Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Case-Base Neural Networks: survival analysis with time-varying, higher-order interactions

Jesse Islam[1] | **Maxime Turgeon**[2] | **Robert Sladek**[3] | **Sahir Bhatnagar**[4]

[1]Department of Quantitative life sciences, McGill University, Quebec, Canada
[2]Department of Statistics, University of Manitoba, Manitoba, Canada
[3]Department of Human Genetics, McGill University, Quebec, Canada
[4]Department of Biostatistics, McGill University, Quebec, Canada

Correspondence
Jesse Islam's Email:
jesse.islam@mail.mcgill.ca

Present Address
McGill genome center: 740 Dr Penfield Ave, Montreal, Quebec H3A 0G1

Summary

Neural network-based survival methods can model data-driven covariate interactions. While these methods have led to better predictive performance than regression-based approaches, they cannot model both time-varying interactions and complex baseline hazards. To address this, we propose Case-Base Neural Networks (CBNN) as a new approach that combines the case-base sampling framework with flexible architectures. Our method naturally accounts for censoring and does not require method specific hyperparameters. Using a novel sampling scheme and data augmentation, we incorporate time directly into a feed-forward neural network. CBNN predicts the probability of an event occurring at a given moment to estimate the hazard function. We compare the performance of CBNN to survival methods based on regression and neural networks in a simulation and three real data applications. We report two time-dependent metrics for each model. For a complex simulation, which highlights the ability of CBNN to model both a complex baseline hazard and time-varying interactions and two real data applications, CBNN outperforms all competitors on a simulation with a complex baseline hazard and time-varying interaction and two case studies. A final case study shows competitive performance. We highlight the benefit of combining case-base sampling with deep learning to provide a simple and flexible modeling framework for data-driven, time-varying interaction modeling of survival outcomes. An R package is available at https://github.com/Jesse-Islam/cbnn.

KEYWORDS:
survival analysis, machine learning, case-base, neural network

1 | INTRODUCTION

Smooth-in-time accelerated failure time (AFT) models can estimate absolute risks by modeling the hazard directly through a user-specified baseline hazard distribution (Kleinbaum & Klein 2012). Analysts use Cox proportional hazards models more often than AFT models (Hanley & Miettinen 2009). This preference causes analyses to be based on hazard ratios and relative risks rather than on survival curves and absolute risks (Hanley & Miettinen 2009). With AFT models, it may be difficult to identify an appropriate baseline hazard for common diseases with many interacting risk factors (Royston & Parmar 2002). In studies where the disease pathogenesis may change with age, we may see non-proportional hazards (Coradini et al. 2000). For example, previous studies of breast cancer incidence have discovered time-varying interactions with covariates of interest, such as tumor size (Coradini et al. 2000). One approach to provide flexibility in the baseline hazard involves using the basis of splines on time in the model (Royston & Parmar 2002). However, regression-based models require prior knowledge of potential time-varying interactions and their quantitative effects.

Neural networks provide a data-driven approach to approximating interaction terms. For example, DeepSurv is a neural network-based proportional hazards model that implements the Cox partial log-likelihood as a custom loss function (Katzman et al. 2018). This results in a stepwise absolute risk curve that cannot accommodate time-varying interactions. Compared to Cox regression, DeepSurv shows better performance on a real dataset (Katzman et al. 2018). Previous work suggests we may change the loss function to address non-proportional hazards (Faraggi & Simon 1995).

DeepHit assumes an inverse Gaussian distribution as the baseline hazard (Lee, Zame, Yoon, & Schaar 2018). It specifies each follow-up time time of interest in the model and directly estimates survival curves, rather than deriving a hazard function (Lee et al. 2018). DeepHit outperformed DeepSurv (Lee et al. 2018).

We note that these alternative neural network approaches require custom loss functions (Katzman et al. 2018) (Lee et al. 2018). DeepHit introduces a hyperparameter weighing its two loss functions (negative log-likelihood and ranking losses) (Lee et al. 2018). Regression-based approaches require prior specification of all interaction terms, which makes it challenging to model covariate effects that change with time. The current neural network models provide flexibility at the cost of clarity, while regression models provide clarity at the cost of flexibility.

In this article, we propose Case-Base Neural Networks (CBNN) as a method that models time-varying interactions and a flexible baseline hazard. Our approach to modeling the full hazard uses case-base sampling (Hanley & Miettinen 2009). This sampling technique allows probabilistic models to predict survival outcomes. After case-base sampling, we can easily implement the model in any language using common neural network components. As part of the case-base framework, we use transformations of time as a feature (covariate) to specify different baseline hazards. For example, by including splines of time as covariates, we can approximate the Royston-Parmar flexible baseline hazard model (Royston & Parmar 2002)(Hanley & Miettinen 2009). However, this still requires explicit use of time-varying interactions. CBNN can model both without extra tuning parameters.

We describe how case-base sampling and neural networks combine in Section 2, along with our hyperparameter selection procedure, metrics and software implementation. We explore the performance of CBNN, DeepSurv, DeepHit, Cox regression and case-base using logistic regression (CBLR) on simulated data in Section 3. Section 4 describes the performance of the same methods in three case studies. Section 5 explores the implications of our results and contextualizes them within neural network survival analysis in a single event setting.

2 | CASE-BASE NEURAL NETWORKS, METRICS, HYPERPARAMETERS AND SOFTWARE

In this section, we define case-base sampling, which converts the total survival time into discrete person-specific moments (person-moments). Next, we detail how neural networks can be used within this framework, explicitly incorporating time as a feature while adjusting for the sampling bias. Then, we describe the metrics and hyperparameter selection procedure. Finally, we report on the software versions used. An R package is available for use at `https://github.com/Jesse-Islam/cbnn`. The entire code base to reproduce the figures and empirical results in this paper is available at `https://github.com/Jesse-Islam/cbnnManuscript`.

2.1 | Case-base sampling

Case-base sampling is an alternative framework for survival analysis (Hanley & Miettinen 2009). Without this preliminary step, we cannot model survival outcomes with probabilistic methods. In case-base sampling, we sample from the continuous survival time of each person in our dataset to create a *base series* of *person-moments*. This *base series* complements the *case series*, which contains all person-moments at which the event of interest occurs.

For each person-moment sampled, let X_i be the corresponding covariate profile $(x_{i1}, x_{i2}, ..., x_{ip})$, T_i be the time of the person-moment and Y_i be the indicator variable for whether the event of interest occurred at time T_i. We estimate the hazard function $h(t \mid X_i)$ using the sampled person-moments. Recall that $h(t \mid X_i)$ is the instantaneous potential of experiencing the event at time t for a given set of covariates X_i, assuming $T_i \geq t$.

