Mathematical Statistics
Stockholm University

Efficient use of data for LSTM mortality forecasting

Mathias Lindholm
Lina Palmborg

Research Report 2021:2

ISSN 1650-0377

Postal address:

Mathematical Statistics
Dept. of Mathematics
Stockholm University
SE-106 91 Stockholm
Sweden

Internet:

http://www.math.su.se

Mathematical Statistics
Stockholm University
Research Report **2021:2**,
http://www.math.su.se

Efficient use of data for LSTM mortality forecasting

Mathias Lindholm and Lina Palmborg

March 2021

Abstract

We consider a simple long-short term memory (LSTM) neural network extension of the Poisson Lee-Carter model, with a particular focus on different procedures for how to use training data efficiently, combined with ensembling to stabilise the predictive performance. We compare the standard approach of withholding the last fraction of observations for validation, with two other approaches: sampling a fraction of observations randomly in time; and splitting the population into two parts by sampling individual life histories. We provide empirical and theoretical support for using these alternative approaches.

Furthermore, to improve the stability of long-term predictions, we consider boosted versions of the Poisson Lee-Carter LSTM. In the numerical illustrations it is seen that even in situations where mortality rates are essentially log-linear as a function of calendar time, the boosted model does not perform significantly worse than a simple random walk with drift, and when non-linearities are present the predictive performance is improved. Moreover, boosting allows us to obtain reasonable model calibrations based on as few data points as 20 years.

1 Introduction

The perhaps most famous mortality forecasting model is the Lee-Carter model, see [16], which is a simple model for mortality rates. This model assumes that the age and calendar time effects follow a log-linear relationship, which makes parameter estimation very simple. That is, if we let $\mu_{x,t}$ denote the mortality rate for age x during calendar year t, it is assumed that given estimates $\widehat{\mu}_{x,t}$ it holds that

$$\log(\widehat{\mu}_{x,t}) = \alpha_x + \beta_x \kappa_t,$$

for $x \in \mathcal{X}$ and $t \in \mathcal{I}$, where $\mathcal{X}$ denotes observed ages, and $\mathcal{I}$ denotes observed time points. When it comes to producing forecasts of future mortality the "trick" used is to model the *estimated* κ_ts as a univariate Gaussian process, often a random walk with drift, i.e.

$$\widehat{\kappa}_{t+1} = \gamma + \widehat{\kappa}_t + \epsilon_{t+1}, \tag{1}$$

where $\epsilon_t \sim \mathsf{N}(0, \sigma^2)$ and i.i.d. The Gaussian process used to describe the variation in the κ_ts is easy to forecast into the future, and given these forecasts it is straightforward to produce future values of $\mu_{x,t}$.

The above outlined description of the Lee-Carter model is a model describing the evolution of mortality rates, whereas what we *observe* are death counts. [16] discuss ways for adjusting for this, but a more natural approach is the one underlying the Poisson Lee-Carter model from [5]: Let $D_{x,t}$ denote the number of individuals dying being of age x during calendar year t, and let $r_{x,t}$ denote the total exposure-to-risk for individuals being x years old during calendar year t. It is then assumed that

$$D_{x,t} \mid r_{x,t} \sim \mathsf{Po}(r_{x,t}\mu_{x,t}(\boldsymbol{\theta})), \tag{2}$$

where

$$\mu_{x,t}(\boldsymbol{\theta}) := \exp\{\boldsymbol{\theta}_{x,t}\} := \exp\{\alpha_x + \beta_x \kappa_t\}, \tag{3}$$

which corresponds to a Poisson regression model with a log-link function, whose parameters can be estimated using standard maximum likelihood theory. Still, in order to be able to produce forecasts the estimated κ_ts are modelled as a one-dimensional (Gaussian) process, e.g. following (1).

If one wants to avoid the inconsistency of using the above described two-step estimation procedure, first estimating parameters, and second treating the

estimated parameters as outcomes of a stochastic process, one can use state-space models, see e.g. [7] for the standard Lee-Carter model, and [1] for the Poisson Lee-Carter model.

Apart from improving on the estimation procedure, another line of research concerns using more flexible modelling approaches. From a constructive modelling perspective the Poisson distribution assumption is natural, see e.g. [5, 1], and one can hence consider the following generalisation concerning the modelling of the $\widehat{\kappa}_t$s:

Model (Generalised Lee-Carter)**.**

$$\widehat{\kappa}_{t+1} = f(\mathcal{F}_t; \boldsymbol{\eta}) + \epsilon_{t+1}, \tag{4}$$

where $\epsilon_t \sim \mathsf{N}(0, \sigma^2)$ and i.i.d., and where $\mathcal{F}_t = \sigma\{\widehat{\kappa}_s; s \leq t\}$ and

$$f(\mathcal{F}_t; \boldsymbol{\eta}) := \mathbb{E}[\widehat{\kappa}_{t+1} \mid \mathcal{F}_t](\boldsymbol{\eta}).$$

The approach that will be taken in the present paper is to model $f(\mathcal{F}_t; \boldsymbol{\eta})$ as a long-short term memory (LSTM) neural network, see e.g. [11] for a general introduction to recurrent neural networks, and e.g. [21, 19] for LSTM versions of [16] and [5] and e.g. [26, 24, 22, 12] for other neural network models used for mortality forecasting, and e.g. [8, 17] for tree-based machine learning techniques. Hence, we do not investigate the appropriateness of the model structure in (2) and (3) as compared to other model structures or the inconsistency of the two-step estimation procedure. Instead we see the model structure and the MLE of $\boldsymbol{\theta}$ as given, and focus on modelling the $\widehat{\kappa}_t$s. In this respect, the MLE of the κ_ts can be regarded as "data" when calibrating the neural network model.

Important aspects of using neural network models is to decide on (i) the architecture of the model, and (ii) the number of epochs to be used for calibrating the model. In the present paper we will illustrate the performance for a number of architectures, not claiming to show the performance of the best possible architecture. We will instead focus considerably more on (ii) and, in particular, on the effect of using different amounts of data for calibration. The reason for this is that LSTM neural network models tend to have a large number of parameters that needs to be calibrated. This implies that you need to have access to a rather large amount of data to be used for calibration. Moreover, neural network models (in general) are calibrated using iterative procedures and the question is for how long this iterative procedure should be carried out. The standard procedure of how to decide on the number of iterations to be used is based on so-called "early stopping"

where you have one set of data for in-sample training and another set of data for validation (out-of-sample training), where you stop the iterative calibration procedure when the performance on the out-of-sample validation data starts to deteriorate. An obvious risk with using early stopping is that the optimisation method might have converged to a local minimum. One way of reducing this problem is to average over a number of different models, using random initialisation of the parameters. This is an example of an ensemble model.

Further, in the present paper the primary interest is on *one-dimensional* LSTM neural network models, where the only dimension is time. Based on the discussion in the previous paragraph, this implies that we would expect to need *long* time series in order to obtain reliable model calibrations. When it comes to data for entire countries this may be feasible, but for e.g. life insurers this may become problematic. Moreover, even if mortality data for longer time periods is available, using the full historical data set might not be appropriate when staying within the simple model structure in (2) and (3), since when increasing the length of the time series, one also increases the time period for which it should be reasonable to use *constant* α_xs and β_xs. Thus a compromise is needed between having enough data to improve the performance of the LSTM model, and at the same time ensuring that the performance of the global model (2) and (3) does not degrade based on the time window being too long.

For the calibration we will consider the following three approaches for splitting the data into data used for in-sample training and out-of-sample validation:

(i) The standard approach of withholding the last fraction of observations (chronologically in time) as validation data.

(ii) Sampling the validation data randomly in time.

(iii) Sampling individuals and randomly assigning them to subsets of the underlying *population*, where one subset is used for in-sample training and another subset is used for out-of-sample validation, *without splitting the time dimension.*

The idea behind approaches (ii) and (iii) is to make more efficient use of data, considering that the amount of available information might be restricted. The obvious drawback with approach (i) is that if we have a small data set and split it into data for in-sample training and out-of-sample validation, the

number of examples used for training will be reduced further. Furthermore, if the validation set is inherently different from the rest of training data or future data, then the model that minimises the error on the validation set might not generalise well. With approach (ii), since we are sampling validation data randomly in time, this approach will also reduce the number of examples used for training, but here we have the ability to draw the validation set randomly several times, and train a number of different models on these different splits into training data and validation data. An ensemble model based on these individual model calibrations will as a whole be trained on the full data set and does, hence, not risk choosing one single validation set which is not representative of the time series. Concerning approach (iii), this allows us to use the whole time period for both in-sample training and out-of-sample validation by sampling i.i.d. individual life histories. That is, since both training and validation data are based on sampling i.i.d. individual life histories, the calibrated $\widehat{\kappa}_t$ predictor from the training dataset should capture the relevant time-dynamics in the validation dataset as well. In Section 3.1-3.3 these three approaches are discussed in more detail.

Furthermore, it is worth noting that the methods contain different implicit views on the (trend-)stationarity of data. Method (ii) and (iii) essentially treat all observations in the time-dimension as equally relevant, which aligns with the underlying model assumptions in (2) and (3). Method (i) instead views the last observations as more relevant for the future, since the parameters chosen are the ones that give the best performance on the validation set only consisting of the last observations. Depending on the data, this might be an appropriate assumption to make. However, such an assumption also indicates that the underlying model defined by (2) and (3) is not suitable for the task at hand.

Finally, in order to try to keep the amount of information necessary for calibration at a minimum, we will combine the above three approaches with the following approach:

(iv) LSTM boosting of the standard Poisson Lee-Carter model from [5].

This means that we will use the estimated mean-function for the $\widehat{\kappa}_t$ process from the standard Poisson Lee-Carter model from [5] as an intercept in the LSTM model. Given that the mean-function from the Poisson Lee-Carter model is reasonably representative for the observed data, the LSTM model only needs to *improve* on this baseline model. This should be considerably more stable than trying to learn all data dynamics from start when only having access to a limited amount of data. Moreover, other potential benefits

of using boosting is that this procedure likely will make the data fed into the LSTM model approximately trend adjusted. This in turn may prove beneficial when making long-term prediction. This is described in more detail in Section 3.7.

The approach of modelling $f(\mathcal{F}_t; \boldsymbol{\eta})$ in (4) as an LSTM neural network is the same as in [21, 19]. There are, however, some important differences in both implementation and methodology. First, we ensure that there is a clear distinction between the out-of-sample validation set, used when training the model, and the test set representing future data used for evaluating the model performance. This is important in order to ensure that the model evaluation is not biased by the model having seen the test data during training. Secondly, we construct an LSTM model with lag order larger than one, to be able to see if any improvement is due to using a *recurrent* network model, or if it is only the effect of allowing for non-linearities. As discussed in Section 2, an LSTM model where sequences are of length 1 is essentially no different from a feedforward neural network model. Thirdly, we also evaluate the model performance based on the log-likelihood of the full model, not only the MSE of the $\widehat{\kappa}_t$s, since the goal is to predict mortality rates. In particular, in our numerical illustrations in Section 5 examples are given where the MSE based on the $\widehat{\kappa}_t$s contradict the log-likelihood for the observed deaths.

The remainder of the paper is organised as follows: Section 2 provides a brief background to LSTM neural network models, Section 3 introduces the different calibration procedures, model aggregation (ensembling), and boosting, followed by a short section on likelihood considerations and performance measures in Section 4. The effects of using the different calibration techniques are illustrated on Swedish, Italian and US mortality data from [15], which is done in Section 5. For more detailed numerical analyses and additional comparisons, see the Supplementary Materials [18]. The paper ends with a number of closing remarks in Section 6.

