Survival Analysis and Machine Learning in R

Rugile Ulianskaite

Contents

1 Introduction **3**

2 Key Concepts in Survival Analysis **3**

3 Survival Data **3**

3.1 Example Data 3

3.2 Defining the Survival Object 4

4 Descriptive Methods **5**

4.1 Kaplan-Meier Estimator 5

4.2 Nelson-Aalen Estimator 6

4.3 Hazard Function Estimation 7

5 Regression Models **8**

5.1 Semi-Parametric Models 8

5.2 Parametric Models 12

5.3 Other Regression Models 13

6 Advanced Data Structures **13**

6.1 Time-Varying Covariates 13

6.2 Competing Risks 16

7 Advanced Models **17**

7.1 Forests 17

7.2 Deep Learning 20

8 Model Evaluation and Comparison **23**

8.1 C-Index 23

8.2 Brier Score 24

9 Conclusion **25**

10 References **26**

A Appendix **28**

1 Introduction

Survival analysis is a statistical methodology used to analyze time-to-event data, focusing on the time until a specific event occurs. This technique is invaluable across various fields, including medicine, epidemiology, engineering, and economics, due to its ability to handle complex scenarios such as censored observations or time-dependent covariates.

With advancements in computational power and the rapid growth of statistical software, survival analysis has become increasingly accessible. The rise of big data and machine learning has further enhanced its applicability and popularity, allowing for more sophisticated analyses and insights.

This tutorial aims to provide a comprehensive guide to understanding survival analysis in R. We will delve into the essential R packages tailored for survival analysis, exploring their functionalities and applications.

2 Key Concepts in Survival Analysis

Survival data of the form (Y, δ).

$$Y = \min(T, C)$$

is the survival time where T is the duration from a well-defined time origin until the event of interest occurs, and C is the censoring time, which is a form of missing data where the event of interest is not observed for all subjects, often due to the end of study or loss to follow-up.

$$\delta = \begin{cases} 1 & \text{if } T \leq C \\ 0 & \text{if } T > C \end{cases} \tag{1}$$

is the status indicator used to distinguish between observed events and censored observations. $\delta = 1$ suggests that the event of interest has occurred, while $\delta = 0$ indicates that the event has been censored.

There are two fundamental concepts used to perform survival analysis. The *survival function*, $S(t) = \Pr(T > t)$, is the probability of an individual surviving beyond a time t and the *hazard function*,

$$h(t) = \lim_{\Delta t \to 0} \frac{\Pr(t < T \leq \Delta t + t | T > t)}{\Delta t},$$

is the instantaneous rate of occurrence of the event at a given time, t, conditional on survival up to that time. It can be approximated using the *probability density function* of T, $f(t)$, and the survival function [1]:

$$h(t) \approx \frac{f(t)}{S(t)}.$$

3 Survival Data

In this section, we explore the fundamental components of survival data in R and provide the necessary foundation for understanding and manipulating survival data, setting the stage for more advanced analyses in subsequent sections.

3.1 Example Data

To demonstrate survival analysis techniques in R, we need a dataset containing survival data of the form $(\mathbf{x}_i, Y_i, \delta_i)$, where $\mathbf{x}_i$ is the vector of covariates for the ith observation. While real-world datasets are often used in tutorials, simulated data can offer several advantages, particularly for method validation as well as reproducibility of the study.

The `coxed` [2] package in R provides a powerful tool for simulating survival data. As highlighted by the authors of the package in their paper *Simulating Duration Data for the Cox Model*, typical methods for simulating survival data use specific distribution. This assumption of functional form may introduce bias and and reduce a simulation's generalizability [3]. `coxed` addresses these limitations by using an iterative method that relies on fitting a cubic spline to randomly generated data points at each iteration, thus avoiding relying on a specific functional form for data generation.

```
library(coxed)
set.seed(311)
#Simulate survival data
sim_data <- sim.survdata(N = 500,             # Number of observations
                         T = 100,             # Maximum time
                         beta = c(.6,-.2,.4), # Adjust covaruate effect sizes
                         censor = 0.25)        # Censoring rate
```

This code generates a dataset with 500 observations, three covariates and a 25% censoring rate. The data frame includes the covariates **x**, the observed time y and the censoring indicator δ (`failed`). We also manually assign each subject a `sex` covariate to showcase how some of the functions apply to factors.

```
#View the first few lines of our dataset
head(sim_data$data)
```

```
##            X1          X2         X3  y failed    sex
## 1 -0.18532386 -0.31038844  0.8298739  1      1   Male
## 2  0.32090922 -0.45568030 -0.3207149  1      0   Male
## 3  0.01647055 -0.04646402  0.5553772 46      0 Female
## 4 -0.52798978  0.24118925 -0.1233927 90      1 Female
## 5 -1.09182274 -0.73011122  0.5553614 91      1 Female
## 6 -0.16937521  0.01426309  0.1520187 27      1 Female
```

By using simulated data, we can ensure that our tutorial examples are clear, reproducible, and demonstrate the full capabilities of the. survival analysis packages in R. In the subsequent sections, we will use this simulated dataset to illustrate various survival analysis techniques and their implementation.