Now, let b be the (user-defined) size of the *base series* and let B be the sum of all follow-up times for the individuals in the study. let c be the number of events in the *case series*. A reasonable concern is determining how large b should be relative to c. The size of b influences how much information is lost in the sampling process (Hanley & Miettinen 2009). Using our notation, Hanley $\&$ Miettinen (2009) adapt a quote from Mantel (1973): "By the reasoning that $\frac{cb}{c+b}[= (\frac{1}{c} + \frac{1}{b}]$ measures the relative information in a comparison of two averages based on sample sizes of c and b respectively, we might expect by analogy, which would of course not be exact in the present case, that this approach would result in only a moderate loss of information. (The practicing statistician is generally aware of this kind of thing. There is little to be gained by letting the size of one series, b, become arbitrarily large if the size of the other series, c, must remain fixed)" (Hanley & Miettinen 2009) (Mantel 1973). If $b = 100c$,

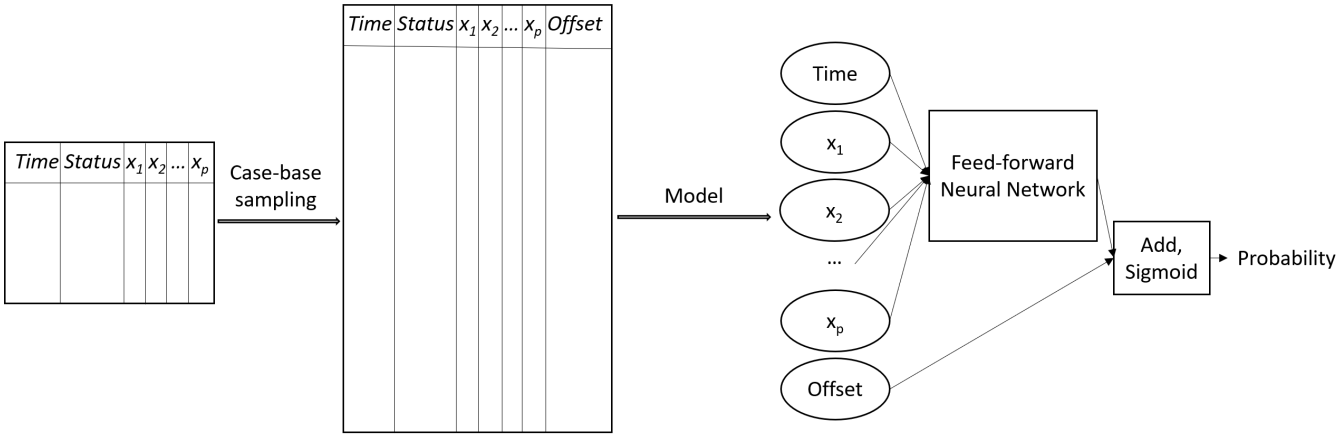

Figure 1 Steps involved in CBNN from case-base sampling to the model framework we use for training. The first step is case-base sampling, completed before training begins. Next, we pass this sampled data through a feed-forward neural network. We add the offset and pass that through a sigmoid activation function, whose output is a probability. Once the neural network model completes its training, we can convert the probability output to a hazard, using it for our survival outcomes of interest.

then we expect learned weights to be at most one percent higher than if the entire study base B was used, as they are proportional to $\frac{1}{c} + \frac{1}{100c}$ rather than $\frac{1}{c} + \frac{1}{\infty c}$ (Hanley & Miettinen 2009).

If we sample the *base series* uniformly across the study base, then the hazard function of the sampling process is equal to b/B. Therefore, we have the following equality Saarela and Hanley (2015) [1]:

$$\frac{P\left(Y_i = 1 \mid X_i, T_i\right)}{P\left(Y_i = 0 \mid X_i, T_i\right)} = \frac{h\left(T_i \mid X_i\right)}{b/B}. \tag{1}$$

The odds of a person-moment being a part of the *case series* is the ratio of the hazard $h(T_i \mid X_i)$ and the uniform rate b/B. Using (1), we can see how the log-hazard function can be estimated from the log-odds arising from case-base sampling:

$$\log\left(h\left(t \mid X_i\right)\right) = \log\left(\frac{P\left(Y_i = 1 \mid X_i, t\right)}{P\left(Y_i = 0 \mid X_i, t\right)}\right) + \log\left(\frac{b}{B}\right). \tag{2}$$

To estimate the correct hazard function, we adjust for the bias introduced when sampling a fraction of the study base ($\frac{b}{B}$) by adding the term $\log\left(\frac{B}{b}\right)$ to offset $\log\left(\frac{b}{B}\right)$ during the fitting process. Next, we propose using neural networks to model the odds.

2.2 | Neural networks to model the hazard function

After case-base sampling, we pass all features, including time, into any user-defined feed-forward component, to which an offset term is added, then passed through a sigmoid activation function (Figure 1). We are interested in predicting the odds. This makes the sigmoid activation function ideal as it is the inverse of the odds. which we can use to calculate the hazard. The general form for the neural network using CBNN is:

$$P\left(Y = 1|X, T\right) = \text{sigmoid}\left(f_\theta(X, T) + \log\left(\frac{B}{b}\right)\right),$$

where T is a random variable representing the event time, X is the random variable for a covariate profile, $f_\theta(X, T)$ represents any feed-forward neural network architecture, $\log\left(\frac{B}{b}\right)$ is the offset term for adjust for the bias ($\log\left(\frac{b}{B}\right)$) set by case-base sampling, θ is the set of parameters learned by the neural network and $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$. By approximating a higher-order polynomial of time using a neural network, the baseline hazard specification is now data-driven, where user-defined hyperparameters such as regularization, number of layers and nodes control the flexibility of the hazard function. We provide a detailed description of the choices we made in the next sub-section.

[1] We are abusing notation here, conflating hazards with probabilities. For a rigorous treatment, see Saarela & Hanley (2015) section 3 Saarela and Hanley (2015) .