2 LSTM neural network models

The long short-term memory (LSTM) neural network model belongs to a specific type of recurrent neural networks (RNNs) called gated RNNs. RNNs are a form of feedforward neural networks that are specialised at processing sequential data. This means that the output of an RNN is determined based on previous elements of the sequence, while the output of a standard feedforward neural network only depend on the current input. For an introduction

to RNNs, see e.g. [11, Ch. 10]. However, in standard RNNs the same function is composed with itself many times, leading to the so-called vanishing gradient problem, which makes it difficult for standard RNNs to learn long-term dependencies. As a solution to this problem [14] developed the LSTM model, which was later extended in [10] where the so-called forget gate was introduced. Thus the LSTM model is a natural model class to consider for time series modelling.

Let $\boldsymbol{x}_t \in \mathbb{R}^c$ be the input vector, and $\boldsymbol{h}_t \in \mathbb{R}^d$ the hidden layer vector, where c is the number of features in the input data, and d is the number of neurons in the hidden layer. Following a similar notation as [11, Ch. 10.10], an LSTM cell is described by:

$$\begin{aligned}
\boldsymbol{f}_t &= \sigma\Big(\boldsymbol{b}^f + \boldsymbol{U}^f \boldsymbol{x}_t + \boldsymbol{W}^f \boldsymbol{h}_{t-1}\Big) \\
\boldsymbol{g}_t &= \sigma\Big(\boldsymbol{b}^g + \boldsymbol{U}^g \boldsymbol{x}_t + \boldsymbol{W}^g \boldsymbol{h}_{t-1}\Big) \\
\boldsymbol{q}_t &= \sigma\Big(\boldsymbol{b}^q + \boldsymbol{U}^q \boldsymbol{x}_t + \boldsymbol{W}^q \boldsymbol{h}_{t-1}\Big) \\
\boldsymbol{s}_t &= \boldsymbol{f}_t \odot \boldsymbol{s}_{t-1} + \boldsymbol{g}_t \odot \phi\Big(\boldsymbol{b} + \boldsymbol{U} \boldsymbol{x}_t + \boldsymbol{W} \boldsymbol{h}_{t-1}\Big) \\
\boldsymbol{h}_t &= \phi(\boldsymbol{s}_t) \odot \boldsymbol{q}_t,
\end{aligned} \tag{5}$$

where $\odot$ denotes the Hadamard product, $\boldsymbol{f}_t$ is the forget gate, $\boldsymbol{g}_t$ is the input gate, $\boldsymbol{q}_t$ is the output gate, $\sigma(\cdot)$ is the logistic sigmoid function, $\phi(\cdot)$ is the activation function, $\boldsymbol{b}^f$, $\boldsymbol{b}^g$, $\boldsymbol{b}^q$, and $\boldsymbol{b}$ denote the biases ($\in \mathbb{R}^d$), $\boldsymbol{U}^f$, $\boldsymbol{U}^g$, $\boldsymbol{U}^q$, and $\boldsymbol{U}$ denote the input weights ($\in \mathbb{R}^{d \times c}$), and $\boldsymbol{W}^f$, $\boldsymbol{W}^g$, $\boldsymbol{W}^q$, and $\boldsymbol{W}$ denote the recurrent weights ($\in \mathbb{R}^{d \times d}$). The initial values are $\boldsymbol{h}_0 = \boldsymbol{0}$ and $\boldsymbol{s}_0 = \boldsymbol{0}$. Since the three gates are all defined in terms of the logistic sigmoid activation function, they take values in $(0, 1)^d$. Hence the gates control to what degree information flows through the memory cell. $\boldsymbol{s}_t$ is the cell state, hence the forget gate controls to what degree the previous cell state $\boldsymbol{s}_{t-1}$ is passed forward to the current state, while the input gate controls to what degree the input at time t and the hidden layer vector at time $t-1$ adjusts the cell state. Finally, the output gate controls to what degree the current cell state is passed forward to the current hidden layer output $\boldsymbol{h}_t$.

Due to the gates in an LSTM cell, each with its own biases, input weights and recurrent weights, an LSTM model tends to have a large number of parameters. To be precise, for each LSTM layer, the number of parameters is $4((c+1)d + d^2)$. As an example, in a shallow LSTM model with 5 neurons and 1-dimensional sequences, the LSTM layer would contribute with 140 parameters.

Since the original LSTM model was introduced, many different variants have been suggested. For an overview of different types of gated RNNs, see [11, Ch. 10.10] and references therein. Here, we focus on the original LSTM model defined above, and restrict our analysis to a shallow model with one hidden LSTM layer.

Remark 1.

(a) In the original LSTM model, two different activation functions were used for updating $\boldsymbol{s}_t$ and $\boldsymbol{h}_t$. In the description of an LSTM cell in (5), we have chosen the same activation function $\phi(\cdot)$, since this is consistent with the implementation in the `R` package `keras`, see [6], used for the numerical illustrations in Section 5.

(b) As in the original LSTM model, we use the logistic sigmoid function for the gate activation, implemented in the `R` package `keras` as "`recurrent_activation`". In several recent papers using LSTM models for forecasting of mortality rates, see e.g. [24, 19], the gate activation has been set to the hyperbolic tangent function. With this choice of gate activation the intuitive interpretation of the gates in the LSTM model does not hold, since they will now take values in $(-1, 1)$ instead of $(0, 1)$.

For time series modelling with an LSTM model, the look-back window of length p of the time series will determine the length of the input sequences in the time-dimension, and is thus a hyperparameter of the model. Let $(\boldsymbol{y}_t)_{t=1}^n$, with $\boldsymbol{y}_t \in \mathbb{R}^c$, be the time series. If the output target is $\boldsymbol{y}_t$, then the matrix of inputs is $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)^\top = (\boldsymbol{y}_{t-p}, \ldots, \boldsymbol{y}_{t-1})^\top \in \mathbb{R}^{p \times c}$. We illustrate the recursions for a model with one LSTM layer when $p = 2$ and $c = 1$ in Figure 1. For this simple example, with output target y_t, we have $x_1 = y_{t-2}$ and $x_2 = y_{t-1}$, and the output is $f(x_1, x_2; \boldsymbol{\eta}) := \mathbb{E}[Y_t \mid x_1, x_2](\boldsymbol{\eta})$.

Remark 2.

(a) Note that the time step index t in the forward propagation equations in (5) corresponds to the position in the sequence $(\boldsymbol{x}_t)_{t=1}^p$, and not the time step index for the original time series. Furthermore, note that the dimension of the weights (and thus the number of parameters) does not depend on the length of the input sequences, hence the same weights are used irrespective of the time step index t.

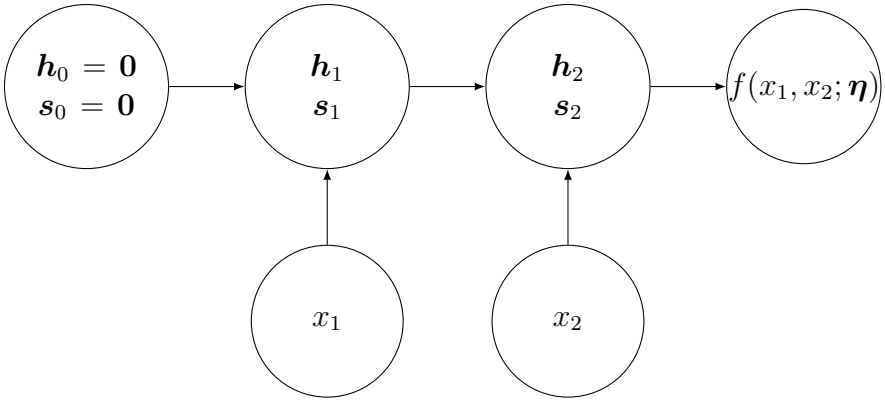

Figure 1: Recursions for model with one LSTM layer, $p = 2$ and $c = 1$.

(b) When the lag order p is set to 1, then the output will simply be a non-linear transformation of the input $\boldsymbol{x}_1$, since the initial value of the hidden layer vector and the cell state is zero. Hence for this case similar results should be achievable with a standard feedforward neural network.

(c) When increasing the lag order p, the hidden layer vector represents the internal short-term memory and the cell state represents the internal long-term memory of previous positions in the sequence. It is for $p > 1$ that one can start taking advantage of the properties of an LSTM model.

The LSTM model is trained using stochastic gradient descent (SGD), hence an estimate of the gradient of the loss function is computed based on a random sample of observations drawn from the training data. Each random sample formed in this way is called a minibatch. The parameter update at each iteration of the SGD algorithm is based on the estimate of the gradient using one minibatch. When the algorithm has cycled through all minibatches in the training data, this constitutes one epoch. The number of epochs is a hyperparameter of the model, and it is commonly determined based early stopping, where training data is split into one set of data that is used for in-sample training and another set of data used for validation. The number of epochs is determined based on the performance on the validation set, with the hope that this improves the performance on the (unseen) test set. Hence, early stopping is a simple method used to prevent overfitting, and is, thus, a form of regularisation.

3 Model calibration, aggregation, and boosting

The general problem that we want to address is how to make efficient use of data for model calibration by analysing three different model calibration procedures. In particular, we are interested in if it is possible to obtain a procedure that produce reasonable model calibrations even when only having access to a limited amount of data. This becomes even more demanding when we want to use early stopping during the calibration of the LSTM model in order to prevent overfitting. When using early stopping, the training data needs to be split into two sets, one which is used for in-sample training, and one which is used for validation (out-of sample training). It is the performance on the out-of-sample validation data that determines when the iterative calibration procedure should be stopped. For time series data, the standard procedure is to withhold the last fraction of observations (chronologically in time) for validation, e.g. according to a 80/20 split. However, when the total number of observations is small, splitting the training data in this way further reduces the number of observations the model can be trained on, which might worsen the performance of the model. Furthermore, there might be important information contained in the withheld observations which the model is never trained on. The smaller the calibration data set, the more likely that there is important information contained in the validation set that is not contained in the data used as input for training, hence the model is never given the opportunity to learn this information.

For a neural network specialised at dealing with sequential data, the lag order p of the model is a hyperparameter. If training data consists of the one-dimensional time series $(\widehat{\kappa}_t)_{t=1}^n$, then $\boldsymbol{x}_t = (\widehat{\kappa}_{t-p}, \ldots, \widehat{\kappa}_{t-1})^\top$ are the covariates for $\widehat{\kappa}_t$, where $t = p+1, \ldots, n$. Hence training data consist of $n-p$ observations. For autoregressive data with lag order p, the standard way to structure data is according to

$$\boldsymbol{K} = \begin{bmatrix} \widehat{\kappa}_1 & \widehat{\kappa}_2 & \ldots & \widehat{\kappa}_p & \widehat{\kappa}_{p+1} \\ \widehat{\kappa}_2 & \widehat{\kappa}_3 & \ldots & \widehat{\kappa}_{p+1} & \widehat{\kappa}_{p+2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \widehat{\kappa}_{n-p} & \widehat{\kappa}_{n-p+1} & \ldots & \widehat{\kappa}_{n-1} & \widehat{\kappa}_n \end{bmatrix}. \tag{6}$$

The matrix $\boldsymbol{K}$ contains the training data for the neural network model, with the first p columns corresponding to the input sequences and the last column corresponding to the output that should be predicted by the model.