3.2 Defining the Survival Object

The `survival` package is the core of survival analysis in R [4]. Work on this package began in 1985 and, over time, it has become the foundation upon which many other R packages for survival analysis are built. The `survival` package provides essential tools and data structures for conducting survival analysis, especially the ability to define survival objects. We first focus on classic survival analysis, which typically deals with a single type of event occurring at most once per subject. The first step in any survival analysis using R is to create a survival object which allows the models to handle censored observations. It includes the key components of survival data, (Y, δ):

```
library(survival)

time <- sim_data$data$y       # Y, observed times, as time
event <- sim_data$data$failed # delta, censoring indicator, as event

#Define the Survival Object
survival_object <- Surv(time, event)
```

4 Descriptive Methods

Descriptive methods in survival analysis provide essential tools for summarizing and visualizing the distribution of survival times. These methods allow researchers to intuitively understand the survival data. By offering graphical representations and stable estimates, these techniques improve the interpretability of survival data and facilitate meaningful insights into the underlying risk patterns.

4.1 Kaplan-Meier Estimator

The Kaplan-Meier estimator, developed in 1958, is one of the foundational tools in survival analysis. As a non-parametric technique it estimates the survival function with minimal data requirements, needing only the event times and censoring status. The estimator is defined as

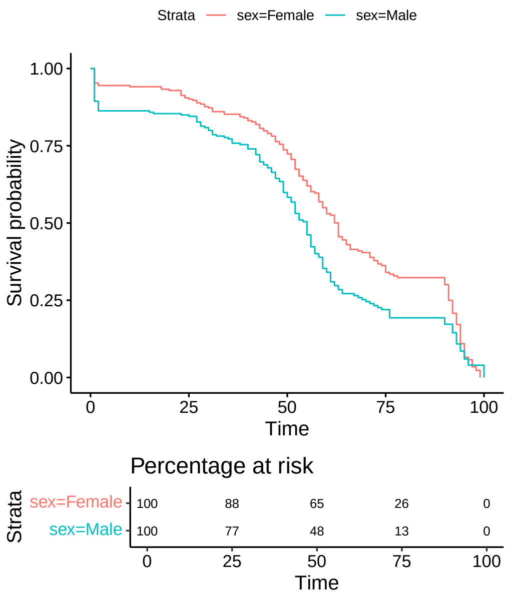
$$\hat{S}(d_k) = \prod_{j=1}^{k} \frac{r_j - q_j}{r_j},$$

where $d_1, ..., d_k$ are the k unique event times for non-censored subjects, r_j is the *risk set* (the number of subjects at risk and in the study right before d_k), and q_j is the number of events at the time d_k [1]. The resulting survival curve has a step-like shape due to its discrete time nature.

To apply the Kaplan-Meier estimator in R, we use the `survival` package [4]. We also enhance the interpretability of survival plots using the `survminer` package, an extension of `ggplot2` designed specifically to handle survival data [5].

```r
library(survminer)
# Fit the Kaplan-Meier survival curves for different sexes
km_fit <- survfit(Surv(time,event) ~ sex, data = sim_data$data)

#Create a plot
ggsurvplot(km_fit, data = sim_data$data, risk.table = 'percentage', size = 0.5,
           censor = FALSE, risk.table.height = 0.25, risk.table.fontsize = 3)
```



The main strength of the survival curves produced using the Kaplan-Meier estimator is their ability to provide a clear graphical representation of the survival data. This graphical approach is particularly useful for comparing survival curves across different groups. In the example, we quickly notice that females tend to have a higher probability of survival. To formally compare survival curves between groups, we can use the log-rank test, implemented through the `survdiff` function. This test assesses whether there is a statistically significant difference between two or more survival curves.

```r
survdiff(Surv(time,event) ~ sex, data = sim_data$data)
```

```
## Call:
## survdiff(formula = Surv(time, event) ~ sex, data = sim_data$data)
##
##                N Observed Expected (O-E)^2/E (O-E)^2/V
## sex=Female 255      196      225      3.72        10
## sex=Male   245      180      151      5.54        10
##
##  Chisq= 10  on 1 degrees of freedom, p= 0.002
```

The output confirms what was visible in the Kaplan-Meier curves - there seems to be a significant difference between the survival of males and females (`p= 1e-06`).

The Kaplan-Meier estimator is primarily descriptive in nature and does not control for covariates. Interpretation of the curves can be challenging, particularly with small sample sizes or in cases of heavy censoring. Furthermore, the estimator represents population averages and cannot predict individual outcomes [6]. Despite these limitations, the Kaplan-Meier estimator is an essential initial exploratory technique before moving on to more complex models. The Kaplan-Meier estimator is closely linked to the Nelson-Aalen estimator discussed below.

