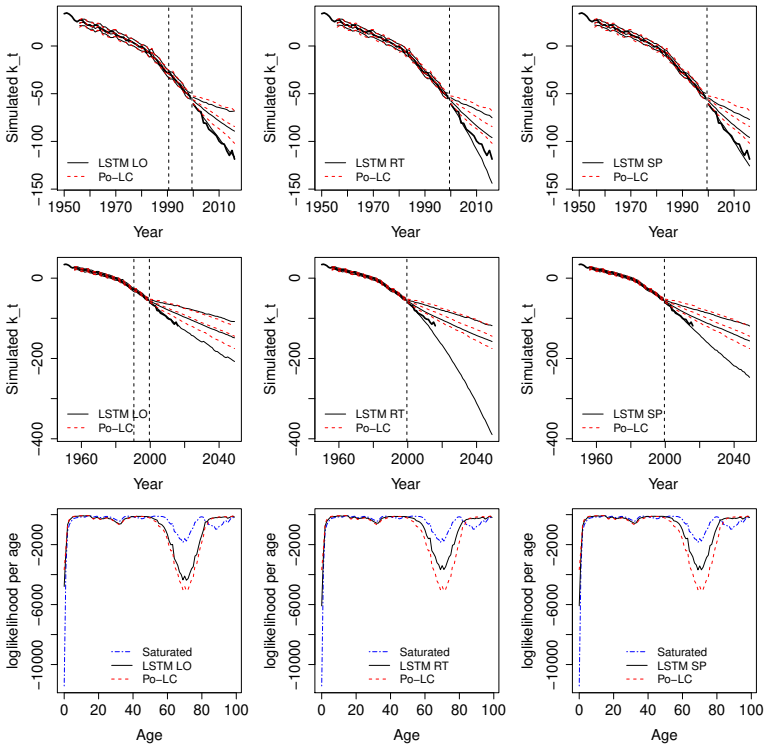

FIGURE 10. Italian males, boosting and scaling (1950–1999) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

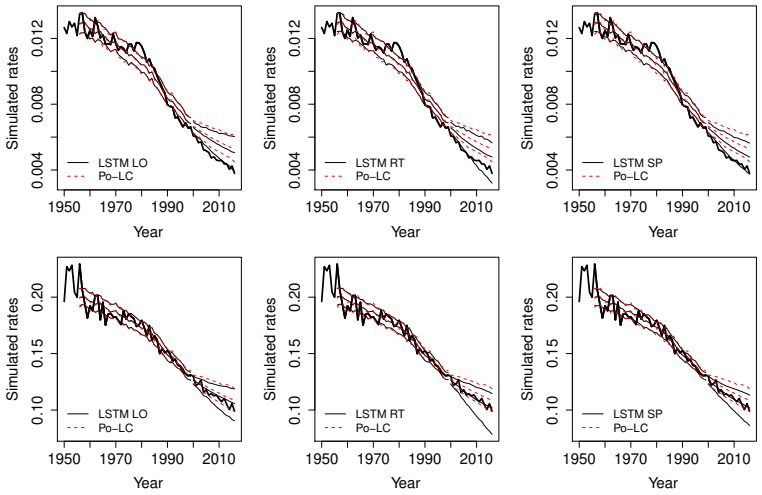

FIGURE 11. Italian males, boosting and scaling (1950–1999) with activation function tanh. Top: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55. Bottom: age 85.

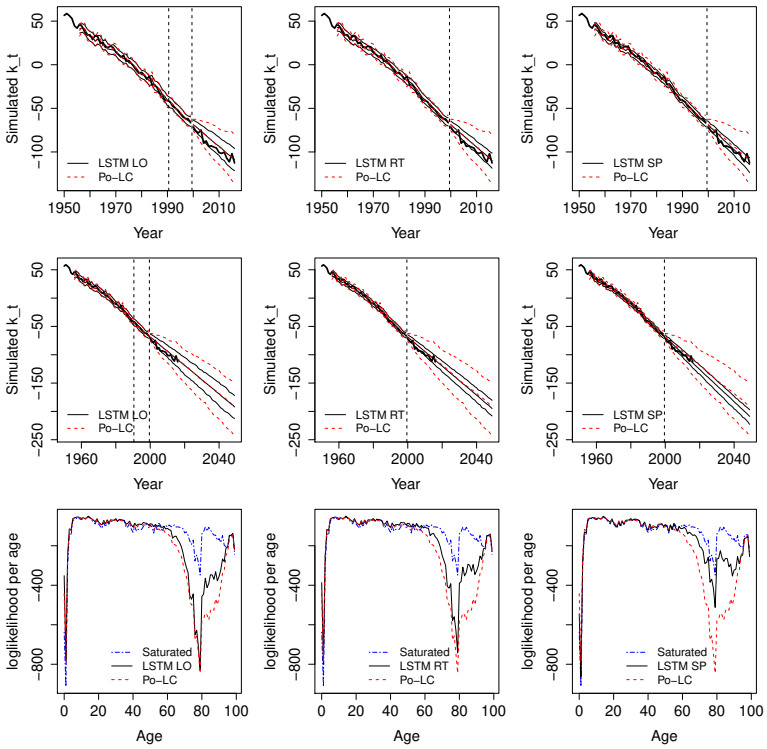

FIGURE 12. Italian females, boosting and scaling (1950–1999) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