The following derivation shows how our probability estimate is converted to odds:

$$\begin{aligned}
\log\left(h(t \mid X)\right) &= \log\left(\frac{\operatorname{sigmoid}\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}{1-\operatorname{sigmoid}\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}\right)+\log\left(\frac{b}{B}\right)\\
&= \log\left(\frac{\frac{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)+1}}{1-\frac{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)+1}}\right)+\log\left(\frac{b}{B}\right)\\
&= \log\left(\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)\right)+\log\left(\frac{b}{B}\right)\\
&= f_\theta(X,T)+\log\left(\frac{B}{b}\right)+\log\left(\frac{b}{B}\right)\\
&= f_\theta(X,T).
\end{aligned}$$

We use binary cross-entropy as our loss function (Gulli & Pal 2017):

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\hat{f}_\theta(x_i,t_i)) + (1-y_i)\cdot\log(1-\hat{f}_\theta(x_i,t_i)),$$

where $\hat{f}_\theta(x_i,t_i)$ is our estimate for a given covariate profile and time, y_i is our target value specifying whether an event occurred and N represents the number of individuals in our training set.

Backpropagation with an appropriate minimization algorithm (e.g. Adam, RMSPropagation, stochastic gradient descent) is used to optimize the parameters in the model (Gulli & Pal 2017). For our analysis, we use Adam as implemented in Keras (Gulli & Pal 2017). Note that the size of the *case series* is fixed as the number of events, but we can make the *base series* as large as we want. A ratio of 100:1 *base series* to *case series* is sufficient (Hanley & Miettinen 2009). We pass our feed-forward neural network through a sigmoid activation function (Figure 1). Finally, we can convert this model output to a hazard. To use CBNN for predictions, we manually set the offset term $\log\left(\frac{B}{b}\right)$ to 0 in the new data as we already account for the bias during the fitting process.

Since we are directly modeling the hazard, we can readily estimate the risk function (F) at time t for a covariate profile X, viz.

$$F\left(t \mid X\right) = 1-\exp\left(-\int_0^t h(u|X)\,\mathrm{d}u\right). \tag{3}$$

We use a finite Riemann sum (Hughes-Hallett, Gleason, & McCallum 2020) to approximate the integral in (3).

2.3 | Performance metrics

To choose our method-specific hyperparameters, we use the Integrated Brier Score (IBS) (Graf, Schmoor, Sauerbrei, & Schumacher 1999). We use two metrics to assess the performance of the different methods of interest on testing data: 1) Index of Prediction Accuracy (IPA) (Kattan & Gerds 2018) and 2) Inverse probability censoring weights-adjusted time dependent area under the receiver operating characteristic curve (AUC_{IPCW}) (Blanche et al. 2015), which we define below. We achieve a summarized assessment of performance for each model with the IBS. The IPA is a metric for both discrimination and calibration. The AUC_{IPCW} provides a metric for discrimination only. Together, we can get a clearer picture about how each model performs in each study and why.

2.3.1 | Index of prediction accuracy (IPA)

The IPA is a function of the Brier score ($BS(t)$) (Graf et al. 1999), which is defined as

$$BS(t) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\left(1-\widehat{F}(t \mid X_i)\right)^2 \cdot I(T_i \le t, \delta_i = 1)}{\widehat{G}(T_i)} + \frac{\widehat{F}(t \mid X_i)^2 \cdot I(T_i > t)}{\widehat{G}(t)}\right), \tag{4}$$

where $\delta_i = 1$ shows individuals who have experienced the event, N represents the number of samples in our dataset over which we calculate $BS(t)$, $\widehat{G}(t) = P[c > t]$ is a non-parametric estimate of the censoring distribution, c is censoring time and T_i is an individual's survival or censoring time. The BS provides a score that accounts for the information loss because of censoring. There are three categories of individuals that may appear within the dataset once we fix our t of interest. Individuals who experienced the event before t are present in the first term of the equation. The second term of the equation includes individuals who experience the event or are censored after t. Those censored before t are the third category of people. The inverse probability censoring weights (IPCW) adjustment ($G(\cdot)$) is to account for these category three individuals whose information is missing.

The IPA as a function of time is given by

$$\text{IPA}(t) = 1 - \frac{BS_{model}(t)}{BS_{null}(t)},$$

where $BS_{model}(t)$ represents the BS over time t for the model of interest and $BS_{null}(t)$ represents the BS if we use an unadjusted Kaplan-Meier (KM) curve as the prediction for all observations (Kattan & Gerds 2018). Note that IPA has an upper bound of one, where positive values show an increase in performance over the null model and negative values show that the null model performs better. These scores show how performance changes over follow-up time. As an extension of BS, IPA models for both calibration and discrimination(Graf et al. 1999) (Kattan & Gerds 2018).

2.3.2 | Integrated Brier Score (IBS)

The Integrated Brier Score (IBS) is a function of the BS 4 as well (Graf et al. 1999), which is defined as

$$IBS(t) = \int_{t}^{t_{max}} BS(t)dw(t),$$

where $w(t) = \frac{t}{t_{max}}$. As this is a score for a range of follow-up-times, it is a useful metric to track during cross-validation. The performance at a specific time is lost, and is why we opt for the IPA in assessing performance on a test set.

2.3.3 | Inverse probability censoring weights-adjusted time dependent area under the receiver operating characteristic curve (AUC_{IPCW})

The IPCW-adjusted AUC (AUC_{IPCW}) is a time-dependent metric that considers censoring (Blanche et al. 2015). For a given follow-up time of interest,

$$AUC_{IPCW}(t) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} I\left(\widehat{F}(t|X_i) > \widehat{F}(t|X_j)\right) \cdot I(T_i \leq t, \delta_i = 1) \cdot (1 - I(T_j \leq t, \delta_j = 1)) \cdot W_i(t) \cdot W_j(t)}{\sum_{i=1}^{n}\sum_{j=1}^{n} I(T_i \leq t, \delta_i = 1) \cdot (1 - I(T_j \leq t, \delta_j = 1)) \cdot W_i(t) \cdot W_j(t)},$$

where

$$W_i(t) = \frac{I(T_i \leq t, \delta_i = 1)}{\widehat{G}(T_i)} + \frac{I(T_i > t)}{\widehat{G}(t)}.$$

The AUC_{IPCW} measures discrimination (Blanche et al. 2015). By taking both IPA and AUC_{IPCW} together, we can better understand whether a model is better in terms of calibration or discrimination.