We use three different methods for splitting the training data into data used for in-sample training and out-of-sample validation; (i) the aforementioned method of withholding the last fraction of rows of $\boldsymbol{K}$ for validation; (ii) randomly sampling observations in the form of rows of $\boldsymbol{K}$ for validation; and (iii) splitting the underlying population into sub-populations, using one sub-population for in-sample training and one for out-of-sample validation.

3.1 Calibration LO – Withholding the last fraction of observations

When validation data consist of the last $100\alpha\%$, $\alpha \in (0,1)$, of observations, this means that the last $[\alpha(n-p)]$ rows of $\boldsymbol{K}$ will be kept aside as validation data, and the first $n-p-[\alpha(n-p)]$ rows are used as input when training the model, where $[x]$ denotes the integer closest to x. Let $\mathcal{I}$ denote the set of row indices of the matrix $\boldsymbol{K}$, i.e. $\mathcal{I} := \{t : 1 \leq t \leq n-p\}$. Let $\mathcal{V}$ denote the set of row indices of $\boldsymbol{K}$ corresponding to the out-of-sample validation data. Then $\mathcal{V} = \{t : n-p-[\alpha(n-p)]+1 \leq t \leq n-p\}$, and the in-sample training data has index set $\mathcal{T} := \mathcal{I} \setminus \mathcal{V}$. Let $(\boldsymbol{x}_t^*, \widehat{\kappa}_t^*)_{t\in\mathcal{V}}$ denote the validation data. The Calibration LO can then be described according to:

Model calibration.

(i) Let $\widehat{\boldsymbol{\eta}}^{(i)}$ denote the estimate of $\boldsymbol{\eta}$ from $f(\boldsymbol{x}_t;\boldsymbol{\eta}), t \in \mathcal{T}$, of model (4) in the ith calibration epoch.

(ii) Calculate the prediction error

$$(s^{(i)})^2 = \frac{1}{|\mathcal{V}|}\sum_{t\in\mathcal{V}}(\widehat{\kappa}_t^* - f(\boldsymbol{x}_t^*;\widehat{\boldsymbol{\eta}}^{(i)}))^2 \tag{7}$$

and continue the updating procedure of $\widehat{\boldsymbol{\eta}}^{(i)}$ as long as $(s^{(i)})^2$ is decreasing.

3.2 Calibration RT – Sampling randomly in time

An alternative way of choosing the validation set is to sample $[\alpha(n-p)]$ rows of $\boldsymbol{K}$ *randomly*. This method enables us to draw the validation data several times and averaging over the models calibrated to each split into training data and validation data, thus allowing us to utilise data better. An

ensemble model formed in such a way will be trained on the whole training set, given that each observation in the training set is contained in the training examples used as input for the calibration of at least one individual model, see further Section 3.5.

This somewhat unorthodox procedure means that the validation set and the training set will be dependent, since one row of $\boldsymbol{K}$ drawn to be included in the validation set will likely contain observations that are also in the training set. As shown in [2] this type of procedure can still work well in the context of cross-validation for general autoregressive models. The motivation is that $\widehat{\epsilon}_t = \widehat{\kappa}_t - f(\boldsymbol{x}_t; \widehat{\boldsymbol{\eta}})$ are uncorrelated, provided that $f(\boldsymbol{x}_t; \boldsymbol{\eta})$ is estimated appropriately. That using this procedure for creating an ensemble model also tends to work well within our setting is supported empirically by the results in Section 5.

The calibration procedure for one calibration follows steps (i)-(iii) in the previous section, with $\mathcal{V}$ consisting of the set of row indices of matrix $\boldsymbol{K}$ that were sampled randomly.

3.3 Calibration SP – Splitting the population by sampling individuals randomly.

Compared with many other time series data situations, such as e.g. stock indices, mortality data is based on an underlying population of individuals, which may be split into sub-populations. That is, by uniformly at random assigning individuals into, e.g. either of two cohorts at birth, the resulting sub-populations should consist of i.i.d. samples from the same underlying distribution. When it comes to mortality data this means that entire individual life-histories are assigned to different groups.

Consequently, instead of splitting data in the time dimension, using one part of the data for training and the other part for validation, we can split the population in two parts. In the latter split the entire observed time interval is used for training and validation, but based on different sets of individuals. This approach is particularly interesting in the situation where we have a sufficiently large underlying population, but where the observed time interval is short, which is a situation that is relevant for e.g. larger insurance companies. Furthermore, it enables us to construct bootstrapped samples of the original population, which makes it possible to form an ensemble model using *bagging*, see e.g. [3], [13, Ch. 8.7], something that is in general not possible for time series data, since we would normally only have access to one

realisation from the underlying stochastic process. This is discussed further in Section 3.5.

For model (4) the calibrating procedure based on a single split is focused on the $\widehat{\kappa}_t$s:

Creating calibration data.

(i) Split the total population into two subpopulations producing the two datasets $(D_{x,t}, r_{x,t})$ and $(D^*_{x,t}, r^*_{x,t}), x \in \mathcal{X}, t \in \mathcal{I}$.

(ii) Calculate $\widehat{\boldsymbol{\theta}} = (\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\kappa}})$ based on $(D_{x,t}, r_{x,t})$, and calculate $\widehat{\boldsymbol{\theta}}^*$ based on $(D^*_{x,t}, r^*_{x,t})$.

Model calibration.

(iii) Let $\widehat{\boldsymbol{\eta}}^{(i)}$ denote the estimate of $\boldsymbol{\eta}$ from $f(\boldsymbol{x}_t; \boldsymbol{\eta}), t \in \mathcal{I}$, of model (4) in the ith calibration epoch.

(iv) Calculate the prediction error

$$(s^{(i)})^2 = \frac{1}{|\mathcal{V}|} \sum_{t \in \mathcal{V}} (\widehat{\kappa}^*_t - f(\boldsymbol{x}^*_t; \widehat{\boldsymbol{\eta}}^{(i)}))^2 \tag{8}$$

where $\boldsymbol{x}^*_t = (\widehat{\kappa}^*_{t-p}, \ldots, \widehat{\kappa}^*_{t-1})^\top$, $\mathcal{V} = \mathcal{I}$, and continue the updating procedure of $\widehat{\boldsymbol{\eta}}^{(i)}$ as long as $(s^{(i)})^2$ is decreasing.

Remark 3. Note that for this calibration procedure the full parameter vector $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\kappa})$ will be re-estimated for the two populations. Since $\widehat{\boldsymbol{\theta}}$ and $\widehat{\boldsymbol{\theta}}^*$ are estimated based on two independent subpopulations for the whole training period $\mathcal{I}$, $\widehat{\boldsymbol{\eta}}^{(i)}$ will be independent of $\widehat{\boldsymbol{\kappa}}^*$, and thus $(\boldsymbol{x}^*_t, \widehat{\kappa}^*_t)$ for $t \in \mathcal{I}$.

3.4 Early stopping rule

If we consider an estimate of $\boldsymbol{\eta}$, without stressing which epoch it is related to, and let

$$e_t := \widehat{\kappa}^*_t - f(\boldsymbol{x}^*_t; \widehat{\boldsymbol{\eta}}), \quad t \in \mathcal{V}$$

it follows that (7) and (8) can be expressed as

$$s^2 = \frac{1}{|\mathcal{V}|} \sum_{t \in \mathcal{V}} e_t^2. \tag{9}$$

Further,

$$\mathbb{E}[e_t^2 \mid \boldsymbol{x}_t^*] = \sigma^2 + \text{Bias}_t^2(\boldsymbol{\eta}, \sigma^2; \boldsymbol{x}_t^*) + \text{Var}_t(\boldsymbol{\eta}, \sigma^2; \boldsymbol{x}_t^*)$$

where

$$\text{Bias}_t^2(\boldsymbol{\eta}, \sigma^2; \boldsymbol{x}_t^*) = \mathbb{E}[(f(\boldsymbol{x}_t^*; \boldsymbol{\eta}) - \mathbb{E}[f(\boldsymbol{x}_t^*; \widehat{\boldsymbol{\eta}})])^2 \mid \boldsymbol{x}_t^*],$$
$$\text{Var}_t(\boldsymbol{\eta}, \sigma^2; \boldsymbol{x}_t^*) = \mathbb{E}[(\mathbb{E}[f(\boldsymbol{x}_t^*; \widehat{\boldsymbol{\eta}})] - f(\boldsymbol{x}_t^*; \widehat{\boldsymbol{\eta}}))^2 \mid \boldsymbol{x}_t^*],$$

which gives us that

$$\begin{aligned}\mathbb{E}[s^2] &= \sigma^2 + \frac{1}{|\mathcal{V}|}\sum_{t\in\mathcal{V}} \mathbb{E}[\text{Bias}_t^2(\boldsymbol{\eta}, \sigma^2; \boldsymbol{x}_t^*)] + \frac{1}{|\mathcal{V}|}\sum_{t\in\mathcal{V}} \mathbb{E}[\text{Var}_t(\boldsymbol{\eta}, \sigma^2; \boldsymbol{x}_t^*) \\ &= \sigma^2 + \overline{\text{Bias}}^2(\boldsymbol{\eta}, \sigma^2) + \overline{\text{Var}}(\boldsymbol{\eta}, \sigma^2).\end{aligned}$$

When using early stopping, training will be stopped after the number of epochs at which s^2 is minimised. Hence using early stopping corresponds to finding the optimal balance between the the prediction bias and the variance.

3.5 Model aggregation

It has long been known that model aggregation, or ensembling, can improve the accuracy of predictions in both classification and regression problems, se e.g. [23, 9, 20]. Hence, to get more stable predictions, we create an ensemble model, by aggregating over m model calibrations. If $\widehat{\boldsymbol{\eta}}_0^{(j)}$ is the optimally stopped estimate of $\boldsymbol{\eta}$ in model calibration j, the aggregated predictor is defined as

$$\bar{f}\Big(\boldsymbol{x}_t; (\widehat{\boldsymbol{\eta}}_0^{(j)})_{j=1}^m\Big) := \frac{1}{m}\sum_{j=1}^m f(\boldsymbol{x}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}). \tag{10}$$

For a single time point t, the prediction error of the ensemble model will always be less than or equal to the average of the prediction errors of the individual models. Let $(\widetilde{\kappa}_t)_{t=1}^l$ denote observations from the test data (future data), and let $\widetilde{\boldsymbol{x}}_t = (\widetilde{\kappa}_{t-1}, \ldots, \widetilde{\kappa}_{t-p})^\top$. Define $\bar{\kappa}_t := \bar{f}\Big(\widetilde{\boldsymbol{x}}_t; (\widehat{\boldsymbol{\eta}}_0^{(j)})_{j=1}^m\Big)$. Then, for a single time point t,

$$\begin{aligned}\frac{1}{m}\sum_{j=1}^m \mathbb{E}[(\widetilde{\kappa}_t - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}))^2 \mid \widetilde{\boldsymbol{x}}_t] &= \mathbb{E}\Big[\frac{1}{m}\sum_{j=1}^m \Big(\bar{\kappa}_t - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}) + \widetilde{\kappa}_t - \bar{\kappa}_t\Big)^2 \mid \widetilde{\boldsymbol{x}}_t\Big] \\ &= \frac{1}{m}\sum_{j=1}^m \mathbb{E}[(\bar{\kappa}_t - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}))^2 \mid \widetilde{\boldsymbol{x}}_t] \\ &\quad + \mathbb{E}[(\widetilde{\kappa}_t - \bar{\kappa}_t)^2 \mid \widetilde{\boldsymbol{x}}_t] \\ &\geq \mathbb{E}[(\widetilde{\kappa}_t - \bar{\kappa}_t)^2 \mid \widetilde{\boldsymbol{x}}_t].\end{aligned}$$