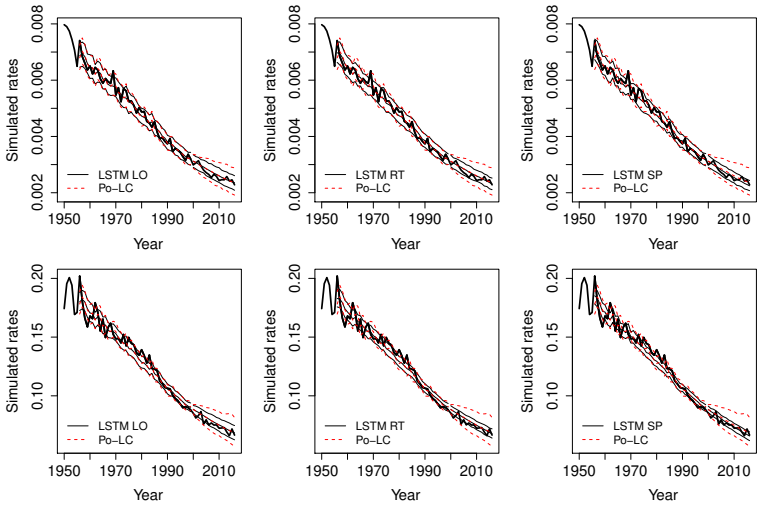

FIGURE 13. Italian females, boosting and scaling (1950–1999) with activation function tanh. Top: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55. Bottom: age 85.

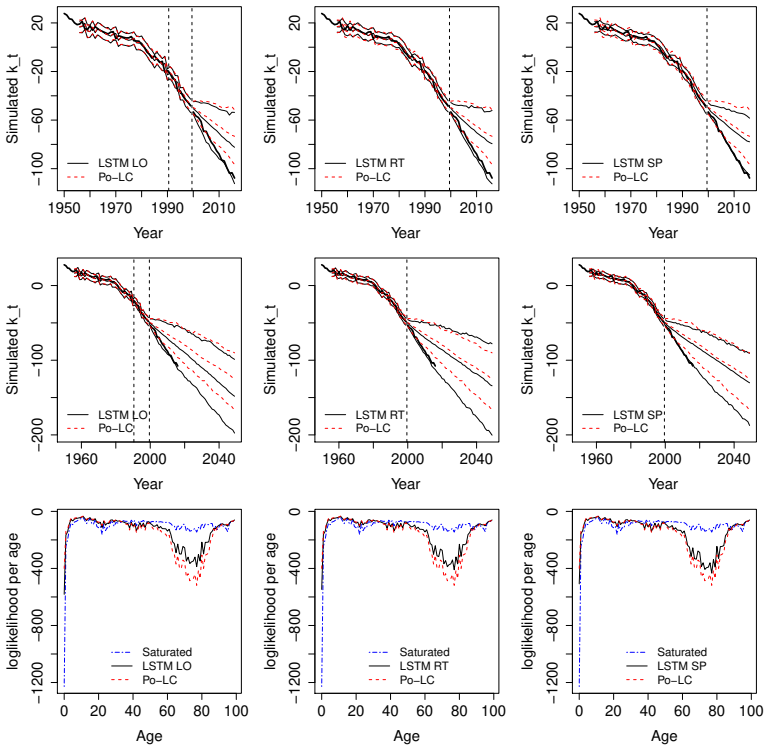

FIGURE 14. Swedish males, boosting and scaling (1950–1999) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

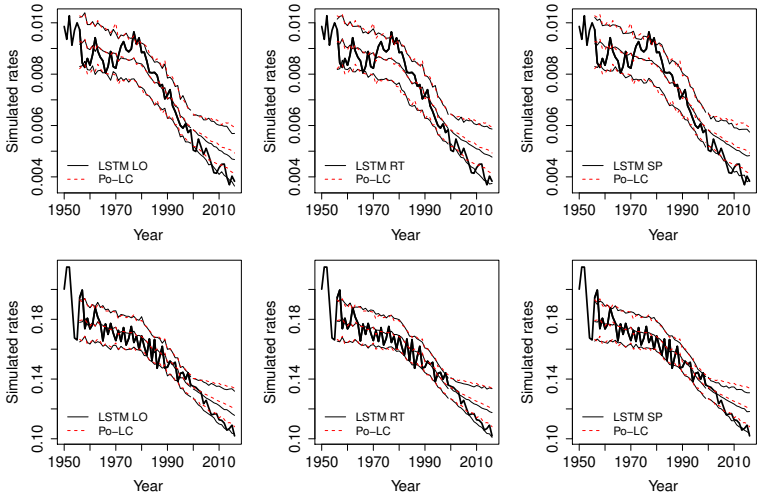

FIGURE 15. Swedish males, boosting and scaling (1950–1999) with activation function tanh. Top: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55. Bottom: age 85.