4.2 Nelson-Aalen Estimator

The Nelson-Aalen estimator is a non-parametric method used to estimate the *cumulative hazard function*,

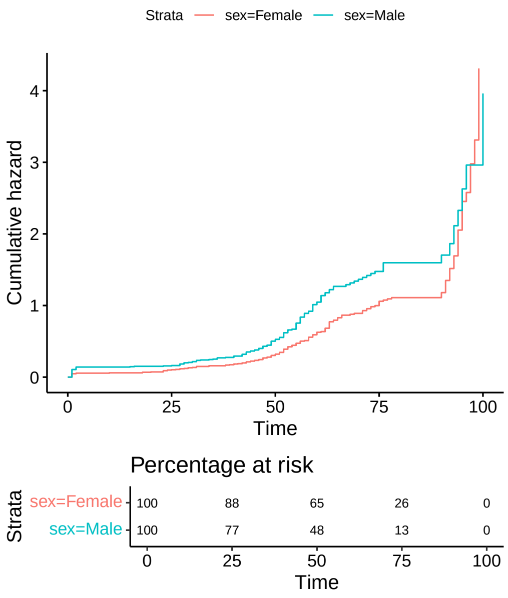
$$H(t) = \int_0^t h(s)ds,$$

from censored survival data. It is defined as

$$\hat{H}(t) = \sum_{d_j \leq t} \frac{q_j}{r_j},$$

where q_j is the number of events at time d_j and r_j is the risk set at that time [7].

```r
# Fit the curve using the Nelson-Aalen estimator
na_fit <- survfit(Surv(time,event) ~ sex, data = sim_data$data,
                  type = "fleming-harrington")
# Create plot
ggsurvplot(na_fit, data = sim_data$data, fun = "cumhaz", risk.table = 'percentage',
           size = 0.5, censor = FALSE, risk.table.height = 0.25, risk.table.fontsize = 3)
```



While this curve can be particularly useful for checking the fit of parametric models, the Nelson-Aalen and Kaplan-Meier estimators are similar as they fit the cumulative hazard, $H(t)$, and the survival, $S(t)$ functions, respectively, which are related as follows:

$$H(t) = -\ln S(t) \text{ and } S(t) = \exp(-H(t))$$

Due to this similarity, we will not delve into further detail about the Nelson-Aalen estimator. For additional information, refer to *Theory and Applications of Hazard Plotting for Censored Failure Data* [8] and *Nelson–Aalen Estimator* [7].

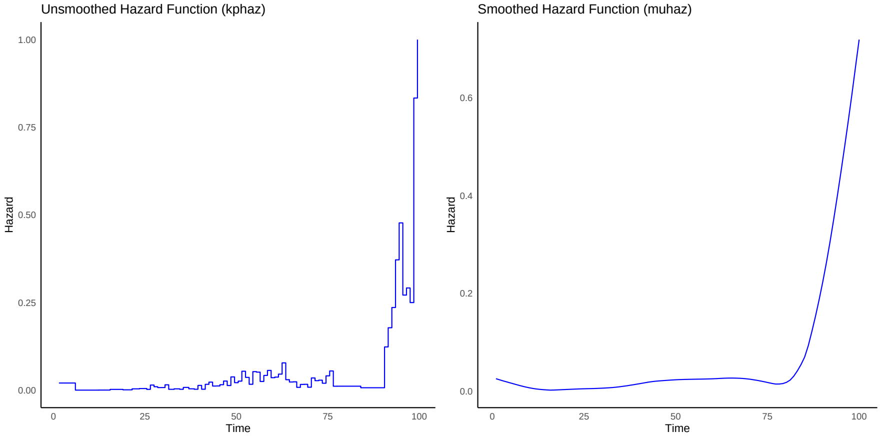
4.3 Hazard Function Estimation

The survival function is crucial in survival analysis as it helps us understand how the risk of an event changes over time, which can aid in identifying periods of high risk and when intervention might be the most effective. As the hazard function represents the instantaneous rate of failure at time t given survival up to that time, calculating it directly from data is challenging as the results can be unstable, especially with semi-parametric or non-parametric estimation. To address this, we use smoothing techniques, which help to reveal the underlying trend in the hazard over time by reducing the noise in the raw data [9].

The `muhaz` package in R provides a non-parametric method for estimating the hazard function using kernel-based smoothing techniques [10]. By providing stable estimates, the package enhances our understanding of how the risk of an event changes over time.

```
library(muhaz)
# Fit hazard function without smoothing
kphaz <- kphaz.fit(time, event)
# Fit hazard function with smoothing
muhaz <- muhaz(time, event, min.time = min(time), max.time = max(time))
```

The `kphaz` are the Kaplan-Meier type hazard function estimates, while the `muhaz` are the kernel-smoothed hazard function estimates, where the kernel smoothing intuitively averages nearby data points to create a clearer and more stable representation of the risk over time:



These visualizations of the hazard functions allow researchers to identify periods of high and low risk, detect potential time-dependent effects, and compare risk patterns across different groups, complementing other descriptive methods.

5 Regression Models

Regression models in survival analysis are powerful tools that allow researchers to examine the relationship between various factors and the time until an event of interest occurs. These models, which can generally be classified into semi-parametric and parametric models, enable the estimation of how different covariates affect survival probabilities and hazard rates.

5.1 Semi-Parametric Models

Semi-parametric models, particularly the Cox proportional hazards model, are widely used in survival analysis due to their flexibility in handling covariates without specifying the baseline hazard function.