Furthermore, as shown in [23], if the error terms $e_t^{(i)}$ and $e_t^{(j)}$ are uncorrelated for $i \neq j$, then

$$\mathbb{E}[(\widetilde{\kappa}_t - \bar{\kappa}_t)^2 \mid \widetilde{\boldsymbol{x}}_t] = \frac{1}{m^2} \sum_{j=1}^{m} \mathbb{E}[(\widetilde{\kappa}_t - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}))^2 \mid \widetilde{\boldsymbol{x}}_t],$$

i.e. the prediction error is reduced by a factor $1/m$ when ensembling as compared to the average of the prediction errors of the individual models. If the error terms of the individual models are perfectly correlated and have the same variance, then there is no gain from ensembling, since then

$$\mathbb{E}[(\widetilde{\kappa}_t - \bar{\kappa}_t)^2 \mid \widetilde{\boldsymbol{x}}_t] = \mathbb{E}[(\widetilde{\kappa}_t - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(1)}))^2 \mid \widetilde{\boldsymbol{x}}_t].$$

Hence, when constructing an ensemble model, we would like the individual models to be diverse, in the sense that the correlations between the error terms of the models are low. There are several different methods for constructing ensembles, see e.g. [9]. We focus on the methods of injecting randomness and manipulating training examples, and our choice will depend on the calibration method used, since manipulating training examples is not straightforward for all calibration methods.

Calibration LO – using the last fraction of observations as validation data. We construct the ensemble by injecting randomness into the learning algorithm. When using neural network regression models, each time the model is calibrated we will get a slightly different prediction due to the random initialisation of the calibration, and due to using stochastic gradient descent. However, each individual model will use the same training and validation data. Hence, for this case each parameter estimate $\widehat{\boldsymbol{\eta}}_0^{(j)}$ in the aggregated predictor in (10) has been determined according to steps (i) - (ii) in Section 3.1, over the same sets $\mathcal{T}$ and $\mathcal{V}$. In [25] this is what is called the nagging predictor, from combining networks and aggregating, as opposed to bagging, combined bootstrapping and aggregation.

Calibration RT – sampling the validation data randomly in time. We will combine the method of injecting randomness through the random initialisation of the calibration, and using stochastic gradient descent, with manipulating training examples. In each run this is done by drawing a new sample for the validation data. Hence, for each model calibration, the set $\mathcal{V}^{(j)}$ consists of a random sample of row indices of matrix $\boldsymbol{K}$, and $\widehat{\boldsymbol{\eta}}_0^{(j)}$, the optimally stopped estimate of $\boldsymbol{\eta}$ for model calibration j, will depend both on the random initialisation of the calibration, and the random split of the row indices of matrix $\boldsymbol{K}$ into $\mathcal{V}^{(j)}$ and $\mathcal{T}^{(j)} := \mathcal{I} \setminus \mathcal{V}^{(j)}$. By both injecting randomness and manipulating training examples, we make the individual models

more diverse. Furthermore, by randomly drawing a different validation set for each model we can utilise data better in the sense that the ensemble model will have used most observations both for training and validation.

Calibration SP – creating validation data by splitting the population by sampling individuals. We again combine injecting randomness via random initialisation with manipulating training examples, but in a slightly different way as compared to the second calibration method. Since we here sample individuals randomly, we are able to manipulate training examples through bootstrapping, which is not straightforward for the other two calibration methods where the training data is split in the time dimension. Hence our ensemble model in this case will be a combination of injecting randomness via random initialisation and population bagging, see [3]. Thus, instead of splitting the total population into two subpopulations, we sample individuals at random with replacement of the same population size as the original population, and then split this bootstrapped population into two subpopulations. This can also be combined with subsampling, where the sampled number of individuals is less than the original population size. This strategy can be used to prevent overfitting for the case when the total population is very large, thus essentially leading to $\widehat{\boldsymbol{\kappa}}$ and $\widehat{\boldsymbol{\kappa}}^*$ being equal if using the original population (or a bootstrapped population of the same size) as a starting point when making the population split. Furthermore, subsampling also has the benefit of creating more diverse models, since the training sets used for each individual model will be less similar when using subsampling for large populations. Examples of subsampling are given in Section 5.

Variance parameter estimation and simulation of ensemble models. For repeated one-step simulations of future data, assuming having an estimate of variance parameter, the simulated future outcome from the ensemble model at time step $t-1$ is used as an observation when predicting the value at time t. Hence, for the jth trajectory and $t > n$,

$$\widetilde{\kappa}_t^{(j)} = \bar{f}\Big((\widetilde{\boldsymbol{x}}_t^{(j)}); (\widehat{\boldsymbol{\eta}}_0^{(i)})_{i=1}^m\Big) + \epsilon_t,$$

where $\epsilon_t \sim \mathsf{N}(0, \sigma_{\text{ens}}^2)$ and i.i.d., and

$$\begin{aligned}
\widetilde{\boldsymbol{x}}_{n+1}^{(j)} &= \boldsymbol{x}_{n+1} = (\widehat{\kappa}_n, \ldots, \widehat{\kappa}_{n+1-p})^\top \\
\widetilde{\boldsymbol{x}}_{n+2}^{(j)} &= (\widetilde{\kappa}_{n+1}^{(j)}, \widehat{\kappa}_n \ldots, \widehat{\kappa}_{n+1-p})^\top \\
&\vdots \\
\widetilde{\boldsymbol{x}}_{n+p}^{(j)} &= (\widetilde{\kappa}_{n+p-1}^{(j)}, \ldots, \widetilde{\kappa}_{n+1}^{(j)}, \widehat{\kappa}_n)^\top
\end{aligned}$$

and $\widetilde{\boldsymbol{x}}_t^{(j)} = (\widetilde{\kappa}_{t-1}^{(j)}, \ldots, \widetilde{\kappa}_{t-p}^{(j)})^\top$ for $t > n + p$. Finally, the prediction of the ensemble model at time step t over N simulated trajectories is given by the median of $(\bar{f}\Big((\widetilde{\boldsymbol{x}}_t^{(j)}); (\widehat{\boldsymbol{\eta}}_0^{(i)})_{i=1}^m\Big)_{j=1}^N$, and in a similar manner prediction intervals can be constructed. Note that since the estimated predictor $\widehat{f}$ in general will be highly non-linear, it is important not to make predictions by directly inserting $\widehat{\boldsymbol{x}}_t$ corresponding to expected values into f.

Concerning the estimation of σ_{ens}^2, the natural estimator is to use the in-sample variance $\bar{s}^2$:

$$\bar{s}^2 = \frac{1}{|\mathcal{I}|} \sum_{t \in \mathcal{I}} \Big(\widehat{\kappa}_t - \bar{f}\Big(\boldsymbol{x}_t; (\widehat{\boldsymbol{\eta}}_0^{(j)})_{j=1}^m\Big)\Big)^2.$$

Similarly as when looking at the prediction error, let $\hat{\bar{f}}(\boldsymbol{x}_t) := \bar{f}\Big(\boldsymbol{x}_t; (\widehat{\boldsymbol{\eta}}_0^{(j)})_{j=1}^m\Big)$, and it follows that

$$\begin{aligned}
\frac{1}{m} \sum_{j=1}^m \Big(\widehat{\kappa}_t - f(\boldsymbol{x}_t, \widehat{\boldsymbol{\eta}}_0^{(j)}\Big)^2 &= \frac{1}{m} \sum_{j=1}^m \Big(\widehat{\kappa}_t - \hat{\bar{f}}(\boldsymbol{x}_t) + \hat{\bar{f}}(\boldsymbol{x}_t) - f(\boldsymbol{x}_t, \widehat{\boldsymbol{\eta}}_0^{(j)}\Big)^2 \\
&= (\widehat{\kappa}_t - \hat{\bar{f}}(\boldsymbol{x}_t))^2 + \frac{1}{m} \sum_{j=1}^m (\hat{\bar{f}} - f(\boldsymbol{x}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}))^2 \\
&\geq (\widehat{\kappa}_t - \hat{\bar{f}}(\boldsymbol{x}_t))^2,
\end{aligned}$$

hence

$$(\bar{s})^2 = \frac{1}{|\mathcal{I}|} \sum_{t \in \mathcal{I}} (\widehat{\kappa}_t - \hat{\bar{f}}(\boldsymbol{x}_t))^2 \leq \frac{1}{m} \sum_{j=1}^m (s^{(j)})^2$$

where $(s^{(j)})^2$ is the in-sample variance for model calibration j:

$$(s^{(j)})^2 = \frac{1}{|\mathcal{I}|} \sum_{t \in \mathcal{I}} (\widehat{\kappa}_t - f(\boldsymbol{x}_t, \widehat{\boldsymbol{\eta}}_0^{(j)})^2.$$

Moreover, similarly to Section 3.4, although defined in-sample, one can note that $\mathbb{E}[(\bar{s})^2]$ is equal to σ^2, here referring to the σ^2 from (4), plus a (reducible) error part.

3.6 Relating calibration RT to calibration LO

When creating the validation data by withholding the last fraction of observations, we will have the same in-sample training data and out-of-sample

validation data for each of the individual models that make up the ensemble model. If we disregard any differences of the models due to the random initialisation of the calibration, each model will give the same estimate $\widehat{\boldsymbol{\eta}}_0^{\mathrm{LO}}$ of $\boldsymbol{\eta}$, where LO, as above, refers to the Calibration LO ("last observations"). Hence, the ensemble model in (10) using this calibration method becomes

$$\widehat{\bar{f}}_{\mathrm{LO}}(\boldsymbol{x}_t) := \bar{f}(\boldsymbol{x}_t; \widehat{\boldsymbol{\eta}}_0^{\mathrm{LO}}) = f(\boldsymbol{x}_t; \widehat{\boldsymbol{\eta}}_0^{\mathrm{LO}}).$$

When we create the validation data by sampling observations randomly in time, we get the estimates $\widehat{\boldsymbol{\eta}}_0^{(j)}$, $j = 1, \ldots, m$ of $\boldsymbol{\eta}$, and the ensemble model is given by (10),

$$\widehat{\bar{f}}_{\mathrm{RT}}(\boldsymbol{x}_t) := \bar{f}(\boldsymbol{x}_t; (\widehat{\boldsymbol{\eta}}_0^{(j)})_{j=1}^m),$$

where RT refers to Calibration RT ("random time").