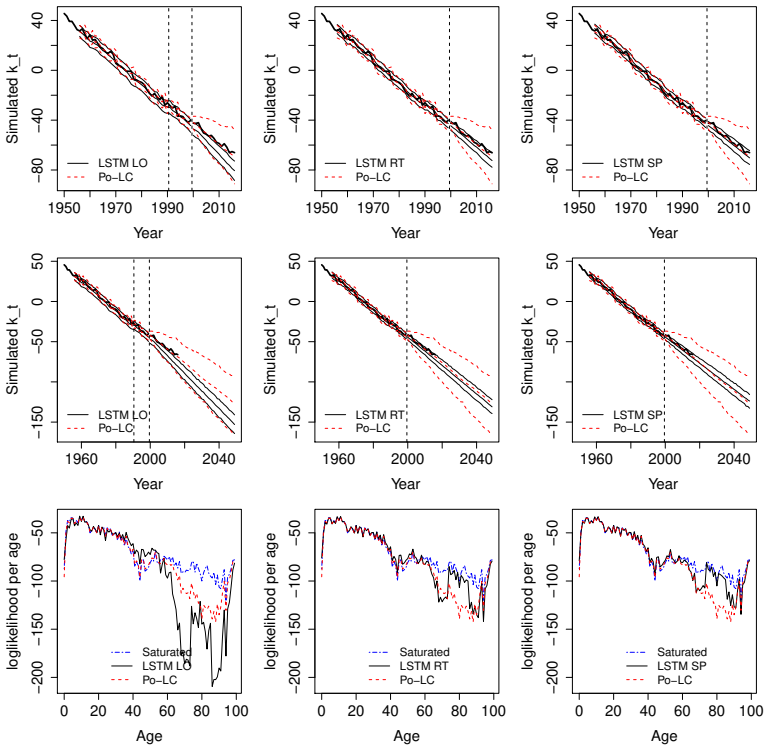

FIGURE 16. Swedish females, boosting and scaling (1950–1999) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

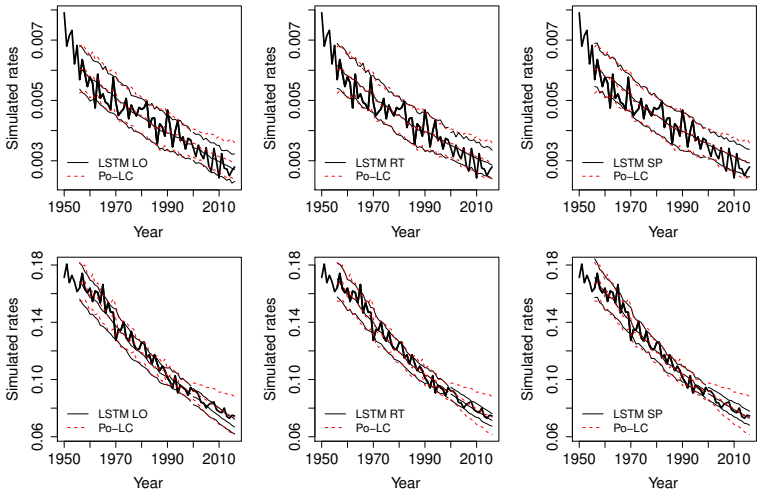

FIGURE 17. Swedish females, boosting and scaling (1950–1999) with activation function tanh. Top: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55. Bottom: age 85.

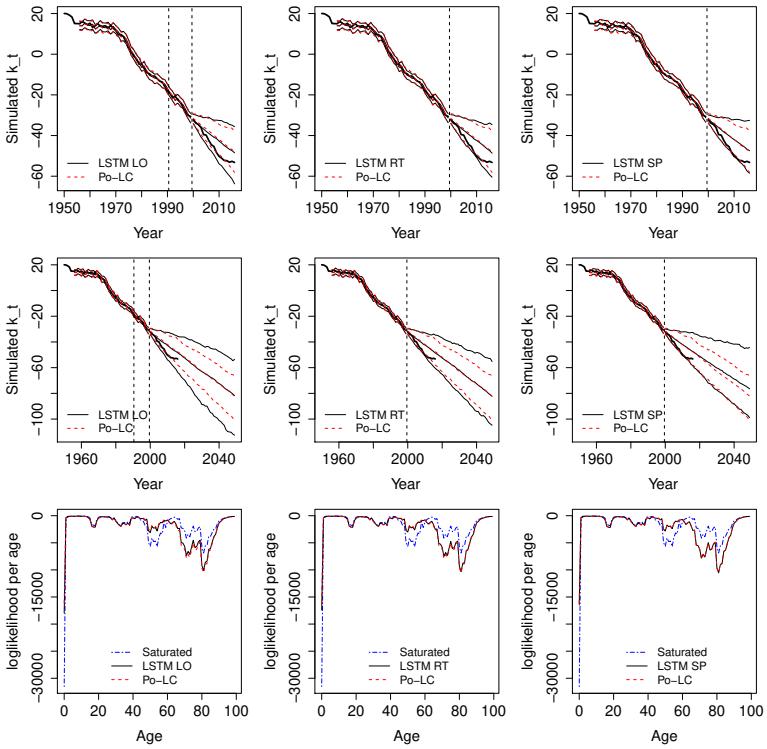

FIGURE 18. USA males, boosting and scaling (1950–1999) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

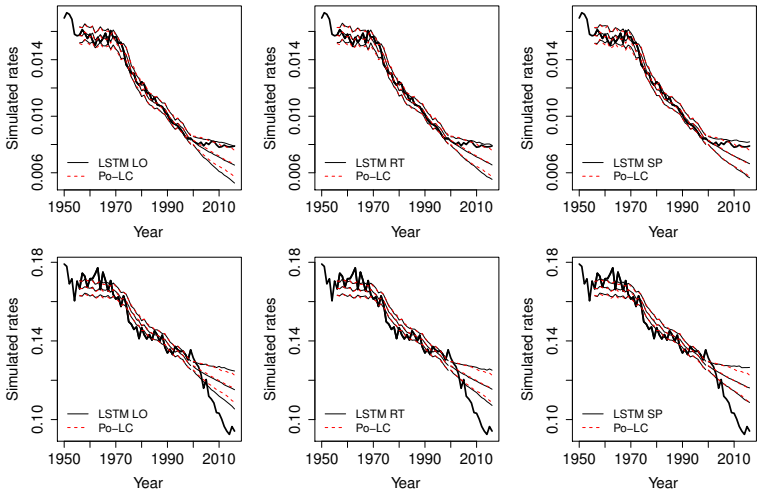

FIGURE 19. USA males, boosting and scaling (1950–1999) with activation function tanh. Top: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55. Bottom: age 85.