5.1.1 Traditional Cox PH Model

The Cox Proportional Hazards (PH) model, introduced by Cox (1972) [11], is one of the fundamental models in survival analysis due to its semi-parametric nature. The model is based on the *proportional hazards assumption*,

$$h(t|\mathbf{x}_i) = h_0(t)\exp(\mathbf{x}_i^T\beta),$$

where $h_0(t) \geq 0$ is the unspecified *baseline hazard* for the covariate $\mathbf{x} = \mathbf{0}$. The term $\exp(\mathbf{x}_i^T\beta)$ denotes the *relative risk* associated with the covariate $\mathbf{x}_i$ of the observation i, with regression coefficients β. The proportional hazards assumption implies that while hazard functions for different individuals can vary over time, they remail proportional to each other.

A key feature of the Cox model is its ability to cancel out the baseline hazard $h_0(t)$ during estimation, focusing only on the relative risks. The *partial likelihood function* used for estimating the regression coefficients for distinct failure times t_i is given by

$$\text{PL}(\beta) = \prod_{i:\delta_i=1} \frac{\exp(\mathbf{x}_i^T\beta)}{\sum_{\mathbf{x}_j \in R(t_i)} \exp(\mathbf{x}_j^T\beta)},$$

where $R(t_i)$ is the risk set at time t_i. It can be thought of as the relative risk over the estimate of total hazard at the time t_i for the risk set. Maximizing this partial likelihood function allows for the estimation of the regression coefficients β without needing to specify the baseline hazard function [1].

The implementation of the Cox PH model using the `coxph` function from the `survival` package [4] is quite simple:

```
# Fit the model
cph_fit <- coxph(Surv(time, event) ~ X1 + X2 + X3 + sex, data = sim_data$data,
                 x = TRUE, y = TRUE)
# View the results
summary(cph_fit)
```

```
## Call:
## coxph(formula = Surv(time, event) ~ X1 + X2 + X3 + sex, data = sim_data$data,
##     x = TRUE, y = TRUE)
##
##   n= 500, number of events= 376
##
##             coef exp(coef) se(coef)      z Pr(>|z|)
## X1       0.55722   1.74582  0.09728  5.728 1.02e-08 ***
## X2      -0.15074   0.86007  0.10950 -1.377   0.1686
## X3       0.28649   1.33174  0.10033  2.855   0.0043 **
## sexMale  0.26619   1.30498  0.10527  2.529   0.0115 *
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
##         exp(coef) exp(-coef) lower .95 upper .95
## X1         1.7458     0.5728     1.443     2.113
## X2         0.8601     1.1627     0.694     1.066
## X3         1.3317     0.7509     1.094     1.621
## sexMale    1.3050     0.7663     1.062     1.604
##
## Concordance= 0.61  (se = 0.017 )
## Likelihood ratio test= 47.76  on 4 df,   p=1e-09
## Wald test            = 48.99  on 4 df,   p=6e-10
## Score (logrank) test = 49.33  on 4 df,   p=5e-10
```

In the resulting summary, we see that the model was fitted using 500 observations, of which 376 experienced the event of interest.

The regression coefficients give us information on the effects and significance of covariates, for example, the covariate `sexMale` has a positive effect (`coef = 0.26619`), meaning that being male is associated with an increased hazard compared to being female. The hazard ration (`exp(coef) = 1.3050`) indicates that the risk for males is about 1.3 times the risk for females. The concordance index, discussed in Section 8.1, is 0.61 suggesting moderate predictive ability of the model.

Testing the proportional hazards assumption is crucial as it validates a key assumption of the Cox model that ensures its accuracy and reliability. We can test the assumption using the `cox.zph` function from the `survival` package:

```
# Test the PH assumption
cox.zph(cph_fit)
```

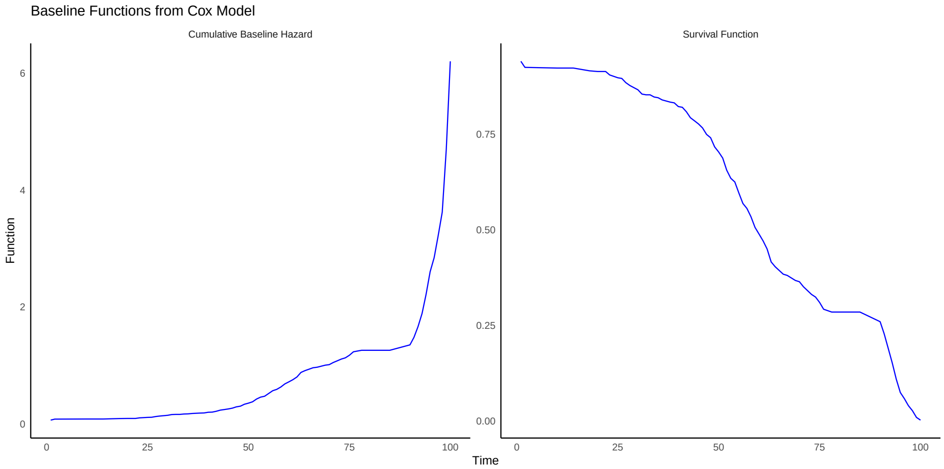
```
##         chisq df    p
## X1     0.1747  1 0.68
## X2     0.0796  1 0.78
## X3     0.1105  1 0.74
## sex    4.7148  1 0.03
## GLOBAL 5.1370  4 0.27
```

The results of the proportional hazards test indicate that there is no evidence that the covariates X1, X2 and X3 violate the proportional hazards assumption at a 5% significance level. However, the p-value for sex is well below 0.05, which suggests some evidence against the PH assumption for this covariate, indicating that the effects of sex on the hazard might change over time.