If $\mathbb{E}[(\widetilde{\kappa}_t - \widehat{\bar{f}}_{\mathrm{LO}}(\widetilde{\boldsymbol{x}}_t))^2 \mid \widetilde{\boldsymbol{x}}_t] \geq \mathbb{E}[(\widetilde{\kappa}_t - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}))^2 \mid \widetilde{\boldsymbol{x}}_t]$ for $j = 1, \ldots, m$, where $(\widetilde{\kappa}_t, \widetilde{\boldsymbol{x}}_t)_{t=1}^l$ is unseen future data, then

$$\begin{aligned}\mathbb{E}[(\widetilde{\kappa}_t - \widehat{\bar{f}}_{\mathrm{LO}}(\widetilde{\boldsymbol{x}}_t))^2 \mid \widetilde{\boldsymbol{x}}_t] &\geq \frac{1}{m}\sum_{j=1}^m \mathbb{E}[(\widetilde{\kappa}_t - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}))^2 \mid \widetilde{\boldsymbol{x}}_t] \\ &\geq \mathbb{E}[(\widetilde{\kappa}_t - \widehat{\bar{f}}_{\mathrm{RT}}(\widetilde{\boldsymbol{x}}_t))^2 \mid \widetilde{\boldsymbol{x}}_t].\end{aligned}$$

Conversely, if $\mathbb{E}[(\widetilde{\kappa}_t - \widehat{\bar{f}}_{\mathrm{LO}}(\widetilde{\boldsymbol{x}}_t))^2 \mid \widetilde{\boldsymbol{x}}_t] \leq \mathbb{E}[(\widetilde{\kappa}_t - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}))^2 \mid \widetilde{\boldsymbol{x}}_t]$ for $j = 1, \ldots, m$, then

$$\begin{aligned}\mathbb{E}[(\widetilde{\kappa}_t - \widehat{\bar{f}}_{\mathrm{LO}}(\widetilde{\boldsymbol{x}}_t))^2 \mid \widetilde{\boldsymbol{x}}_t] &\leq \frac{1}{m}\sum_{j=1}^m \mathbb{E}[(\widetilde{\kappa}_t - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}))^2 \mid \widetilde{\boldsymbol{x}}_t] \\ &= \frac{1}{m}\sum_{j=1}^m \mathbb{E}[(\widehat{\bar{f}}_{\mathrm{RT}}(\widetilde{\boldsymbol{x}}_t) - f(\widetilde{\boldsymbol{x}}_t; \widehat{\boldsymbol{\eta}}_0^{(j)}))^2 \mid \widetilde{\boldsymbol{x}}_t] \\ &\quad + \mathbb{E}[(\widetilde{\kappa}_t - \widehat{\bar{f}}_{\mathrm{RT}}(\widetilde{\boldsymbol{x}}_t))^2 \mid \widetilde{\boldsymbol{x}}_t].\end{aligned}$$

Hence, if $\widehat{\bar{f}}_{\mathrm{LO}}(\cdot)$ is the worst individual model, in the sense that this model achieves the largest out-of-sample error, then $\widehat{\bar{f}}_{\mathrm{RT}}(\cdot)$ will be better. If, on the other hand, $\widehat{\bar{f}}_{\mathrm{LO}}(\cdot)$ is the best individual model, i.e. achieves the smallest out-of-sample error, it is still not guaranteed to be better than the ensemble model $\widehat{\bar{f}}_{\mathrm{RT}}(\cdot)$, since this will depend on how much better it is compared to the individual models that make up the ensemble model $\widehat{\bar{f}}_{\mathrm{RT}}(\cdot)$, as well as how large the correlation is between the error terms of the individual models. To conclude, unless there is reason to believe that $\widehat{\bar{f}}_{\mathrm{LO}}(\cdot)$ will produce a substantially better model than the models based on sampling the validation set randomly in time, essentially "putting all eggs in one basket", a more agnostic alternative is to use the ensemble model $\widehat{\bar{f}}_{\mathrm{RT}}(\cdot)$.

3.7 LSTM boosting

Regardless of how data is split and used for model calibration / assessment, once the amount of data being used is reduced, it becomes even more important to have good starting values for the $\widehat{\kappa}_t$-process. The idea with boosting is as follows:

(i) Decide on a reference time series model for the $\widehat{\kappa}_t$s, thinking of this as a "standard" time series model, e.g. an ARIMA model. Let $h(\mathcal{F}_{t-1};\boldsymbol{\xi})$ denote the mean-function of the reference model, i.e.

$$\widehat{\kappa}_{t+1} := h(\mathcal{F}_t;\boldsymbol{\xi}) + \widetilde{\epsilon}_{t+1},$$

where $\widetilde{\epsilon}_t \sim \mathsf{N}(0,\chi^2)$ and i.i.d.

(ii) Obtain an estimate $\widehat{\boldsymbol{\xi}}$ of $\boldsymbol{\xi}$.

(iii) Given $\widehat{\boldsymbol{\xi}}$, introduce the version of the LSTM neural network model (4) defined by

$$\widehat{\kappa}_{t+1} = h(\mathcal{F}_t;\widehat{\boldsymbol{\xi}}) + f(\mathcal{F}_t;\boldsymbol{\eta}) + \epsilon_{t+1}, \tag{11}$$

where $\epsilon_t \sim \mathsf{N}(0,\sigma^2)$ and i.i.d., and where $h(\mathcal{F}_t;\widehat{\boldsymbol{\xi}})$ acts like an $\mathcal{F}_t$-measurable (non-trainable) intercept function in the LSTM model.

The above boosting procedure is exemplified for model (4), but the steps hold verbatim for generalisations of this model as well.

4 Likelihoods and performance measures

As discussed in the introduction, the starting point for the modelling is the Poisson Lee-Carter model from [5], see (2) and (3), whose log-likehood is given by

$$l(\boldsymbol{\theta}) = \sum_{x,t}(-r_{x,t}\exp\{\alpha_x+\beta_x\kappa_t\} + d_{x,t}(\alpha_x+\beta_x\kappa_t)), \tag{12}$$

which gives us the MLE of $\boldsymbol{\theta}$.

Next, if we introduce randomness by treating the $\widehat{\kappa}_t$s as outcomes of a stochastic process, whose parameters are contained in $\boldsymbol{\eta}$ together with the relevant

filtration given by the $\mathcal{F}_t$s, it follows that the *complete data likelihood*, when treating the estimated κ_ts as observations, is given by

$$L(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}) = \prod_{x,t} p(d_{x,t} \mid r_{x,t}; \alpha_x, \beta_x, \widehat{\kappa}_t) q(\widehat{\kappa}_t; \mathcal{F}_{t-1}, \boldsymbol{\eta}), \tag{13}$$

where $p(\cdot)$ corresponds to the probability mass function of the Poisson distribution and where $q(\cdot)$ corresponds to Gaussian densities.

The corresponding *incomplete data likelihood*, when we only observe death counts, is given by

$$L^*(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbb{E}_{\boldsymbol{\kappa}} \left[\prod_{x,t} p(d_{x,t} \mid r_{x,t}; \alpha_x, \beta_x, \widehat{\kappa}_t) \right], \tag{14}$$

where $\mathbb{E}_{\boldsymbol{\kappa}}[\cdot]$ corresponds to the expectation taken over the joint distribution of the $\widehat{\kappa}_t$s. Thus, by simulating $\widehat{\kappa}_t$s from $q(\cdot; \mathcal{F}_{t-1}, \widehat{\boldsymbol{\eta}})$ it is possible to estimate $L^*(\boldsymbol{\alpha}, \boldsymbol{\beta})$, which, after taking the logarithm, is comparable with (12).

However, when computing (14) in practice, using the average over all trajectories will often turn out to be numerically unstable. The reason for this is that by sampling $\widehat{\kappa}_t$ trajectories without conditioning on the observed death counts, only a small fraction of trajectories will be in reasonable agreement with the deaths actually observed. Due to this the contribution to the likelihood will for most simulated trajectories be zero, hence leading to a very unstable estimate of (14). Phrased differently, this procedure can be thought of as a very naive Monte Carlo importance sampling procedure, where the sampling weights are uniform, leading to a low effective number of sampled trajectories. Because of this we instead evaluate different models in Section 5 based on the median of incomplete likelihoods per trajectory given by

$$\widetilde{L}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \text{median}_{\boldsymbol{\kappa}} \left[\prod_{x,t} p(d_{x,t} \mid r_{x,t}; \alpha_x, \beta_x, \widehat{\kappa}_t) \right], \tag{15}$$

which will indicate the typical performance of the κ_t-models. Moreover, note that (15) is

(i) computable for any $\widehat{\kappa}_t$-model, and that it is possible to compute both in-sample and out-of-sample, given the $r_{x,t}$s,

(ii) smaller or equal to the corresponding likelihood consistent with (12), with equality if and only if the $\widehat{\kappa}_t$ process is degenerate putting all probability mass in the points corresponding to the $\widehat{\kappa}_t$s from the MLE of $\boldsymbol{\theta}$.

Remark 4. In Section 5 we also plot the incomplete data log-likelihood per age x, and per time t. For age x we define

$$L_x^*(\alpha_x, \beta_x) = \mathbb{E}_{\boldsymbol{\kappa}}\left[\prod_t p(d_{x,t} \mid r_{x,t}; \alpha_x, \beta_x, \widehat{\kappa}_t)\right],$$

and plot $\log L_x^*(\alpha_x, \beta_x)$ as a function of x, and similarly for each time point t. However

$$\sum_x \log L_x^*(\alpha_x, \beta_x) \neq \log L^*(\boldsymbol{\alpha}, \boldsymbol{\beta}),$$

hence this should only be seen as an indication of for which ages and time points the fit is better or worse for each model and does not completely align with the log-likelihood defined by (14). Due to this it is not possible to ascertain that the incomplete log-likelihood marginalised w.r.t. time or age should be lower than the corresponding saturated log-likelihood. This is also the case when changing from (14) to (15). However, when comparing the log-likelihood defined by (14) for each model, this will never exceed the log-likelihood for the saturated model.

Henceforth, we call the model with estimates of κ_t corresponding to the $\widehat{\kappa}_t$s from the MLE of $\boldsymbol{\theta}$ the saturated model. This is since this choice of κ_t corresponds to perfect fit within our model structure, where $\widehat{\boldsymbol{\theta}} = (\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\kappa}})$ are seen as given, and we want to find a suitable model for the $\widehat{\kappa}_t$s in order to forecast them into the future. Hence, the $\widehat{\kappa}_t$s correspond to the best fit we can achieve based on observed death count data, given the model structure and the estimates $(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}})$. Furthermore, for the prediction period we define the saturated model as the one with estimates of κ_t corresponding to the MLE of κ_t based on mortality data for the prediction period, given $(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{\beta}})$ estimated for the in-sample period. Hence, we maximise the log-likelihood in (12) over $\boldsymbol{\kappa}$ for the prediction period, using the previous estimates of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Thus, if we could look into the future, but were restricted by our model choice and the previous estimates of $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, these estimates of the κ_ts give the best possible fit to data.

Note, however, that even if we may compute (15) based on observed death counts, we still only use the inner Gaussian likelihood when optimising $\boldsymbol{\eta}$ in the $\widehat{\kappa}_t$-model, i.e. by minimising the (validation) MSE. This is in line with e.g. [21]. Naturally, due to this it is not necessarily the best $\widehat{\kappa}_t$-model chosen w.r.t. the minimal (validation) MSE that will maximise the incomplete data likelihood (15). In Section 5 we give an example of when this occurs.

Remark 5 (MSE for the different calibration approaches)**.** The validation MSEs for the three different calibration approaches are not comparable with each other, since the validation sets are different. This is especially true for calibration SP compared to the other two approaches, since for this approach the validation set consists of the same number of observations as the full training set, due to not splitting the observations in the time dimension. Thus, it is only the test MSEs that are comparable between the calibration approaches. Still, the validation MSE can, of course, be used when comparing different models that are based on the same calibration approach, e.g. models with different number of neurons in the LSTM layer or using different activation functions.