2.4 | Hyperparameter selection

Neural networks are flexible when defining the architecture and optimization parameters. These hyperparameter choices can affect the estimated parameters and are not necessarily the same for each model. We apply the same procedure to each study in this paper. First, we fix a test set with 15% of the data. We keep aside this data during hyperparameter selection and we fit a model for each method in each study. To determine the best hyperparameters for each neural network method, we use three-fold cross-validated grid search (Gulli & Pal 2017) on the remaining data (85% training, 15% validation) for the following range of hyperparameters:

$$Grid\ search: \begin{cases} Learning\ rate \sim \{0.001, 0.01\} \\ Dropout \sim \{0.01, 0.05, 0.1\} \\ First\ layer\ nodes \sim \{50, 75, 100\} \\ Second\ layer\ nodes \sim \{10, 25, 50\} \\ Number\ of\ batches \sim \{100, 500\} \\ Activation\ function \sim \{ReLU, Linear\} \\ \alpha \sim \{0, 0.5, 1\}. \end{cases}$$

DeepHit uses α as a specific hyperparameter, while DeepSurv and CBNN do not. We track IBS on the validation set for each hyperparameter combination, choosing the combination with the lowest score for each method. We present the chosen hyperparameters in Table 1.

Table 1 Four tables showing the chosen hyperparameters along with the IBS for each neural network model in the complex simulation (A), multiple myeloma (MM) case study (B), free light chain (FLC) case study (C) and prostate cancer (Prostate) case study (D).

A: Complex

Hyper parameter	CBNN	DeepSurv	DeepHit
Learning rate	0.001	0.001	0.001
Dropout	0.01	0.01	0.05
First layer nodes	75	50	50
Second layer nodes	50	50	50
Number of batches	100	500	100
Activation function	relu	relu	linear
α	-	-	1
IBS	0.06744259	0.07529775	0.08821864

B: MM

Hyper parameter	CBNN	DeepSurv	DeepHit
Learning rate	0.001	0.01	0.01
Dropout	0.1	0.05	0.01
First layer nodes	50	100	100
Second layer nodes	50	25	25
Number of batches	500	100	100
Activation function	relu	relu	relu
α	-	-	0
IBS	0.09414844	0.09912649	0.1097949

C: FLC

Hyper parameter	CBNN	DeepSurv	DeepHit
Learning rate	0.001	0.001	0.01
Dropout	0.05	0.05	0.1
First layer nodes	50	100	50
Second layer nodes	10	10	10
Number of batches	100	100	100
Activation function	relu	relu	relu
α	-	-	1
IBS	0.09903534	0.09900328	0.09979871

D: Prostate

Hyper parameter	CBNN	DeepSurv	DeepHit
Learning rate	0.001	0.01	0.01
Dropout	0.01	0.1	0.05
First layer nodes	100	75	75
Second layer nodes	25	25	25
Number of batches	100	100	500
Activation function	relu	relu	linear
α	-	-	1
IBS	0.07618916	0.07631809	0.07634624

2.5 | Software implementation

R (R Core Team 2021) and python (Van Rossum & Drake 2009) are used to evaluate methods from both languages. We fit the Cox model using the **survival** package (Terry M. Therneau & Patricia M. Grambsch 2000), the CBLR model using the **casebase** package (Bhatnagar, Turgeon, Islam, Hanley, & Saarela 2020), DeepSurv and DeepHit using **pycox** (Lee et al. 2018). We made the components of CBNN using the **casebase** package for the sampling step and the **keras** (Allaire & Chollet 2021) package for our neural network architecture. The **simsurv** package (Brilleman, Wolfe, Moreno-Betancur, & Crowther 2020) is used for our simulation studies, while **flexsurv** (Jackson 2016) is used to fit a flexible baseline hazard using splines for our complex simulation. The **riskRegression** package (Gerds & Kattan 2021) is used to get the Index of Prediction Accuracy (IPA metric) and AUC. Both metrics are described in detail in the following section. We modify the **riskRegression** package to be used with any user supplied risk function F. To ensure that both R and Python-based models are running in unison on the same data through our simulations and bootstrap, we use the **reticulate** package (Ushey, Allaire, & Tang 2021).

3 | SIMULATION STUDY

In this section, we simulate data to evaluate the performance of CBNN and compare our approach with existing regression-based (Cox, CBLR) and neural network-based (DeepHit, DeepSurv) methods. We specify a linear combination of each covariate as the linear predictor in regression-based approaches (Cox, CBLR), which contrasts with neural network approaches that allow for non-linear interactions. We simulate data under a complex baseline hazard with time-varying interactions, each with 10% random censoring. For both settings, we simulate three covariates for 5000 individuals:

$$z_1 \sim \text{Bernoulli}(0.5) \qquad z_2 \sim \begin{cases} N(0, 0.5) & \text{if } z_1 = 0 \\ N(1, 0.5) & \text{if } z_1 = 1 \end{cases} \qquad z_3 \sim N(1, 0.5).$$

Besides the methods mentioned above, we include the Optimal model in our comparisons using CBLR. We include the exact functional form of the covariates in a CBLR model (referred to as Optimal for simplicity). We conduct 100 bootstrap re-samples on the training data to get confidence intervals. We kept 15% of the data for testing before hyperparameter selection. 15% of the remaining data is used for validation and the rest is reserved for training. We predict risk functions for individuals in the test set, which are used to calculate our AUC_{IPCW} and IPA.