The Cox PH model estimates the hazard function, the instantaneous rate of failure, which can be vague and difficult to understand intuitively. To facilitate interpreting the results from a Cox model, we can convert them to duration-based quantities using the coxed package [2].

```r
# Convert the results to duration-based quantities
cph_durfit <- coxed(cph_fit)
```

The cph_durfit object is of class coxedExpdur and it contains exp.dur, the expected survival times for each individual based on the model, mean, median, and a data frame baseline.functions with the cumulative baseline hazard function and the baseline survival function in the data:



Using the coxed function on our model proved beneficial as it allowed us to translate abstract hazard ratios into more interpretable expected survival times. The resulting plots of the cumulative baseline hazard and survival functions offer a visual representation of the underlying risk patterns over time. Additionally, expected durations for each individual (exp.dur) enable us to make more personalized assessments of survival prospects.

The Cox Proportional Hazards model offers several advantages. Its semi-parametric flexibility does not require us to make assumptions about the functional form of the baseline hazard, making the model adaptable to various types of survival data. The model also demonstrates robustness to violations of its assumptions, particularly in large sample sizes.

Despite its strengths, the Cox model cannot directly estimate the survival function without additional assumptions about the baseline hazard and it also relies on the proportional hazards assumption, which may not always hold in practice. Furthermore, the Cox PH model does not handle multicollinearity effectively and is not well-suited for high-dimensional data without regularization techniques [12].

5.1.2 Regularized Cox Proportional Hazards Model

Regularized Cox Proportional Hazards (PH) models are an extension of the traditional Cox model, designed to handle high-dimensional data and prevent overfitting. The two main types of regularization are L1 (lasso) and L2 (ridge) regularization. L1 regularization can shrink some coefficients to exactly zero, effectively performing variable selection, while L2 regularization shrinks all coefficients towards but not exactly to zero, which can be useful for handling multicollinearity. We regularize the Cox model using the elastic net penalty, which is the combination of L1 and L2, combining the strengths of both and balancing between variable selection and coefficient shrinkage.

In the regularized Cox PH model, we maximize the penalized log-partial likelihood of the Cox model as described in Section 5.1.1:

$$\mathrm{PL}_{\text{log-reg}}(\beta) = \sum_{i:\delta_i=1} [\mathbf{x}_i^T \beta - \log(\sum_{\mathbf{x}_j \in R(t_i)} \exp(\mathbf{x}_j^T \beta))] - \lambda P(\beta),$$

where λ is the regularization parameter and $P(\beta)$ is the elastic net penalty function given by

$$P((\beta)) = \alpha \sum_{i=1}^{p} |\beta_i| + \frac{1}{2}(1-\alpha) \sum_{i=1}^{p} \beta_i^2,$$

where p is the number of predictors [13].

The `glmnet` package in R allows us to implement regularized Cox models. The `cv.glmnet` function can be used to fit these models with cross-validation to select the optimal regularization parameter [14]:

```r
library(glmnet)
# Prepare the matrix of covariates
sim_covariates <- as.matrix(cbind(sim_data$data[, c("X1", "X2", "X3")],
                                  sex = as.numeric(sim_data$data$sex == "Male")))
# Fit regularized Cox odel with cross-validation
set.seed(512)
regcox_fit <- cv.glmnet(sim_covariates, Surv(time,event),
                        family = "cox", alpha = 0.5, # Alpha for elastic net
                        type.measure = "C")          # Model evaluation with C-Index
# View the results
coef(regcox_fit)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##              1
## X1  0.26562764
## X2  .
## X3  0.02505557
## sex 0.04841265
```

The model selected three out of four covariates as significant predictors. Here, `sex` shows a slight positive association with hazard (coefficient 0.04841265) indicating a marginally better survival for females as we have used `Male = 1` in the covariate matrix.

The regularized Cox model requires careful tuning of the regularization parameter, λ, and the mixing parameter, α, which can be computationally intensive [15]. Moreover, the model still relies on the proportional hazards assumption, which may not always hold in practice. As highlighted by the authors of the `glmnet` package, the standard implementation of regularized Cox models does not handle time-dependent covariates as effectively as some other approaches [13]. This becomes particularly significant in scenarios where risk factors change over the course of the study, potentially leading to biased estimates and reduced predictive accuracy.

5.2 Parametric Models

Parametric models in survival analysis require assumptions about the underlying distribution of survival times, but they offer several benefits, including the ability to represent complex data structures and avoid the black-box effect (where the relationship between inputs and outputs is not easily interpretable) of some non-parametric models [16]. These models give the mathematical forms of the survival and hazard functions, allowing for more straightforward interpretation and extrapolation beyond the observed data.

One of the models that rivals the Cox PH model in popularity is the Accelerated Failure Time (AFT) model. The AFT model assumes that the logarithm of the survival time, $Y = \log(T)$, can be expressed as

$$Y = \mathbf{x}^T \beta + W,$$

where W is the error variable in the density function f. This leads to the following model for survival time:

$$T = S \exp(\mathbf{x}^T \beta),$$

where $S = \exp(W) > 0$ models the baseline hazard [17]. The AFT model directly models the survival time rather than the hazard, proving insights into how different factors accelerate or decelerate the time to an event. In this model, a one-unit change in a covariate x_i results in a change in the survival time by a factor of $\exp(\beta_i)$. For instance, if $\exp(\beta_i) > 0$, the event is accelerated, leading to a shorter survival time.