5 Numerical illustrations

We will now illustrate the methods for calibration based on data from the [15] for Italy, Sweden, and the USA. As mentioned early on, the ambition in the current section is not to obtain optimal model architectures, but rather to find architectures that work reasonably well for all populations. Moreover, the numerical illustrations are only used to highlight certain observations and artefacts of the methods used, but more details can be found in the Supplementary Materials [18] (available online).

Concerning estimation, the first step is to estimate all $(\alpha_x, \beta_x)_x$ and κ_ts using the Poisson Lee-Carter model defined by (2) and (3) using the `R` package `StMoMo`, see [27]. Given these initial estimates, the $\widehat{\kappa}_t$s are in a second step modelled as a univariate Gaussian LSTM model defined by (4) using the `R` package `keras`, see [6].

The structure of the numerical illustrations is as follows:

Base case. The base case is to use death count data from 1950 - 1999 for training and to use 2000 - 2016 for out-of-sample testing. That is, the data from 1950 - 1999 will be split into in-sample training and validation sets in different ways depending on the different calibration procedures (i.e. LO, RT, or SP). The reason for not using data older than 1950 is in order to avoid the influence of WWII. Moreover, the assumption of having $(\alpha_x, \beta_x)_x$s independent of time will provide a poor model fit when using longer time periods. This is due to structural breaks in data.

Long-term predictions. After having analysed the base case, which focus on short to medium range predictions, we move on to analysing the influence

of the calibrations on long term predictions. In this situation the focus is on the predictions themselves, since we lack suitable test data.

Calibration using a limited amount of data. The last part focus on the situation when having small amounts of data to be used for calibration. The data periods used are 1970 - 1989 for training together with 1990 - 2006 for testing, and 1980 - 1999 for training together with 2000 - 2016 for testing.

Before going into the numerical examples, we start with discussing a number of considerations necessary for the LSTM implementation and the calibration procedures.

5.1 Comments on architecture and implementation

We will now discuss a number of 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 [6] 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.

5.1.1 Lags

Lag 5 is used for all methods and time periods. In [21] lag 1 is used, but this choice of lag does not fully exploit the properties of an LSTM model, as discussed in Section 2. 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. [24, 22].

5.1.2 Activation function

As a starting point the ReLu activation function is used, which is in accordance with [21]. 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.

5.1.3 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.

5.1.4 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.

5.1.5 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 all 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.

5.1.6 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 expect 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. The choice of using 20 calibrations is perhaps a bit low for Calibration SP, but as will be seen below, the results for this calibration method will still be satisfactory. For more on how to decide on the number of calibrations in each ensemble, see the Supplementary Materials [18]. In a non-life insurance setting, [25] saw stability after about 20 calibrations. Within the context of mortality modelling, [22] settled on 10 calibrations for their ensemble model.

5.1.7 Data pre-processing

As described in Section 5.1.2, 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. [13, Ch. 11.5.3]. However, we consider this form of scaling of data as inappropriate to use unless the trend is has been removed from data first, since otherwise the predicted values will tend to consistently lie outside of the interval $[-1, 1]$.

5.1.8 Subsampling

For Calibration SP ("split population") 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 testing. 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 [11, Ch. 8.3] for an overview, and see [6] 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. [11, Ch. 7.12], hence it is possible that the network performance could be improved further.

N.B. All model parameters are summarised in Appendix A.

5.2 Data analyses and predictive performance

5.2.1 Base case

As discussed in Section 5.1.2 we start by considering the ReLu activation function when using un-scaled data (no pre-processing). Further, since lag 5 is being used, the effective training data consists of the time period 1955 - 1999. In Table 1 the total MSE for the $\widehat{\kappa}_t$ ensemble models used with Calibration LO, SP, and RT are shown, with the MSE for the best performing model of the three marked in bold for each population. As a point of reference a standard Gaussian random walk with drift model (RWD) for the $\widehat{\kappa}_t$ process is used. The MSE for the RWD is underlined for the populations where the RWD outperforms the LSTM models. From Table 1 it is seen that Calibration SP in general outperforms the RWD in the test set. The only exceptions being Swedish females, where the trend is very close to linear, and for USA males and females, where the performance of the RWD is good by chance. That is, the $\widehat{\kappa}_t$ processes for USA females and males exhibit structural breaks that are un-reasonable to capture based on the training data, see Figure 2. Concerning Calibration RT it is seen that it overall performs well. When turning to Calibration LO, this calibration produces the best test MSE for Italian females and males, but the closeness to the RWD for females indicates a quite linear evolution of the $\widehat{\kappa}_t$ process, whereas the dynamics for males is less linear. Still, the performance of the LO calibration in these cases is comparable with those from Calibration RT and SP. On the other hand, it is clear that Calibration LO is considerably worse for Swedish males, indicating that the last observations are not too representative for the future evolution of the $\widehat{\kappa}_t$ process, see Figure 3. The goal with the predictive modelling is of course to forecast mortality rates. One way of doing this is

to take into account not only the variation in the $\widehat{\kappa}_t$ process in (3), but also the Poisson variation in the number of deaths in (2). This is discussed in [1, Eq. (16)] where a two-step procedure is used. In the first step the $\widehat{\kappa}_t$ process is simulated and $\mu_{x,t}$ from (3) is calculated for each trajectory, denoted $\mu^*_{x,t}$, and in the second step the number of deaths $D^*_{x,t}$ are simulated, given $\mu^*_{x,t}$ according to (2). Combining this, the predicted simulated mortality rates are calculated according to

$$\widehat{\mu}^*_{x,t} = \frac{D^*_{x,t}}{r_{x,t}}. \tag{16}$$

Figure 4 shows the predicted simulated mortality rates for age 55 and 85 calculated according to (16), for calibration approach LO and RT. Clearly the predicted mortality rates for calibration LO are far too low, while calibration RT works fairly well.

Further, as discussed in Section 4, the MSE does not provide the full picture. In Table 2 the total log-likelihood based on (15) for test data is summarised together with the saturated model based on the raw estimates of $(\alpha_x, \beta_x)_x$ and $(\kappa_t)_t$. From Table 2 it is seen that the general ordering of the predictive performance of using the different calibrations remain essentially the same. Note, however, that for Swedish females all three LSTM models are better than the Poisson Lee-Carter, whereas the RWD outperformed the LSTM models in terms of MSE in Table 1. This illustrates the importance of assessing the global performance of the model in terms of deaths (or mortality rates), not only focusing on the inner $\widehat{\kappa}_t$ process. The reason for that Calibration SP has a higher log-likelihood than the Poisson Lee-Carter model is explained by that it captures the dynamics in older ages better. This is illustrated for Calibration SP and RT compared to the Poisson Lee-Carter model in Figure 5. See also the simulated predicted mortality rates for Swedish females for age 55 and 85 in Figure 6.

To conclude this far, Calibration RT and SP tend to perform best, and rarely considerably worse than Calibration LO. For Calibration LO we have seen examples where its predictive performance deteriorates, when at the same time Calibration RT and SP produce reasonable predictions.

5.2.2 Long-term predictions

Compared with standard time-series models it is not obvious whether an LSTM model calibration will produce predictions that are "non-explosive", i.e. not tending to $\pm\infty$. In Section 5.2.1 it was seen that the LSTM model

	RWD	LO	RT	SP
ITA male	489.55	**25.37**	49.00	30.12
ITA female	35.65	**19.07**	30.60	25.79
SWE male	459.02	5 008.94	**75.99**	133.12
SWE female	3.31	17.62	21.86	**4.97**
USA male	32.11	104.36	110.37	**36.85**
USA female	10.37	146.86	**127.88**	178.39

Table 1: Out-of-sample MSE (2000-2016) for LSTM-ensemble trained on "raw data" (no pre-processing) with activation function ReLu. Full set of parameters are given in Appendix A.

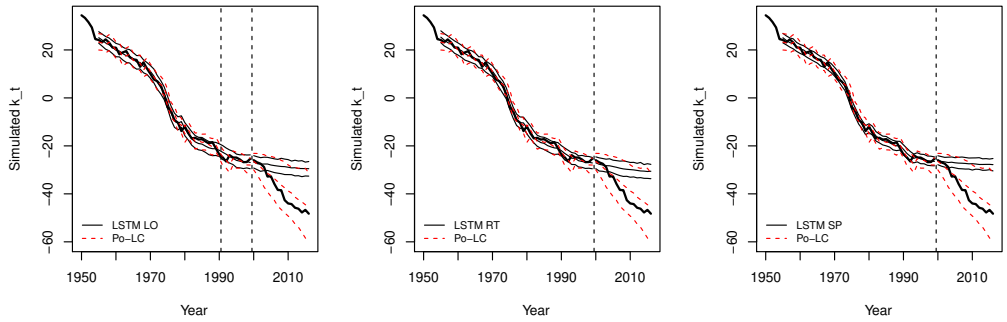

Figure 2: USA females: κ_t in-sample and out-of-sample. Bold solid black line shows raw ("observed") estimates. Thin solid black lines correspond to median, 2.5 and 97.5 percentiles for the LSTM-ensembles trained on "raw data" (no pre-processing) with activation function ReLu. Thin dashed red lines correspond to median and percentiles for the RWD. Data to the right of the last vertical dashed line correspond to test data. For calibration LO, the area between the vertical dashed lines show in-sample validation data.

may be calibrated successfully in order to produce short to medium-term predictions that out-performed an RWD, when only looking at the $\widehat{\kappa}_t$ process, or the Poisson Lee-Carter model when considering actual death counts or mortality rates. As an example, all calibrations, LO, RT, and SP, produced reasonable predictions for Italian males in the short to medium term. This is, however, not the case when pushing the predictions further into the future, see Figure 7, where all calibrations decrease super linearly producing mortality rates that are practically zero – including extremely narrow prediction intervals. This indicates that even if we have used early stopping based on validation data when calibrating the LSTM model, all calibrations seem to have overfitted to non-linearities in the training data.

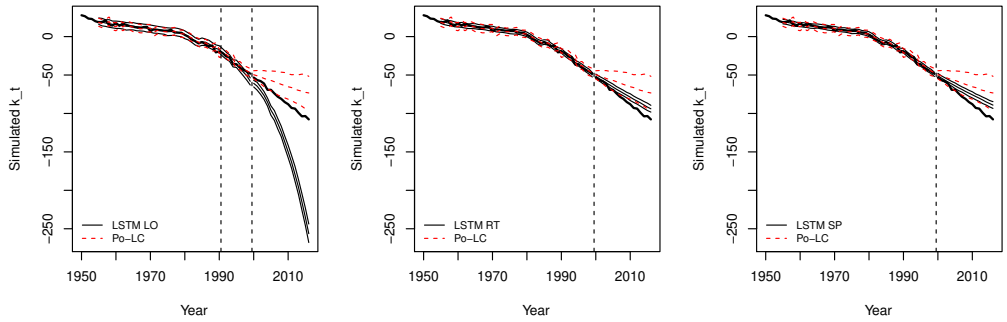

Figure 3: Swedish males: κ_t in-sample and out-of-sample. Bold solid black line shows raw ("observed") estimates. Thin solid black lines correspond to median, 2.5 and 97.5 percentiles for the LSTM-ensembles trained on "raw data" (no pre-processing) with activation function ReLu. Thin dashed red lines correspond to median and percentiles for the RWD. Data to the right of the last vertical dashed line correspond to test data. For calibration LO, the area between the vertical dashed lines show in-sample validation data.