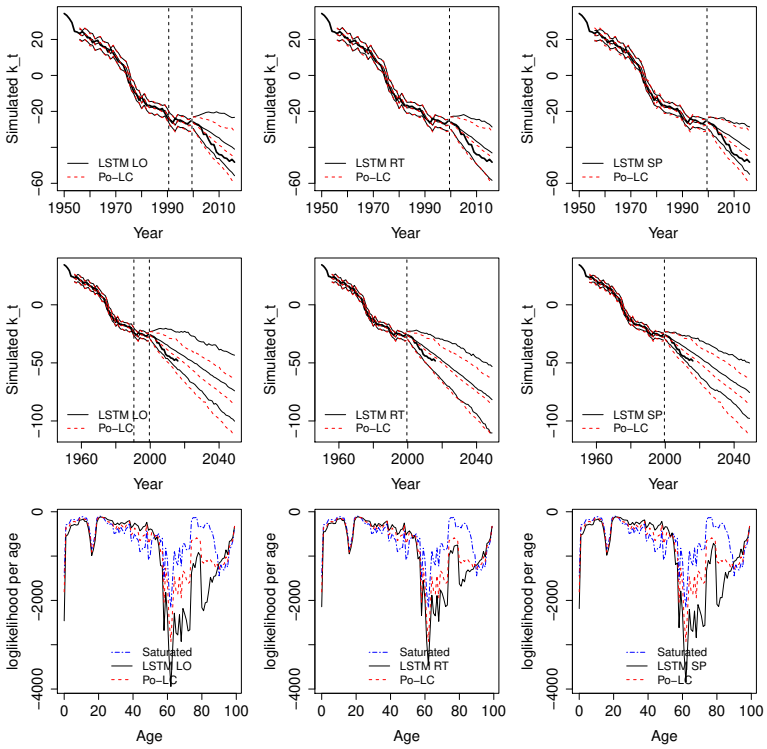

FIGURE 20. USA females, boosting and scaling (1950–1999) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

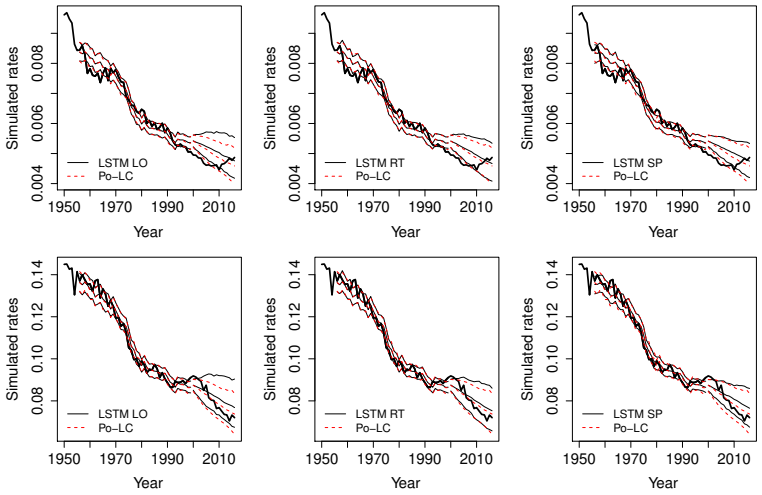

FIGURE 21. USA females, boosting and scaling (1950–1999) with activation function tanh. Top: Observed mortality rates 1950–2016, and predicted simulated mortality rates, age 55. Bottom: age 85.

6. 1970-1989, BOOSTING & SCALING

	RWD	LO	RT	SP
ITAmale	88.79	**22.66**	120.50	47.04
ITAfemale	8.75	16.29	12.70	**10.63**
SWEmale	217.27	330.35	19.21	**19.05**
SWEfemale	28.88	**25.32**	67.48	49.97
USAmale	1.08	2.08	**1.27**	4.34
USAfemale	126.81	25.10	44.65	**6.06**

TABLE 4. Out-of-sample MSE (1990–2006) for LSTM ensembles trained on residual after boosting and scaling (1970–1989) with activation function tanh.

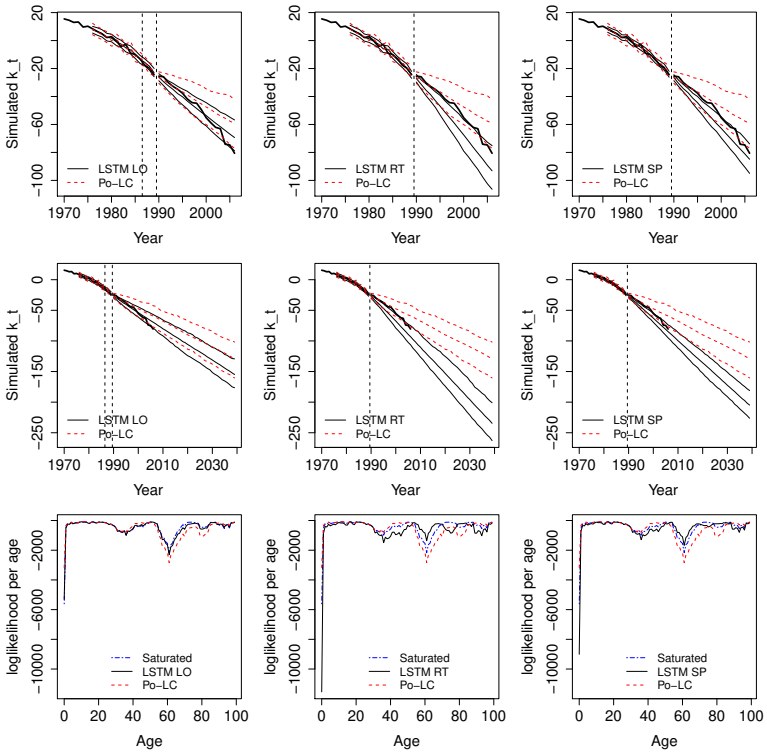

FIGURE 22. Italian males, boosting and scaling (1970–1989) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