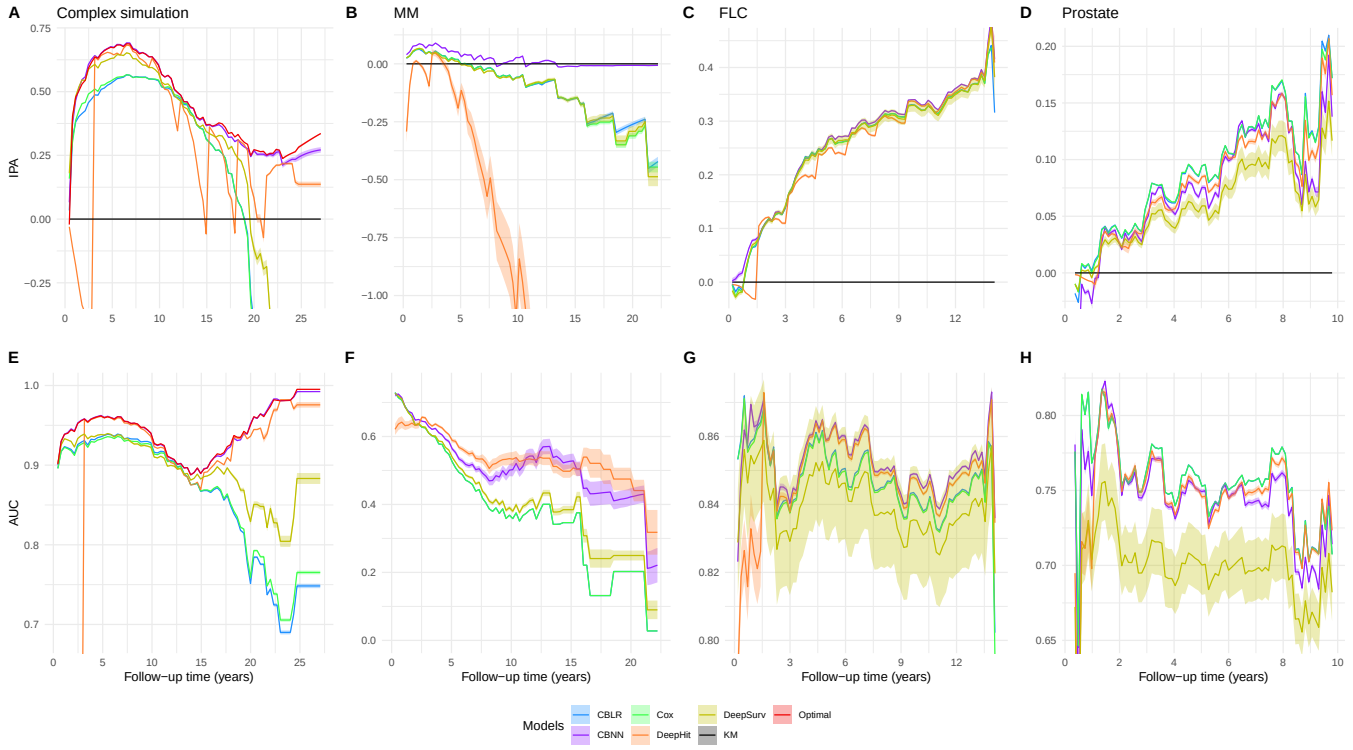

Figure 2 Summarizes the complex simulation (A, E), multiple myeloma (MM) case study (B, F), free light chain (FLC) case study (C, G) and prostate cancer (Prostate) case study (D, H). The first row shows the IPA for each model in each study over follow-up time. Negative values mean our model performs worse than the null model and positive values mean the model performs better. The second row shows the AUC_{IPCW} for each model in each study over follow-up time. Each model-specific metric in each study shows a 95% confidence interval over 100 iterations. The models of interest are case-base with logistic regression (CBLR), Case-Base Neural Networks (CBNN), Cox, DeepHit and DeepSurv. Both CBLR and Cox have near identical performance, resulting in curves that are overlapped. The Kaplan-Meier (KM) model serves as a baseline, predicting the average curve for all individuals. The Optimal (a CBLR model with the exact interaction terms and baseline hazard specified) model shows the best performance we can expect on the simulated data.

3.1 | Complex simulation: flexible baseline hazard, time-varying interactions

This simulation shows performance with the presence of a complex baseline hazard and a time-varying interaction. Inspired by a cancer treatment that initially reduces risk of death, but slowly returns as time progresses. Originally used to show the spline-based hazard model proposed by Royston and Parmar (Royston & Parmar 2002), the breast cancer dataset provides a complex baseline hazard from which we simulate, available in the **flexsurv** R package (Jackson 2016). To increase the complexity of our data-generating mechanism for this simulation, we design the model

$$\log h(t \mid X_i) = \sum_{i=1}^{5}(\gamma_i \cdot \psi_i) + \beta_1(z_1) + \beta_2(z_2) + \beta_3(z_3) + \tau_1(z_1 \cdot t) + \tau_2(z_2 \cdot z_3),$$

where $\gamma_1 = 3.9, \gamma_2 = 3, \gamma_3 = -0.43, \gamma_4 = 1.33, \gamma_5 = -0.86, \beta_1 = -5, \beta_2 = -1, \beta_3 = 1, \tau_1 = 0.001, \tau_2 = -1$ and ψ are basis splines. The $gamma$ coefficients are obtained from an intercept-only cubic splines model with three knots using the *flexsurvspline* function from the **flexsurv** package (Jackson 2016). Note that we fix these values for the analysis. The β coefficients represent direct effects, τ_2 represents an interaction and τ_1 is a time-varying interaction.

3.2 | Performance comparison in complex simulation

Figure 2 A, E and Table 2 A shows the performance over time on a test set. For discrimination, the Optimal model performs best as expected, followed by CBNN, DeepHit, DeepSurv and the linear models (Figure 2 E). For both discrimination and calibration, the rankings remain the same aside from DeepHit, where there's a periodic loss in performance (Figure 2 A). The Optimal model acts as a reference for how good a model can

perform, with CBNN performing better than the competitors in the complex simulation. To obtain a more realistic performance assessment, we compared models in three case studies with a time-to-event outcome.

Table 2 Four tables representing performance at certain percentages of follow-up time in the complex simulation (A), multiple myeloma (MM) case study (B), free light chain (FLC) case study (C) and prostate cancer (Prostate) case study (D). Each table shows performance for each method in each study at 25%, 50%. 75% and 100% of follow-up time. The bold elements show the best model for each study, at each follow-up time of interest. These tables are included to provide exact measures at certain intervals. The models of interest are: Cox, case-base with logistic regression (CBLR), DeepSurv, DeepHit, Case-Base Neural Network (CBNN) and Optimal (in the complex simulation). The CBNN label is in bold to make it easier to distinguish. The best score at each percent of follow-up time is highlighted in bold. If the average performance is tied, then all tied values are bold.

A: Complex	IPA				AUC			
Method	25	50	75	100	25	50	75	100
Cox	0.56 (0.56,0.57)	0.41 (0.4,0.41)	-0.53 (-0.54,-0.52)	-1.87 (-1.89,-1.84)	0.93 (0.93,0.94)	0.89 (0.88,0.89)	0.79 (0.79,0.79)	0.77 (0.76,0.77)
CBLR	0.56 (0.56,0.57)	0.4 (0.4,0.41)	-0.46 (-0.46,-0.45)	-1.5 (-1.52,-1.48)	0.94 (0.94,0.94)	0.89 (0.89,0.89)	0.78 (0.78,0.79)	0.75 (0.75,0.75)
DeepSurv	0.65 (0.65,0.65)	0.4 (0.4,0.4)	-0.16 (-0.18,-0.13)	-1.02 (-1.08,-0.95)	0.94 (0.94,0.94)	0.89 (0.89,0.89)	0.85 (0.85,0.85)	0.88 (0.88,0.89)
DeepHit	0.68 (0.68,0.68)	0.3 (0.3,0.31)	0.00 (-0.01,0.02)	0.14 (0.13,0.15)	**0.96 (0.96,0.96)**	0.88 (0.88,0.89)	0.94 (0.94,0.95)	0.98 (0.97,0.98)
CBNN	**0.69 (0.69,0.69)**	0.43 (0.43,0.43)	0.25 (0.25,0.26)	0.27 (0.26,0.28)	**0.96 (0.96,0.96)**	**0.9 (0.9,0.9)**	**0.96 (0.96,0.96)**	0.99 (0.99,0.99)
Optimal	**0.69 (0.69,0.69)**	**0.44 (0.43,0.44)**	**0.27 (0.27,0.27)**	**0.34 (0.33,0.34)**	**0.96 (0.96,0.96)**	**0.9 (0.9,0.9)**	**0.96 (0.96,0.96)**	**1 (1,1)**