We illustrate the AFT model using the `survreg` function in the `survival` package [4]:

```r
# Fit the AFT model
aft_fit <- survreg(Surv(time, event) ~ X1 + X2 + X3 + sex , data = sim_data$data,
                   dist = "weibull") # Specify for the AFT model
# View results
summary(aft_fit)
```

```
##
## Call:
## survreg(formula = Surv(time, event) ~ X1 + X2 + X3 + sex, data = sim_data$data,
##     dist = "weibull")
##               Value Std. Error     z      p
## (Intercept)  4.2425     0.0466 91.03 <2e-16
## X1          -0.2008     0.0633 -3.17 0.0015
## X2           0.0682     0.0683  1.00 0.3183
## X3          -0.1057     0.0646 -1.64 0.1018
## sexMale     -0.1400     0.0676 -2.07 0.0383
## Log(scale)  -0.4389     0.0465 -9.43 <2e-16
##
## Scale= 0.645
##
## Weibull distribution
## Loglik(model)= -1919.3   Loglik(intercept only)= -1929
##  Chisq= 19.32 on 4 degrees of freedom, p= 0.00068
## Number of Newton-Raphson Iterations: 8
## n= 500
```

The results can be interpreted to understand how each covariate affects the survival time. The model's overall fit is highly significant ($p = 0.00068$), indicating that the included variables collectively explain a substantial portion of the variation in survival times compared to an intercept-only model.

Another package that can be useful for fitting parametric survival models is the `flexsurv` package [18]. While `survreg` in `survival` package includes includes distributions such as Weibull, exponential, log-normal, log-logistic, the `flexsurvreg` function extends this to generalized gamma, gamma, Gompertz, generalized F as well as the option for the user to specify their own distribution, provided that they define the density or hazard function. Since `flexsurvreg` functions in a similar manner to `survreg`, we do not include an example. For detailed guidance on specifying distributions or using the Royston and Parmar spline-based model, please refer to the `flexsurv` documentation [18], [16].

Parametric models have limitations in handling time-varying covariates and require strong assumptions about the underlying distribution of survival times. However, they are particularly valuable for when researchers seek to make precise predictions beyond the observed data range, desire easily interpretable coefficients, or need to estimate complex quantities such as median survival times, especially in situations where the data closely follows a known parametric distribution.

5.3 Other Regression Models

Several other regression models are available for survival analysis, each suited for different scenarios and data structures.

The **Proportional-Odds model**, also known as the cumulative odds model, assumes that the effect of covariates on the odds of survival beyond a time is constant [19]. It is particularly useful when the proportional hazards assumption is violated or the hazard ratios between groups converge over time.

The **Proportional Excess Risk model** is designed for situations where there is a background mortality rate that is unrelated to the exposure of interest. This model assumes that the excess risk due to exposure is proportional to a known baseline excess risk [19]. It is commonly used in cancer epidemiology studies where researchers aim to separate the effect of a specific exposure from the background mortality risk.

The **Two-Stage model** is a flexible approach that combines elements of both parametric and semi-parametric models. In the first stage, a parametric model is fitted to estimate the baseline hazard, while in the second stage, a Cox-type model is used to estimate the effects of covariates [19]. This model is useful when there is prior knowledge about the shape of the baseline hazard but flexibility is needed to model covariate effects.

For more information on these models, refer to the textbook *Dynamic Regression Models for Survival Data* [19]. These models can be implemented using the `timereg` package based on this textbook [20], which is introduced in 6.1.

6 Advanced Data Structures

Real-world survival data often exhibits complex characteristics beyond simple time-to-event scenarios. Time-varying covariates and competing risks are two structures requiring specialized analytical approaches.

6.1 Time-Varying Covariates

Time-varying or time-dependent covariates are variables whose values can change over the course of the study or follow-up period. These covariates are particularly relevant in Cox regression models as they can significantly impact the estimation of hazard functions and survival probabilities. There are two main types of time-varying covariates: internal and external. Internal covariates are influenced by the survival status of the subject and are observed as long as the subject remains in the study, such as blood pressure in a clinical trial. External covariates are not influenced by the survival status and are predetermined or fixed, such as patient age [21].