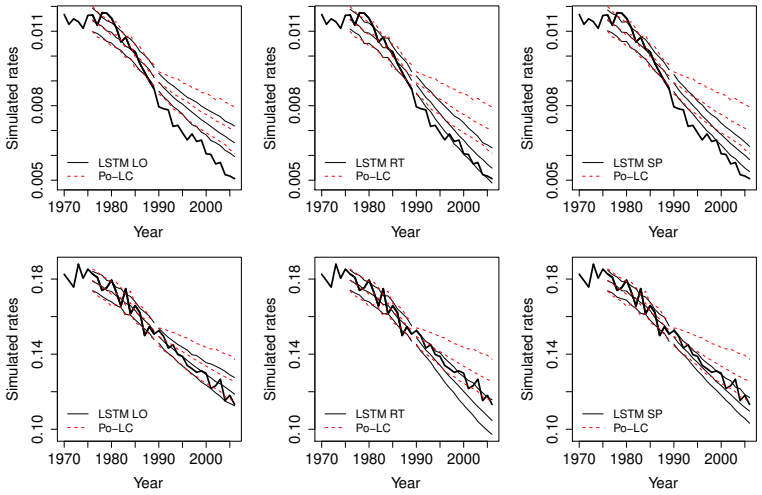

FIGURE 23. Italian males, boosting and scaling (1970–1989) with activation function tanh. Top: Observed mortality rates 1970–2006, and predicted simulated mortality rates, age 55. Bottom: age 85.

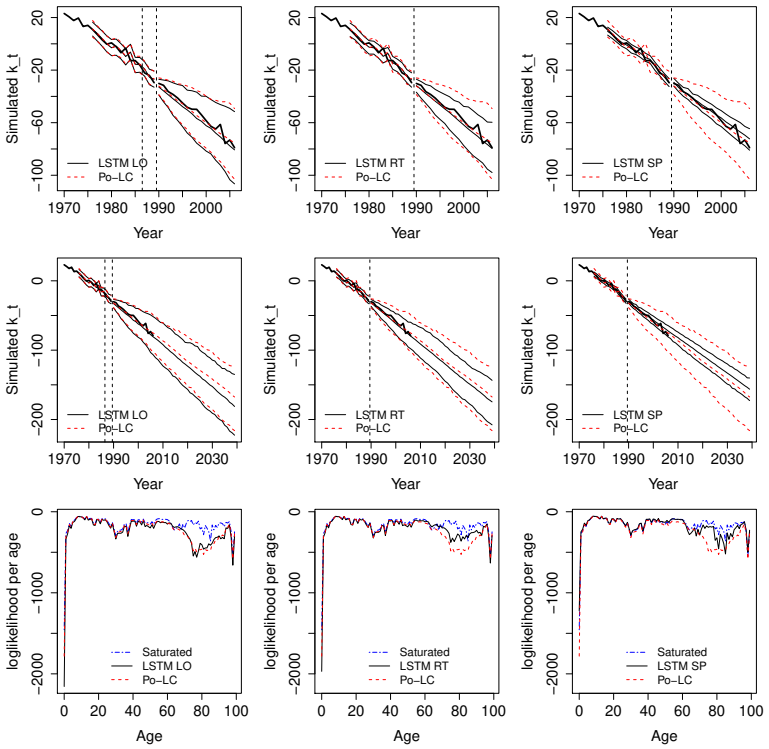

FIGURE 24. Italian females, boosting and scaling (1970–1989) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

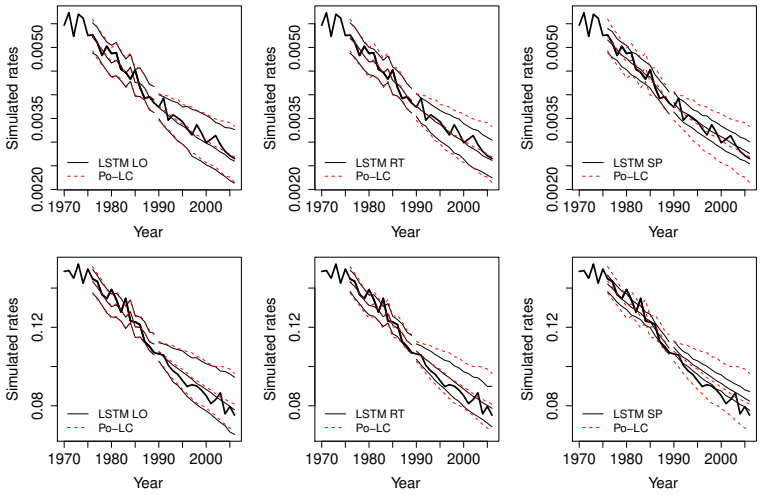

FIGURE 25. Italian females, boosting and scaling (1970–1989) with activation function tanh. Top: Observed mortality rates 1970–2006, and predicted simulated mortality rates, age 55. Bottom: age 85.