B: MM	IPA				AUC			
Method	25	50	75	100	25	50	75	100
Cox	0.00 (0.00,0.00)	-0.09 (-0.1,-0.09)	-0.26 (-0.27,-0.25)	-0.45 (-0.47,-0.42)	0.51 (0.51,0.51)	0.37 (0.37,0.37)	0.13 (0.13,0.13)	0.03 (0.03,0.03)
CBLR	0.00 (0.00,0.00)	-0.1 (-0.1,-0.09)	-0.25 (-0.26,-0.24)	-0.42 (-0.44,-0.4)	0.51 (0.51,0.51)	0.37 (0.37,0.37)	0.13 (0.13,0.13)	0.03 (0.03,0.03)
DeepSurv	0.00 (-0.01,0.00)	-0.09 (-0.09,-0.08)	-0.24 (-0.26,-0.22)	-0.49 (-0.53,-0.45)	0.52 (0.52,0.53)	0.39 (0.39,0.4)	0.24 (0.21,0.27)	0.09 (0.06,0.12)
DeepHit	-0.16 (-0.2,-0.12)	-1.21 (-1.42,-1)	-3.86 (-4.4,-3.31)	-3.66 (-4.17,-3.15)	**0.59 (0.58,0.59)**	**0.53 (0.51,0.55)**	**0.52 (0.48,0.56)**	**0.32 (0.25,0.38)**
CBNN	**0.04 (0.04,0.04)**	**0.00 (-0.01,0.00)**	**-0.01 (-0.01,-0.01)**	**-0.01 (-0.01,-0.01)**	0.55 (0.55,0.56)	0.51 (0.49,0.53)	0.43 (0.4,0.47)	0.22 (0.17,0.27)

C: FLC	IPA				AUC			
Method	25	50	75	100	25	50	75	100
Cox	**0.19 (0.19,0.19)**	0.29 (0.29,0.29)	0.33 (0.33,0.33)	**0.44 (0.44,0.44)**	**0.85 (0.85,0.85)**	0.85 (0.85,0.86)	0.84 (0.84,0.84)	0.8 (0.79,0.8)
CBLR	**0.19 (0.19,0.19)**	**0.3 (0.29,0.3)**	0.33 (0.33,0.33)	0.32 (0.31,0.32)	**0.85 (0.85,0.85)**	**0.86 (0.86,0.86)**	0.84 (0.84,0.84)	0.8 (0.8,0.8)
DeepSurv	**0.19 (0.18,0.2)**	0.29 (0.28,0.31)	0.33 (0.32,0.34)	0.38 (0.33,0.43)	0.84 (0.82,0.85)	0.85 (0.83,0.87)	0.83 (0.82,0.85)	0.82 (0.8,0.84)
DeepHit	0.18 (0.18,0.19)	0.29 (0.29,0.29)	0.33 (0.33,0.33)	0.41 (0.36,0.46)	**0.85 (0.85,0.85)**	**0.86 (0.86,0.86)**	**0.85 (0.85,0.85)**	0.83 (0.83,0.84)
CBNN	**0.19 (0.19,0.2)**	**0.3 (0.3,0.3)**	**0.34 (0.34,0.34)**	0.42 (0.41,0.42)	**0.85 (0.85,0.85)**	**0.86 (0.86,0.86)**	**0.85 (0.85,0.85)**	**0.84 (0.83,0.84)**

D: Prostate	IPA				AUC			
Method	25	50	75	100	25	50	75	100
Cox	**0.04 (0.04,0.04)**	**0.09 (0.09,0.09)**	**0.14 (0.14,0.14)**	**0.17 (0.17,0.17)**	**0.77 (0.76,0.77)**	**0.75 (0.75,0.75)**	**0.76 (0.76,0.76)**	0.71 (0.71,0.71)
CBLR	**0.04 (0.04,0.04)**	**0.09 (0.09,0.09)**	**0.14 (0.13,0.14)**	**0.17 (0.17,0.17)**	**0.77 (0.76,0.77)**	**0.75 (0.75,0.75)**	**0.76 (0.76,0.76)**	0.71 (0.71,0.71)
DeepSurv	0.03 (0.03,0.04)	0.06 (0.05,0.07)	0.1 (0.09,0.11)	0.12 (0.1,0.13)	0.71 (0.69,0.73)	0.7 (0.68,0.72)	0.7 (0.68,0.72)	0.68 (0.66,0.7)
DeepHit	**0.04 (0.03,0.04)**	0.08 (0.08,0.09)	0.12 (0.12,0.13)	0.16 (0.15,0.16)	**0.77 (0.76,0.77)**	**0.75 (0.74,0.75)**	0.75 (0.75,0.75)	**0.72 (0.72,0.73)**
CBNN	**0.04 (0.04,0.04)**	0.08 (0.07,0.08)	0.12 (0.12,0.13)	0.14 (0.13,0.14)	0.76 (0.76,0.76)	**0.75 (0.75,0.75)**	0.74 (0.74,0.74)	0.71 (0.71,0.72)

4 | CASESTUDIES

The first case study examines multiple myeloma (MM) (Kyle 1997). Case study two examines the relationship between serum free light chain (FLC) and mortality (Dispenzieri et al. 2012). The third examines prostate cancer (Prostate) survival on a publicly available simulation of the Surveillance, Epidemiology, and End Results (SEER)-medicare study (Lu-Yao et al. 2009). Both MM and FLC are available as part of the **survival** package (Terry M. Therneau & Patricia M. Grambsch 2000). The Prostate dataset is available as part of the **asaur** package (Moore 2016). As we do not know the true model for the case studies, we exclude the Optimal model. After we perform grid search with the same hyperparameter options, we split the data the same way we did for the simulation (allocated 15% of the data for testing, 15% of the remaining data for validation, and the rest for testing). We predict risk functions for everyone in the test set, which is used to calculate our metrics. We conduct 100-fold bootstrap re-samples on the training data to get confidence intervals.