	Saturated	Po-LC	LO	RT	SP
ITA male	-57 996	-105 612	**-60 624**	-62 480	-60 948
ITA female	-13 284	-23 112	**-16 243**	-17 294	-16 477
SWE male	-10 462	-16 183	-57 265	**-11 387**	-12 089
SWE female	-7 116	-8 019	-7 641	-7 762	**-7 312**
USA male	-189 392	-224 054	-290 696	-297 490	**-227 014**
USA female	-60 155	-78 182	-185 238	**-168 599**	-210 614

Table 2: Log-likelihood calculated according to (15) out-of-sample (2000-2016) for the Poisson Lee-Carter model and for the LSTM-ensemble trained on "raw data" (1950-1999) with activation function ReLu.

This dramatic deterioration of the long-term predictions can, at least partly, be diminished by using boosting, as discussed in Section 3.7, and scaling. That is, we first fit an RWD to the original κ_t estimates, and only feed the resulting residuals, scaled to lie between $[-1, 1]$ using the min-max-scaler, to the LSTM. This boosted LSTM model turns out to work best with tanh as activation function, instead of the previously used ReLu activation. The performance of boosted SP calibrations for Italian and Swedish males are given in Figure 8, where it is clearly seen that the boosted models provide a reasonable compromise between the (possibly very) non-linear pure LSTM model and the linear RWD model. Here one can note that when the RWD and pure LSTM are in conflict, the prediction intervals will be wider, see

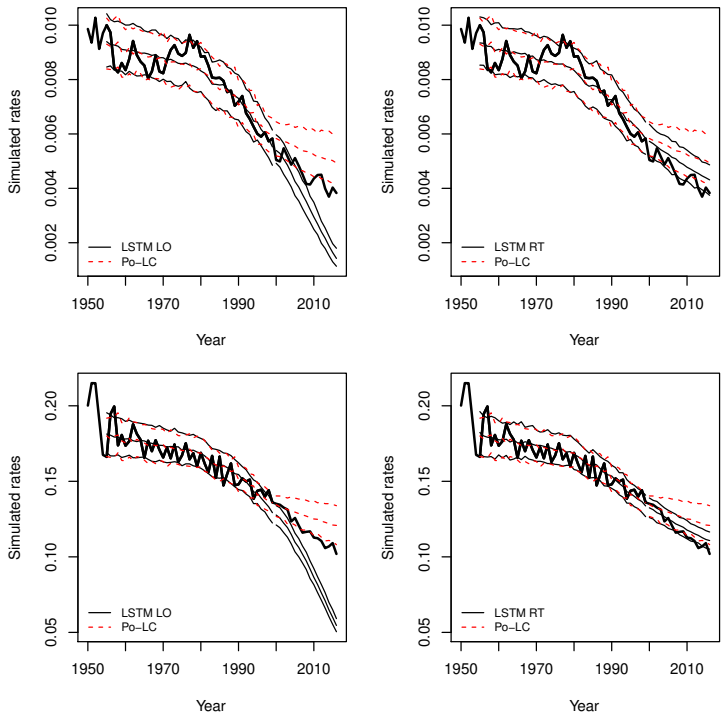

Figure 4: Swedish males: Observed mortality rates 1950-2016, and predicted simulated mortality rates calculated according to (16). Top: Age 55. Bottom: Age 85. Right: Calibration LO. Left: Calibration RT. Bold solid black line shows observed mortality rates. Thin solid black lines correspond to median, 2.5 and 97.5 percentiles for the LSTM-ensembles trained on "raw data" (no pre-processing) with activation function ReLu. Thin dashed red lines correspond to median and percentiles for the Poisson Lee-Carter model.

the analysis for USA women in the Supplementary Materials [18]. Similarly, when the RWD and the pure LSTM are aligned, the prediction intervals may still be narrow, see the analysis for Swedish women in the Supplementary Materials [18].

A summary of test log-likelihoods calculated according to (15) for all populations is given in Table 3, which compared with Table 2 show that the boosted models in general provide good predictive performance.

Before ending this section, it is worth stressing that you can, of course, use another model than a simple RWD as the basis for boosting such as more general ARIMA models. Another simple generalisation is to boost squared residuals, in this way creating an ARCH type LSTM model.

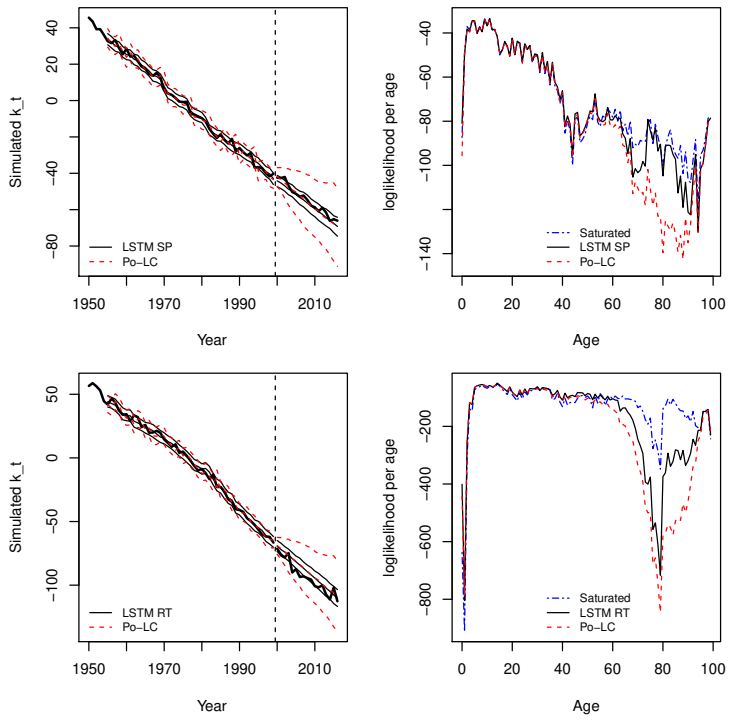

Figure 5: Top: Swedish women. Bottom: Italian women. Left: κ_t in-sample and out-of-sample. Bold solid black line shows raw ("observed") estimates. Thin solid black lines correspond to median, 2.5 and 97.5 percentiles for the LSTM-ensembles trained on "raw data" (no pre-processing) with activation function ReLu. Thin dashed red lines correspond to median and percentiles for the RWD. Data to the right of the last vertical dashed line correspond to test data. Right: log-likelihood per age, calculated in agreement with (15), out-of-sample. The solid black line corresponds to the LSTM-ensembles, the dashed red line corresponds to the Poisson Lee-Carter model, and the dash-dotted blue line corresponds to the saturated model.

The conclusion in the current section is again that Calibration RT and SP tend to outperform the standard LO calibration.

5.2.3 Calibration using a limited amount of data

As already discussed in Section 5.2.2, by using boosting the predictive performance becomes a compromise between a simpler model (here RWD) and a complex non-linear model (here LSTM). This approach tends to stabilise long-term predictions, and if the two model types are in "conflict" the pre-

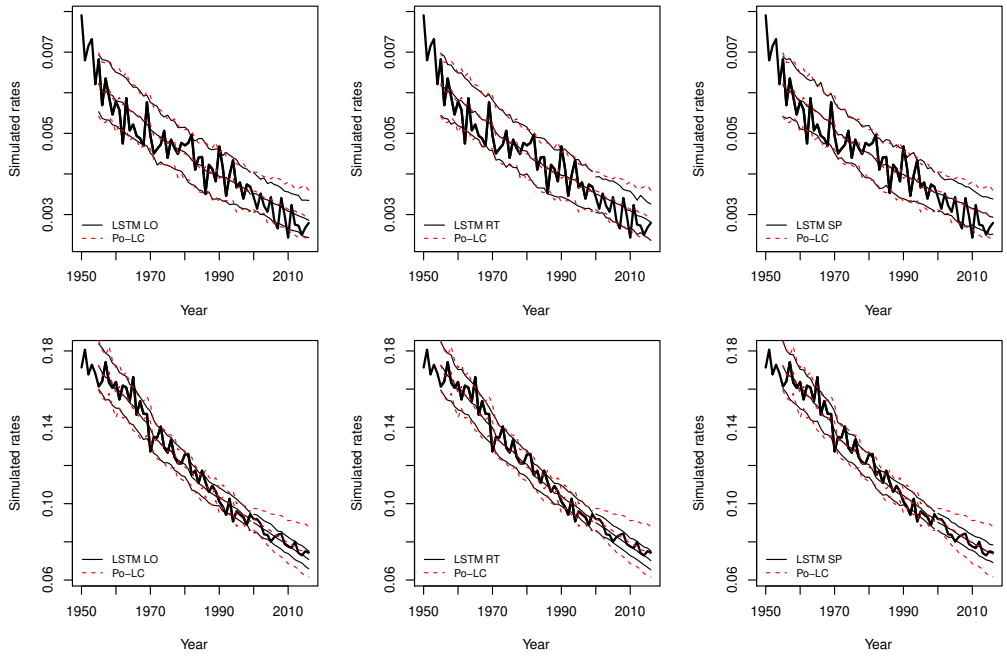

Figure 6: Swedish females: Observed mortality rates 1950-2016, and predicted simulated mortality rates calculated according to (16). Top: age 55. Bottom: age 85. Bold solid black line shows observed mortality rates. Thin solid black lines correspond to median, 2.5 and 97.5 percentiles for the LSTM-ensembles trained on "raw data" (no pre-processing) with activation function ReLu. Thin dashed red lines correspond to median and percentiles for the Poisson Lee-Carter model.

	Saturated	Po-LC	LO	RT	SP
ITA male	-57 996	-105 612	-91 232	-78 431	**-78 139**
ITA female	-13 284	-23 112	-19 012	-17 614	**-16 986**
SWE male	-10 462	-16 183	-13 580	-14 062	**-12 947**
SWE female	-7 116	-8 019	-9 303	**-7 616**	-8 170
USA male	-189 392	-224 054	**-218 720**	-220 277	-236 334
USA female	-60 155	-78 182	-95 775	-85 316	**-82 625**

Table 3: Log-likelihood calculated according to (15) out-of-sample (2000-2016) for the Poisson Lee-Carter model and for the LSTM-ensemble trained on residual after boosting and scaling (1950-1999) with activation function tanh.

diction intervals widens, whereas if they are "aligned" it is possible to still obtain reasonably narrow prediction intervals. Due to this, we only consid-

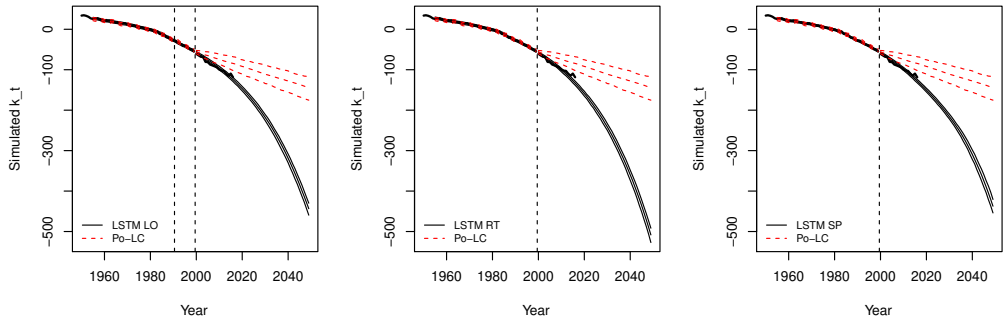

Figure 7: Italian males: κ_t long-term prediction. Bold solid black line shows raw ("observed") estimates. Thin solid black lines correspond to median, 2.5 and 97.5 percentiles for the LSTM-ensembles trained on "raw data" (no pre-processing) with activation function ReLu. Thin dashed red lines correspond to median and percentiles for the RWD. Data to the right of the last vertical dashed line correspond to test data. For calibration LO, the area between the vertical dashed lines show in-sample validation data.

ered boosted models when reducing the amount of data used for calibration even more than previously. In the current section we will consider two different situations: training based on 1970 - 1989 together with testing on 1990 - 2006, and training based on 1980 - 1999 together with testing based on 2000 - 2016. The results for the test log-likelihoods calculated according to (15) for all calibrations are summarised in Table 4 and 5. As in Section 5.2.2 it is seen that the boosted RT and SP calibrations generally outperform Calibration LO. Further, for many of these populations the $\widehat{\kappa}_t$ processes are essentially linear, but the overall boosted model is similar to or only slightly worse than a standard RWD, see the analysis for e.g. Italian and Swedish women in the Supplementary Materials [18]. On the other hand, when there are non-linearities, the boosted models seem to capture these patterns reasonably well. Furthermore, by using boosted models the long-term predictions are reasonable as well.