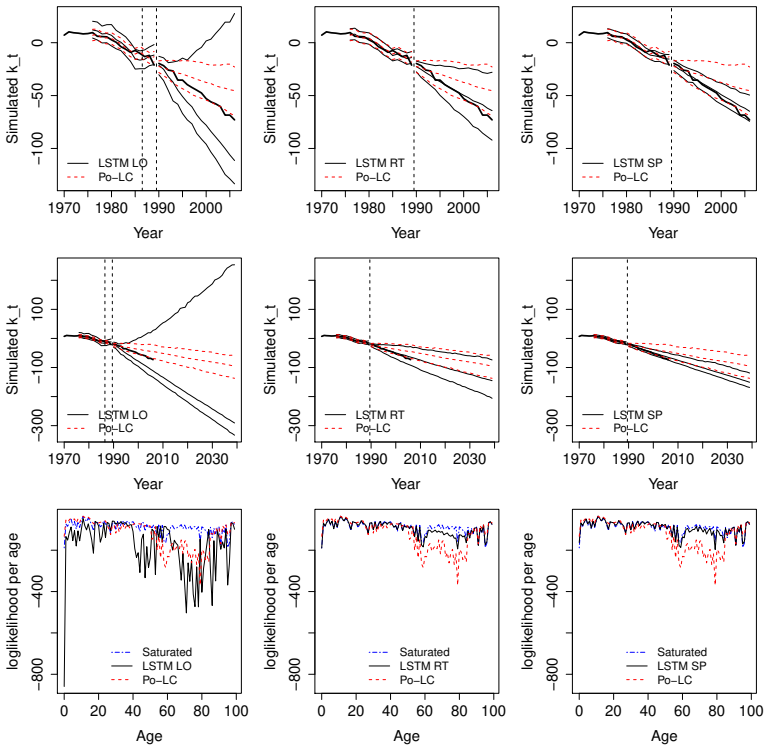

FIGURE 26. Swedish males, boosting and scaling (1970–1989) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

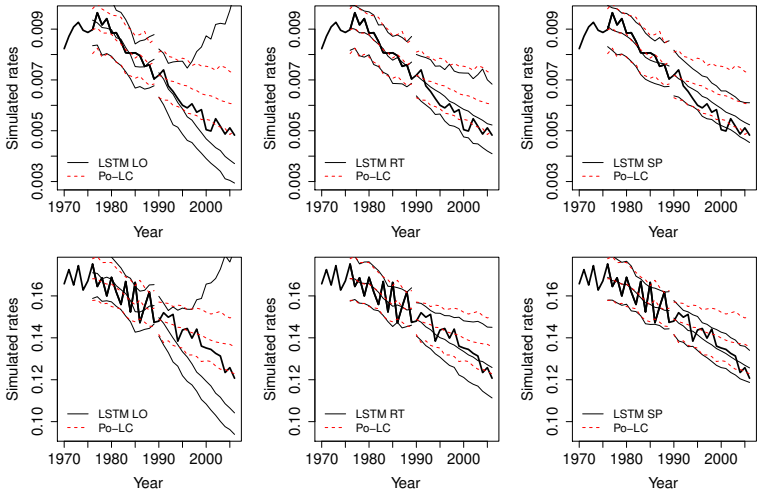

FIGURE 27. Swedish males, boosting and scaling (1970–1989) with activation function tanh. Top: Observed mortality rates 1970–2006, and predicted simulated mortality rates, age 55. Bottom: age 85.

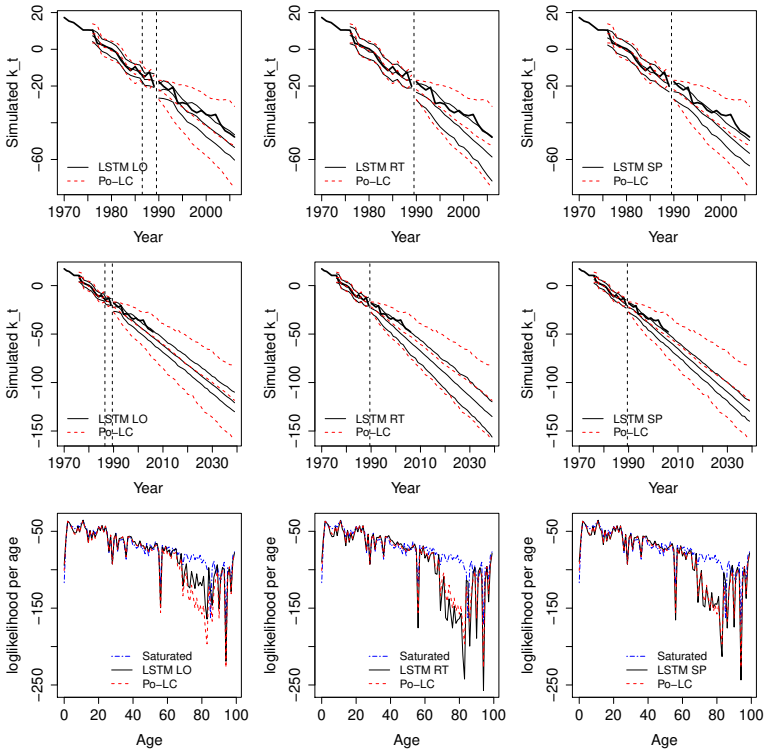

FIGURE 28. Swedish females, boosting and scaling (1970–1989) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