When using time-dependent covariates, it is essential to handle the data appropriately. The data should be arranged in a counting process style, where each time interval is treated as a separate observation with its corresponding covariate values [2]. We can simulate such data using the `coxed` package [2]:

```
set.seed(61)
# Simulate survival data with time-varying covariates
tvc_sim_data <- sim.survdata(N = 500, T = 100, beta = c(.6, -.2, .4), censor = .25,
                             type = "tvc") # Specify time-varying covariates
# View the first few lines of the simulated dataset
head(tvc_sim_data$data)
```

```
##   id failed start end         X1        X2        X3
## 1  1      0     0   1 -1.9715212 0.6685945  1.410535
## 2  1      0     1   2 -0.5247429 0.6685945  1.410535
## 3  1      0     2   3 -1.0164355 0.6685945  1.410535
## 4  1      0     3   4 -1.6828355 0.6685945  1.410535
## 5  1      1     4   5  0.8028387 0.6685945  1.410535
## 6  2      0     0   1  0.1316654 1.0062156 -1.386619
```

```
# We will be interested in start, end of time interval and censoring status
start <- tvc_sim_data$data$start
end <- tvc_sim_data$data$end
event <- tvc_sim_data$data$failed
```

The Cox model can incorporatee time-varying covariates, allowing for dynamic analysis of factors that change over time. We use the `survival` package [4]:

```
# Fit the Cox model to TVC data
tvc_cph_fit <- coxph(Surv(start, end, event) ~ X1 + X2 + X3,
                     data = tvc_sim_data$data)
# View the results
summary(tvc_cph_fit)
```

```
## Call:
## coxph(formula = Surv(start, end, event) ~ X1 + X2 + X3, data = tvc_sim_data$data)
##
##   n= 11124, number of events= 316
##
##        coef exp(coef) se(coef)      z Pr(>|z|)
## X1  0.59179   1.80723  0.05748 10.296  < 2e-16 ***
## X2 -0.19277   0.82467  0.05442 -3.542 0.000397 ***
## X3  0.52490   1.69028  0.06814  7.703 1.32e-14 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
##    exp(coef) exp(-coef) lower .95 upper .95
## X1    1.8072     0.5533    1.6147    2.0227
## X2    0.8247     1.2126    0.7412    0.9175
## X3    1.6903     0.5916    1.4790    1.9318
##
## Concordance= 0.707  (se = 0.018 )
## Likelihood ratio test= 172.7  on 3 df,   p=<2e-16
## Wald test            = 172  on 3 df,   p=<2e-16
## Score (logrank) test = 170.9  on 3 df,   p=<2e-16
```

The interpretation of the results is similar to that in section 5.1.1.

Aalen's Additive Hazards model is an alternative to the Cox PH model. It is defined as

$$h(t|\mathbf{x}(t)) = h_0(t) + \mathbf{x}(t)^T \beta(t),$$

where $h_0(t)$ is the baseline hazard and $\beta(t)$ are time-varying regression coefficients. This model is particularly useful when the proportional hazards assumption is violated or when the effect of covariates is expected to change over time [19]. We fit this model using the `aalen` function in the `timereg` package [20]:

```
library(timereg)
# Fit the Aalen's Additive Hazards model
aalen_fit <- aalen(Surv(start, end, event) ~ X1 + X2 + X3,
                   data = tvc_sim_data$data)
# View results
aalen_fit
```

```
## Additive Aalen Model
##
## Test for nonparametric terms
##
## Test for non-significant effects
##              Supremum-test of significance p-value H_0: B(t)=0
## (Intercept)                         14.30               0.000
## X1                                   9.43               0.000
## X2                                   4.00               0.001
## X3                                   6.93               0.000
##
## Test for time invariant effects
##                    Kolmogorov-Smirnov test p-value H_0:constant effect
## (Intercept)                         2.040                       0.000
## X1                                  0.750                       0.010
## X2                                  0.398                       0.154
## X3                                  0.971                       0.002
##                      Cramer von Mises test p-value H_0:constant effect
## (Intercept)                         93.50                       0.000
## X1                                   5.74                       0.065
## X2                                   3.18                       0.064
## X3                                  15.80                       0.002
##
##
##
##   Call:
## aalen(formula = Surv(start, end, event) ~ X1 + X2 + X3, data = tvc_sim_data$data)
```

The Kolmogorov-Smirnov and Cramer von Mises tests indicate that the effects of the intercept, `X1`, and `X3` are not constant over time ($p < 0.05$), suggesting these covariates have time-varying effects on the hazard, which highlights the model's ability to capture dynamic relationships between covariates and survival outcomes.

We also mention the Cox-Aalen model, which combines the features of both the Cox proportional hazards mopdel and Aalen's additive model and is defined as:

$$h(t|\mathbf{x}(t), \mathbf{z}(t)) = Y_i(t)[h_0(t) + \mathbf{x}(t)^T \alpha(t)] \exp(\mathbf{z}(t)^T \beta(t)),$$

where $Y_i(t)$ is the at-risk indicator, $\mathbf{x}(t)$ are covariates with time-varying effects, and $\mathbf{z}(t)$ are covariates with proportional effects [19]. This model is particularly useful when some covariates are believed to have time-varying effects while others are assumed to have proportional effects.

```r
# Fit the Cox-Aalen Model
coxaalen_fit <- cox.aalen(Surv(start, end, event) ~ X1 + prop(X2) + X3,
                          data = tvc_sim_data$data)
# View the results
summary(coxaalen_fit)
```

```
## Cox-Aalen Model
##
## Test for Aalen terms
## Test not computed, sim=0
##
## Proportional Cox terms :
##           Coef.     SE Robust SE D2log(L)^-1     z    P-val lower2.5%
## prop(X2) -0.212 0.0539    0.0496      0.0548 -4.28 1.89e-05    -0.318
##          upper97.5%
## prop(X2)     -0.106
## Test of Proportionality
##          sup|  hat U(t) | p-value H_0
## prop(X2)            9.24        0.874
```

Here, we assume that `X2` has a proportional effect (specified by `prop()`), while `X1` and `X3` have time-varying effects. The test of proportionality ($p = 0.874$) supports the assumption of a constant effect for `X2` over time.