4.1 | Performance evaluation on multiple myeloma dataset

The MM study tracks 3882 patients seen at the Mayo Clinic from 1947 to 1996 until death (Kyle 1997). We use 2 covariates, year of entry into the study and the time of MM diagnosis(Kyle 1997). We see 71% incidence over 23 years(Kyle 1997).

Figure 2 B, F and Table 2 B demonstrate the performance over time on a test set. For discrimination, CBNN and DeepHit have a similar performance, followed by DeepSurv and finally the linear models performing worst (Figure 2 F). For both discrimination and calibration, CBNN performs substantially better than the linear models and DeepSurv, with DeepHit having the worst performance overall (Figure 2 B). Together, CBNN is the best calibrated model and one of the best at discrimination in the MM case study.

4.2 | Performance evaluation on free light chain dataset

The FLC dataset is a random sample of half the individuals in $\frac{2}{3}$ of the residents of Olmsted County over the age of 50 (Dispenzieri et al. 2012). We have access to 7874 subjects tracked until death (Dispenzieri et al. 2012). We use 5 covariates, total serum FLC (sum of kappa and lambda), age, sex, serum creatine and monoclonal gammapothy state (Dispenzieri et al. 2012). There is 27% incidence over 14 years (Dispenzieri et al. 2012).

Figure 2 C, G and Table 2 C demonstrate the performance over time on a test set. CBNN, DeepHit and the linear models perform best at discrimination, followed DeepSurv (Figure 2 G). The rankings remain the same aside from DeepHit where the performance periodically drops to worse than DeepSurv (Figure 2 C). Together, CBNN outperforms the competing models, in terms of both calibration and discrimination in the FLC study. However, CBNN is only slightly better than the linear ones.

4.3 | Performance evaluation on prostate cancer dataset

The prostate cancer dataset has a record of competing risks (Lu-Yao et al. 2009). As we are only interested in the single event scenario, we only keep individuals with prostate cancer death or censoring (Lu-Yao et al. 2009). This subset tracks 11054 individuals with three covariates: differentiation grade, age group and cancer state (Lu-Yao et al. 2009). There is a 7% incidence over 10 years.

Figure 2 C, G and Table 2 C demonstrate the performance over time on a test set. In the prostate case study, the linear models outperform the other models in both discrimination and calibration, followed by CBNN and DeepHit and finally DeepSurv (Figure 2 C, G). Aside from DeepSurv, the performance is similar across all models.

5 | DISCUSSION

CBNN models survival outcomes by using neural networks on case-base sampled data. We incorporate follow-up time as a feature, providing a data-driven estimate of a flexible baseline hazard and time-varying interactions in our hazard function. The two competing neural network models we evaluated cannot model time-varying interactions (Katzman et al. 2018) (Lee et al. 2018).

A method we want to include is Deep Survival Machines (DSM), a parametric survival model using neural networks with a mixture of distributions as the baseline hazard (Nagpal, Li, & Dubrawski 2021). It can model a flexible baseline hazard. However, like DeepHit and DeepSurv, DSM cannot model time-varying interactions (Nagpal et al. 2021). DSM requires a two-phase learning process and user implementations for distributions beyond Log-Normal or Weibull (Nagpal et al. 2021). DSM outperformed DeepSurv and DeepHit, but only tested a subset of hyperparameter combinations for competing neural network methods while going through a full grid search for their method (Nagpal et al. 2021). With our hyperparameters

options and model design, DSM did not converge in the complex simulation. This may be due to implementation issues or method limitations, therefore we did not include this method.

Compared to CBNN, the neural network competitors also have limitations. DeepSurv is a proportional hazards model and does not estimate the baseline hazard (Katzman et al. 2018). DeepHit requires an alpha hyperparameter, assumes a single distribution for the baseline hazard and models the survival function directly (Lee et al. 2018). The alternative neural network methods match on time, while CBNN models time directly.

Though we applied a full grid search for all neural network models, there are an infinite number of untested hyperparameters we did not test. A completely exhaustive search is computationally infeasible, therefore it is reasonable to expect that there is a set of hyperparameters that is better suited for each model. However, we test a reasonably large range while accounting for potential over-fitting by including dropout (Gulli & Pal 2017). We provided the same options to all models aside from DeepHit, which has a method specific hyperparameter α (Lee et al. 2018). With these limitations in mind, we dive into the differences in performance across all tested methods.

The complex simulation requires a method that can learn both time-varying interactions and have a flexible baseline hazard. Here, CBNN shows a distinct advantage over all other methods. Based on our complex simulation results (Figure 2 A, E and Table 2 A), CBNN outperforms the competitors. This simulation shows how all models perform under ideal conditions with minimal noise in the data, while the three case studies assess their performance in realistic conditions. In the MM case study, flexibility in both interaction modeling and baseline hazard improves CBNN's performance over the other models, suggesting that this flexibility aids calibration (Figure 2 B, F and Table 2 B). upon examination of the FLC case study, CBNN demonstrates small improvement to performance compared to the linear models and DeepHit for both IPA and AUC (Figure 2 C, G and Table 2 C). In the Prostate case study, the linear models outperform the neural network ones, while CBNN and DeepHit alternate their positions depending on the follow-up time of interest and DeepSurv maintains last place (Figure 2 C, G and Table 2 C). We attribute this to potential over-parameterization in the neural network models, as we did not test for fewer nodes in each hidden layer, even with dropout. Though the ranking places the linear models above the neural network ones, the overall performance is comparable aside from DeepSurv, falling within a small range of IPA and AUC values.

If prediction is our goal, we suggest CBNN as the best model in the Complex simulation, MM case study and FLC case study. The linear models were competitive in the FLC case study and performed best in the prostate case study. Though neural network interpretability is steadily improving (Zhang, Tiňo, Leonardis, & Tang 2021), there is still a trade-off compared to regression models, especially when the predictive performance gain is minimal, like in the FLC case study. We suggest that reference models should be included when assessing neural network models. Both a null model (KM curve) and a linear model (either Cox or a flexible baseline hazard model like CBLR) provide insight as to whether the neural network model is learning anything useful beyond linear predictors that were not accounted for.

6 | CONCLUSIONS

CBNN outperforms all competitors in the complex simulation and two case studies while maintaining competitive performance in a final case study. Once we perform case-base sampling and adjust for the sampling bias, we can use a sigmoid activation function to predict our hazard function. Our approach provides an alternative to incorporating censored individuals, treating survival outcomes as binary ones. Forgoing the requirement of custom loss functions, CBNN only requires the use of standard components in machine learning libraries (specifically, the add layer to adjust for sampling bias and the sigmoid activation function) after case-base sampling. Due to the simplicity in its implementation and by extension user experience, CBNN is both a user-friendly approach to data-driven survival analysis and is easily extendable to any feed-forward neural network framework.