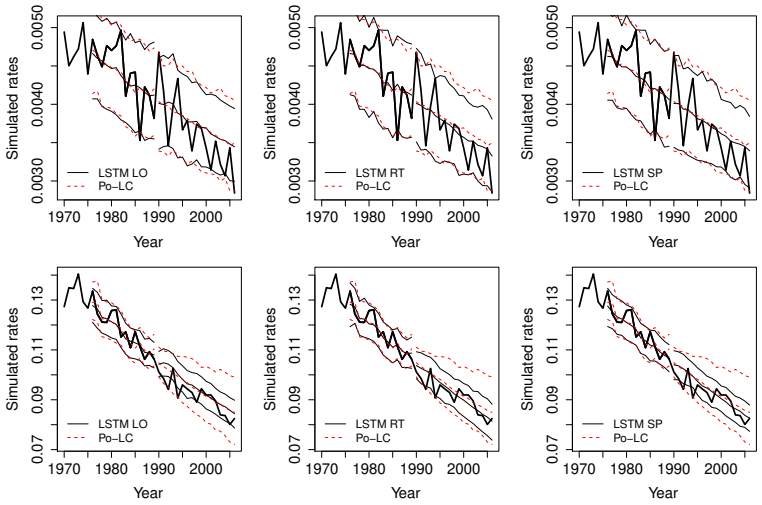

FIGURE 29. Swedish females, boosting and scaling (1970–1989) with activation function tanh. Top: Observed mortality rates 1970–2006, and predicted simulated mortality rates, age 55. Bottom: age 85.

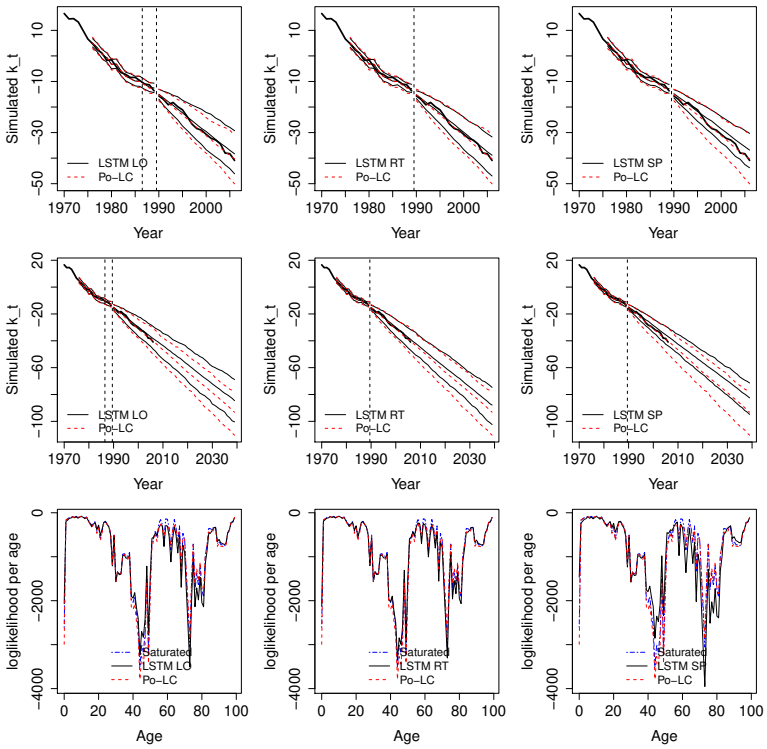

FIGURE 30. USA males, boosting and scaling (1970–1989) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

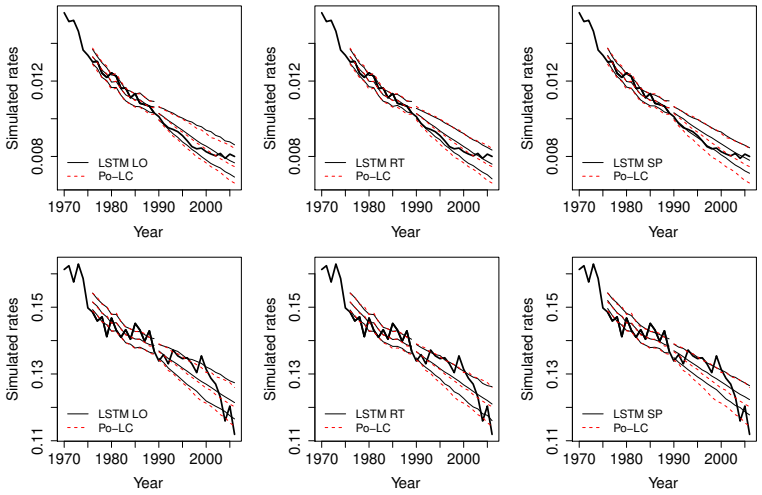

FIGURE 31. USA males, boosting and scaling (1970–1989) with activation function tanh. Top: Observed mortality rates 1970–2006, and predicted simulated mortality rates, age 55. Bottom: age 85.