6 Concluding remarks

In this paper, we focus on how to use data efficiently together with an LSTM neural network extension of the Poisson Lee-Carter model. We introduce alternative methods for calibration of the model, combined with ensembling, and illustrate that Calibration RT and SP are viable alternatives to the more

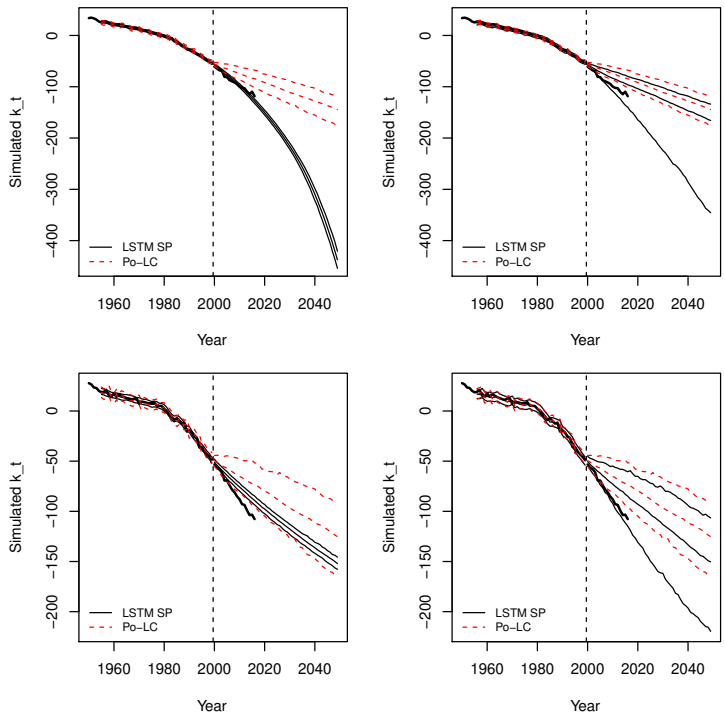

Figure 8: κ_t long-term prediction with Calibration SP. Top: Italian men. Bottom: Swedish men. Left: LSTM-ensemble trained on "raw data" (no pre-processing) with activation function ReLu. Right: LSTM-ensemble trained on residual after boosting and scaling with activation function tanh. Bold solid black line shows raw ("observed") estimates. Thin solid black lines correspond to median, 2.5 and 97.5 percentiles for the LSTM-ensembles. Thin dashed red lines correspond to median and percentiles for the RWD. Data to the right of the last vertical dashed line correspond to test data.

standard LO calibration. This can at least partly be motivated theoretically, see Section 3.6. The general approach to calibration, using LO, RT or SP, is of course not only applicable to LSTM neural network models, but can be used with other models as well, with obvious modifications. The need for using these alternative calibration procedures might be larger when the number of observations in available or relevant data is limited.

Furthermore, as seen in Section 5.2.2, when using boosting and applying the calibration methods to the residuals produced by first using a simpler model, as described in Section 3.7, we obtain more robust models that provide reasonable predictions for long-term forecasting horizons, without degrading the performance too much in the short-term. In our numerical illustrations

	Saturated	Po	LO	RT	SP
ITA male	-46 210	-56 707	**-49 449**	-60 846	-53 002
ITA female	-16 514	-24 465	-24 760	-20 969	**-19 362**
SWE male	-8 967	-12 122	-17 843	-10 007	**-9 522**
SWE female	-7 362	-8 634	**-8 015**	-9 009	-8 867
USA male	-87 053	-93 583	-94 232	-92 548	**-91 304**
USA female	-53 194	-145 147	-80 830	-85 327	**-61 746**

Table 4: Log-likelihood calculated according to (15) out-of-sample (1990-2006) for the Poisson Lee-Carter model and for the LSTM-ensemble trained on residual after boosting and scaling (1970-1989) with activation function tanh.

	Saturated	Po	LO	RT	SP
ITAmale	-54 346	-59 985	-64 402	-62 182	**-61 686**
ITAfemale	-18 477	-29 229	-23 731	-22 634	**-21 341**
SWEmale	-8 721	-9 636	-9 261	-9 098	**-8 929**
SWEfemale	-7 068	-8 169	-8 414	**-7 478**	-7 623
USAmale	-193 619	-201 830	**-199 872**	-203 877	-232 155
USAfemale	-87 416	-108 992	-159 287	**-133 011**	-161 754

Table 5: Log-likelihood calculated according to (15) out-of-sample (2000-2016) for the Poisson Lee-Carter model and for the LSTM-ensemble trained on residual after boosting and scaling (1980-1999) with activation function tanh.

these models consist of a compromise between a linear RWD model used for boosting, and a non-linear LSTM model. The resulting forecasts are close to the ones from a simple RWD when mortality rates are essentially log-linear, but can still capture some of the non-linearity in data when sufficiently strong non-linearities are present, without producing unreasonable long-term predictions. Additionally, boosting combined with Calibration RT and SP enables us to produce reasonable forecasts based on training data consisting of as few as 20 observations, though perhaps one should still be careful when attempting to use highly complex models when data is scarce.

Note that all figures in Section 5.2 only contain future evolutions of processes given point estimates. That is, we have not accounted for any estimation error in the prediction intervals. One way of including this is to use the bootstrap procedure described for the Poisson Lee-Carter model in [4]. However, this procedure would become computationally heavy when applied to

the ensemble models used in the present paper, which have been introduced to enhance the stability of the predictions.

Finally, the analysis in the present paper is based on that the simple model structure (3) and (2) is good enough to capture the dynamics in mortality data. Hence, the model used is rather inflexible when it comes to structural changes over time, since the estimates (α_x, β_x) will be fixed over the whole time period. It is thus too much to hope that this model will be able to produce reasonable results when trained on data over long time periods, giving part of the motivation behind trying to fit the model to limited data, even for cases where data for longer time horizons might be available. This problem can be seen for e.g. simulated in-sample mortality rates for USA females during 1950-1999, even though the corresponding κ_t process behaviour is reasonable, see the Supplementary Materials.

The focus in the present paper has been on one-dimensional models in a Poisson setting. A natural continuation would be to consider higher-dimensional versions of this type of Poisson Lee-Carter models. This might in itself lead to richer data, increasing the possibility of obtaining reliable model calibrations without having to increase the length of the time series in the time dimension.

References

[1] Patrik Andersson and Mathias Lindholm. Mortality forecasting using a Lexis-based state-space model. *Annals of Actuarial Science*, pages 1–30, 2020.

[2] Christoph Bergmeir, Rob J Hyndman, and Bonsoo Koo. A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis*, 120:70–83, 2018.

[3] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[4] Natacha Brouhns, Michel Denuit, and Ingrid Van Keilegom. Bootstrapping the Poisson log-bilinear model for mortality forecasting. *Scandinavian Actuarial Journal*, 2005(3):212–224, 2005.

[5] Natacha Brouhns, Michel Denuit, and Jeroen K Vermunt. A poisson log-bilinear regression approach to the construction of projected lifetables. *Insurance: Mathematics and Economics*, 31(3):373–393, 2002.

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

[7] Piet De Jong and Leonie Tickle. Extending Lee–Carter mortality forecasting. *Mathematical Population Studies*, 13(1):1–18, 2006.

[8] Philippe Deprez, Pavel V Shevchenko, and Mario V Wüthrich. Machine learning techniques for mortality modeling. *European Actuarial Journal*, 7(2):337–352, 2017.

[9] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[10] Felix A Gers, Jürgen A Schmidhuber, and Fred A Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

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

[12] Donatien Hainaut. A neural-network analyzer for mortality forecast. *ASTIN Bulletin: The Journal of the IAA*, 48(2):481–508, 2018.

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

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] 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.

[16] Ronald D Lee and Lawrence R Carter. Modeling and forecasting US mortality. *Journal of the American statistical association*, 87(419):659–671, 1992.

[17] Susanna Levantesi and Virginia Pizzorusso. Application of machine learning to mortality modeling and forecasting. *Risks*, 7(1):26, 2019.

[18] Mathias Lindholm and Lina Palmborg. Supplement to Efficient use of data for LSTM mortality forecasting. Available at `https://staff.math.su.se/lindholm/LSTM_SupplementaryMaterials.pdf`, 2021.

[19] Mario Marino and Susanna Levantesi. Measuring longevity risk through a neural network Lee-Carter model. *Available at SSRN 3599821*, 2020.

[20] Joao Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *Acm computing surveys (csur)*, 45(1):1–40, 2012.

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

[22] 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.

[23] Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble method for neural networks. In R. J. Mammone, editor, *Neural networks for speech and image processing*. Chapman & Hall, New York, 1993.

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

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

[26] Ronald Richman and Mario V. Wüthrich. A neural network extension of the Lee–Carter model to multiple populations. *Annals of Actuarial Science*, page 1–21, 2019.

[27] Andrés M Villegas, Vladimir K Kaishev, and Pietro Millossovich. StMoMo: An R package for stochastic mortality modeling. *Journal of Statistical Software*, 84(1):1–38, 2018.

A Parameters and listings

For all calibrations and time periods, the following parameters were used: lag 5, recurrent activation function sigmoid, 1 hidden layer, batch size 1,

20 model calibrations in each ensemble, patience 50 for the early stopping callback, and maximum 10 000 epochs.

The code listings below illustrate the different parameter values used in the different calibrations, depending on the length of the time period used for training, and whether boosting and scaling is used or not.

Listing 1: 1950-1999, "raw data"

```
model_lstm = keras_model_sequential() %>%
      layer_lstm(units = 50, input_shape = c(5, 1),
                 activation = "relu", recurrent_activation = "sigmoid") %>%
      layer_dense(units = 1, activation = "linear")
```

Listing 2: 1950-1999, boosting & scaling

```
model_lstm = keras_model_sequential() %>%
      layer_lstm(units = 50, input_shape = c(5, 1),
                 activation = "tanh", recurrent_activation = "sigmoid") %>%
      layer_dense(units = 1, activation = "linear")
```

Listing 3: 1970-1989 and 1980-1999, boosting & scaling

```
model_lstm = keras_model_sequential() %>%
      layer_lstm(units = 20, input_shape = c(5, 1),
                 activation = "tanh", recurrent_activation = "sigmoid") %>%
      layer_dense(units = 1, activation = "linear")
```