Data and code availability statement

The Prostate dataset is available as part of the **asaur** package in R. The MM and FLC datasets are available in the **survival** package in R. The code for this manuscript and its analyses can be found at `https://github.com/Jesse-Islam/cbnnManuscript`. The software package making CBNN easier to implement can be found at `https://github.com/Jesse-Islam/cbnn`.

Acknowledgements

We would like to thank Dr. James Meigs, the project leader of these awards, for his support and helpful discussions. The work was also supported as part of the Congressionally Directed Medical Research Programs (CDMRP) award W81XWH-17-1-0347. We would also like to thank Dr. James Hanley for his support and discussions while extending the case-base methodology.

Author contributions

J.I. conceived the proof of concept, developed the theoretical formalism, developed the neural network code base, wrote the majority of the manuscript and performed the simulations and analyses. M.T. wrote the majority of the casebase sampling section 2.1. M.T. and S.B. provided key insights into the core sampling technique (casebase). R.S. provided multiple edits to the manuscript structure and key insights. All authors discussed the results and contributed to the final manuscript.

Financial disclosure

This work was supported by subcontracts from UM1DK078616 and R01HL151855 to Dr. Rob Sladek.

Conflict of interest

The authors declare no potential conflict of interests.

APPENDIX

References

Allaire, J., & Chollet, F. (2021). keras: R interface to 'keras' [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=keras` R package version 2.7.0.

Bhatnagar, S. R., Turgeon, M., Islam, J., Hanley, J. A., & Saarela, O. (2020). casebase: An alternative framework for survival analysis and comparison of event rates. *arXiv preprint arXiv:2009.10264*.

Blanche, P., Proust-Lima, C., Loubere, L., Berr, C., Dartigues, J.-F., & Jacqmin-Gadda, H. (2015). Quantifying and comparing dynamic predictive accuracy of joint models for longitudinal marker and time-to-event in presence of censoring and competing risks. *Biometrics*, *71*(1), 102–113.

Brilleman, S. L., Wolfe, R., Moreno-Betancur, M., & Crowther, M. J. (2020). Simulating survival data using the simsurv R package. *Journal of Statistical Software*, *97*(3), 1–27. doi: 10.18637/jss.v097.i03

Coradini, D., Daidone, M. G., Boracchi, P., Biganzoli, E., Oriana, S., Bresciani, G., … Marubini, E. (2000). Time-dependent relevance of steroid receptors in breast cancer. *Journal of clinical oncology*, *18*(14), 2702–2709.

Dispenzieri, A., Katzmann, J. A., Kyle, R. A., Larson, D. R., Therneau, T. M., Colby, C. L., … others (2012). Use of nonclonal serum immunoglobulin free light chains to predict overall survival in the general population. In *Mayo clinic proceedings* (Vol. 87, pp. 517–523).

Faraggi, D., & Simon, R. (1995). A neural network model for survival data. *Statistics in medicine*, *14*(1), 73–82.

Gerds, T. A., & Kattan, M. W. (2021). *Medical risk prediction models: With ties to machine learning (1st ed.)*. Chapman and Hall/CRC. Retrieved from `https://doi.org/10.1201/9781138384484` R package version 2021.10.10.

Graf, E., Schmoor, C., Sauerbrei, W., & Schumacher, M. (1999). Assessment and comparison of prognostic classification schemes for survival data. *Statistics in medicine*, *18*(17-18), 2529–2545.

Gulli, A., & Pal, S. (2017). *Deep learning with keras*. Packt Publishing Ltd.

Hanley, J. A., & Miettinen, O. S. (2009). Fitting smooth-in-time prognostic risk functions via logistic regression. *The International Journal of Biostatistics*, *5*(1).

Hughes-Hallett, D., Gleason, A. M., & McCallum, W. G. (2020). *Calculus: Single and multivariable*. John Wiley & Sons.

Jackson, C. (2016). flexsurv: A platform for parametric survival modeling in R. *Journal of Statistical Software*, *70*(8), 1–33. R package version 2.0. doi: 10.18637/jss.v070.i08

Kattan, M. W., & Gerds, T. A. (2018). The index of prediction accuracy: an intuitive measure useful for evaluating risk prediction models. *Diagnostic and prognostic research*, *2*(1), 1–7.

Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., & Kluger, Y. (2018). Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, *18*(1), 24.

Kleinbaum, D. G., & Klein, M. (2012). *Survival analysis: a self-learning text* (Vol. 3). Springer.

Kyle, R. (1997). Long term survival in multiple myeloma. *New Eng J Medicine*.

Lee, C., Zame, W. R., Yoon, J., & Schaar, M. v. d. (2018). Deephit: A deep learning approach to survival analysis with competing risks. *AAAI Conference on Artificial Intelligence*, 2314–2321. Software from pycox version 0.2.2.

Lu-Yao, G. L., Albertsen, P. C., Moore, D. F., Shih, W., Lin, Y., DiPaola, R. S., … others (2009). Outcomes of localized prostate cancer following conservative management. *Jama*, *302*(11), 1202–1209.

Mantel, N. (1973). Synthetic retrospective studies and related topics. *Biometrics*, 479–486.

Moore, D. F. (2016). asaur: Data sets for "applied survival analysis using r"" [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=asaur` R package version 0.50.

Nagpal, C., Li, X. R., & Dubrawski, A. (2021). Deep survival machines: Fully parametric survival regression and representation learning for censored data with competing risks. *IEEE Journal of Biomedical and Health Informatics*. Software downloaded March 10, 2021.

R Core Team. (2021). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Royston, P., & Parmar, M. K. (2002). Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in medicine*, *21*(15), 2175–2197.

Saarela, O., & Hanley, J. A. (2015). Case-base methods for studying vaccination safety. *Biometrics*, *71*(1), 42–52.

Terry M. Therneau, & Patricia M. Grambsch. (2000). *Modeling survival data: Extending the Cox model*. New York: Springer. R package version 3.2-11.

Ushey, K., Allaire, J., & Tang, Y. (2021). reticulate: Interface to 'python' [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=reticulate` R package version 1.22.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace. Version 3.6.13.

Zhang, Y., Tiňo, P., Leonardis, A., & Tang, K. (2021). A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *5*(5), 726–742.

AUTHOR BIOGRAPHY

Jesse Islam A PhD Candidate in Quantitative life sciences, studying complex relationships in data in a survival context.