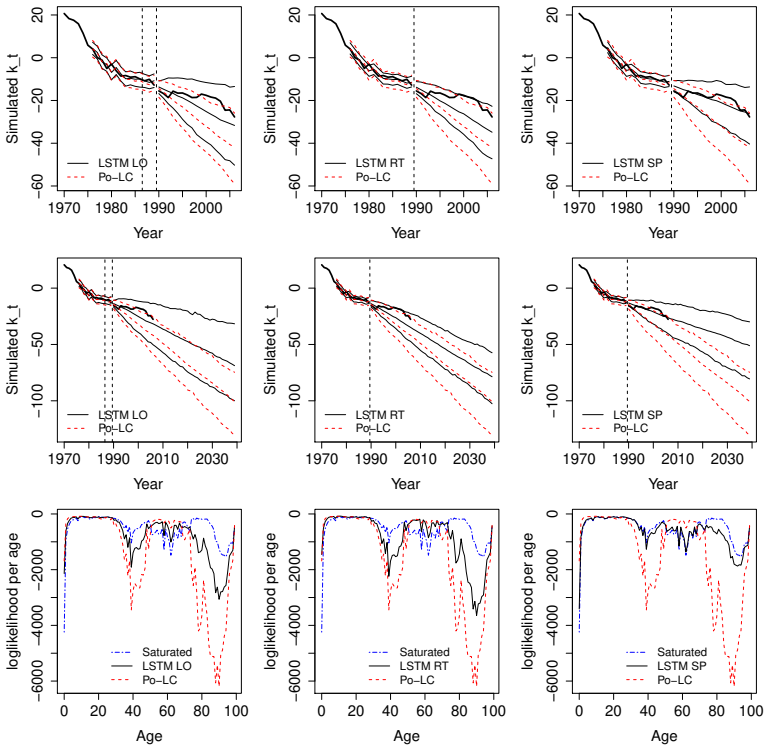

FIGURE 32. USA females, boosting and scaling (1970–1989) with activation function tanh. Top: κ_t in-sample and out-of-sample. Middle: κ_t long-term prediction. Bottom: log-likelihood per age, out-of-sample.

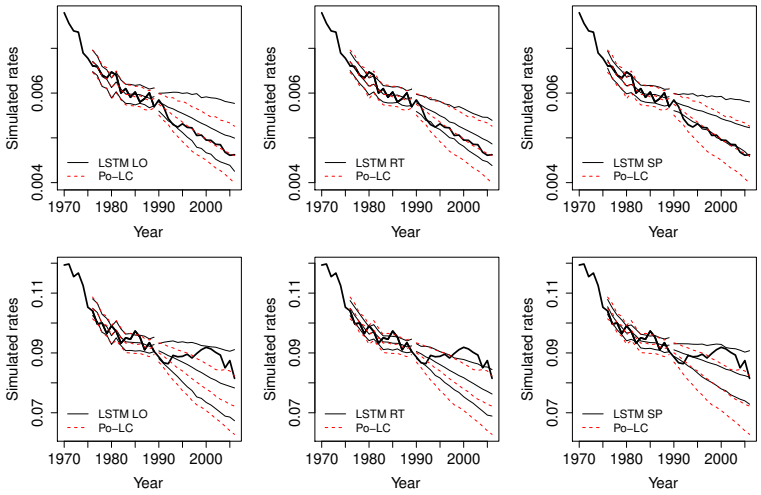

FIGURE 33. USA females, boosting and scaling (1970–1989) with activation function tanh. Top: Observed mortality rates 1970–2006, and predicted simulated mortality rates, age 55. Bottom: age 85.

7. NUMBER OF MODEL CALIBRATIONS IN EACH ENSEMBLE

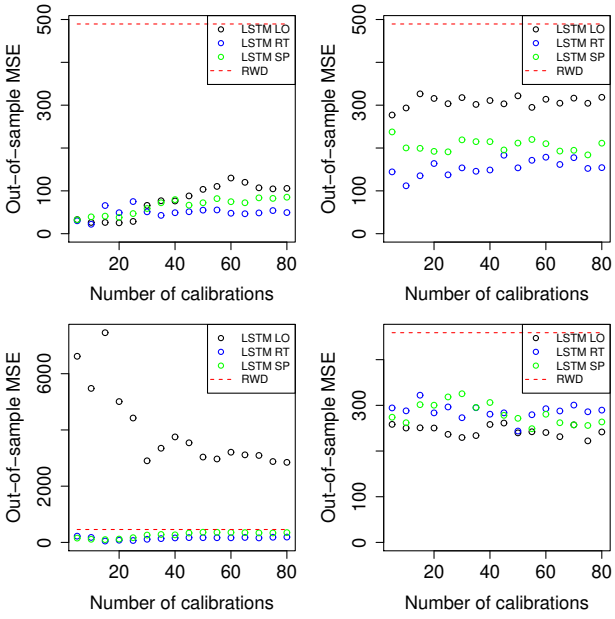

FIGURE 34. Ensemble MSE over different number of calibrations, out-of-sample. Top: Italian males. Bottom: Swedish males. Left: LSTM ensembles trained on "raw" data (1950-1999) with activation function ReLu. Right: LSTM ensembles using boosting and scaling (1950-1999) with activation function tanh.

References

[1] François Chollet, JJ Allaire, et al. R interface to keras. `https://github.com/rstudio/keras`, 2017.

[2] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning, 2nd edition*. Springer series in statistics New York, 2008.

[4] Human Mortality Database. University of California, Berkeley (USA), and Max Planck Institute for Demographic Research (Germany). Available at `http://www.mortality.org` or `http://www.humanmortality.de` (data downloaded on 2020-09-08), 2020.

[5] Mathias Lindholm and Lina Palmborg. Efficient use of data for LSTM mortality forecasting. 2021.

[6] Andrea Nigri, Susanna Levantesi, Mario Marino, Salvatore Scognamiglio, and Francesca Perla. A deep learning integrated Lee–Carter model. *Risks*, 7(1):33, 2019.

[7] Francesca Perla, Ronald Richman, Salvatore Scognamiglio, and Mario V. Wüthrich. Time-series forecasting of mortality rates using deep learning. *Scandinavian Actuarial Journal*, 0(0):1–27, 2021.

[8] Ronald Richman and Mario V Wüthrich. Lee and Carter go machine learning: Recurrent neural networks. *Available at SSRN 3441030*, 2019.

[9] Ronald Richman and Mario V Wüthrich. Nagging predictors. *Risks*, 8(3):83, 2020.