These advanced models offer greater flexibility in handling time-varying covariates. For more information on these models, refer to the `timereg` package documentation [20] and the textbook this package is based on [19].

6.2 Competing Risks

In survival analysis, a competing risk is an event whose occurrence precludes the occurrence of the primary event of interest [22]. This scenario involves multiple event types, with each subject experiencing at most one event. For example, in a study of cancer mortality, death from any other cause would be a competing risk. Competing risks are problematic because they can violate the assumption that censoring is non-informative and may lead to incorrect conclusions if not properly accounted for [22].

The *cumulative incidence function* for the kth type of event is defined as

$$\mathrm{CIF}_k(t) = \Pr(T \leq t, D = k),$$

where T is the time to event and D is the type of the event that has occurred [22].

The *cause-specific hazard*, the instantaneous rate of occurrence of the kth event in subjects who are currently event-free (no event of any type has occured), is defined as

$$h_k^{\mathrm{CS}}(t) = \lim_{\Delta t \to 0} \frac{\Pr(t < T \leq t + \Delta t, D = k | T \geq t)}{\Delta t}.$$

We also introduce the *sub-distribution* or *Fine-Gray hazard*, the instantaneous rate of the kth event type for subjects that have not had an event of **that specific type** before time t:

$$h_k^{\mathrm{FG}} = \lim_{\Delta t \to 0} \frac{\Pr(t < T \leq \Delta t + t, D = k \mid T \geq t \cup (T < t \cap D \neq k))}{\Delta t}$$

[23].

The `cmprsk` package in R is specifically designed for analyzing competing risks data. It provides tools to estimate cumulative incidence functions and perform regression analysis using the Fine-Gray model. This package focuses on cumulative incidence rather than cause-specific hazards, offering a more direct approach [24].

The `crr` function in the `cmprsk` package implements the Fine-Gray model for competing risks regression. It models the sub-distribution hazard of a particular type of even while accounting for competing risks. The Fine-Gray model is defined as:

$$h_k^{\mathrm{FG}}(t_i|\mathbf{x}_i) = h_{k0}^{\mathrm{FG}}(t_i)\exp(\mathbf{x}_i^T\beta_k)$$

where h_{k0}^{FG} is the baseline sub-distribution hazard and β_k are the regression coefficients for covariates $\mathbf{x}_i$. This approach allows for the estimation of covariate effects on the probability of a specific event type occurring before other competing events.

For a detailed understanding and examples of using `cmprsk`, please refer to the package documentation [24] and the paper *A Proportional Hazards Model for the Subdistribution of a Competing Risk* [25] for a better understanding of the model.

7 Advanced Models

Advanced machine learning models in survival analysis emerged to overcome the limitations of traditional methods, such as the Cox PH model, in handling censored data and complex datasets. These models were developed to enhance predictive accuracy and flexibility, particularly in fields like healthcare and finance, where precise event timing predictions are essential.

7.1 Forests

Forests are a powerful type of models for survival analysis, using ensemble learning techniques to effectively handle high-dimensional data and complex-interactions. These methods provide robust predictions while accommodating the challenges of censored data.

7.1.1 Random Survival Forests

Random Survival Forests (RSF) extend the random forest algorithm to survival analysis by constructing an ensemble of decision trees tailored for censored data. We construct a random survival forest as follows:

We draw B bootstrap samples from the original data and grow a survival tree for each. We calculate the cumulative hazard function for each node using the Nelson-Aalen estimator (4.2) and average to obtain the ensemble cumulative hazard function, providing a robust non-parametric estimate of the function over time [26].

Note that for each tree, the nodes are split from p randomly selected candidate variables using the log-rank test statistic, which maximizes survival differences in the child nodes. For more information on how this statistic is constructed, refer to Section 11.4 of *Introduction to Statistical Learning* [1].

We use the `randomForestSRC` package in R [27]. To improve model performance, we conduct a focused grid search over key hyperparameters, employing 5-fold cross-validation and using the C-index (8.1) as our performance metric. We use the results to build our model:

```r
library(randomForestSRC)
set.seed(710)
# Build ntree 500, nodesize 2 mtry 1 nsplit 10
rsf_fit <- rfrc(Surv(y, failed) ~ sex,
              data = sim_data$data,
              ntree = 500,
              nodesize = 2,
              mtry = 1,
              nsplit = 10,
              splitrule = "logrank")

print(rsf_fit)
```

```
##                          Sample size: 500
##                     Number of deaths: 376
##                      Number of trees: 500
##           Forest terminal node size: 2
##       Average no. of terminal nodes: 2
## No. of variables tried at each split: 1
##               Total no. of variables: 1
##       Resampling used to grow trees: swor
##    Resample size used to grow trees: 316
##                             Analysis: RSF
##                               Family: surv
##                       Splitting rule: logrank *random*
##       Number of random split points: 10
##                           (OOB) CRPS: 15.49790691
##                    (OOB) stand. CRPS: 0.15497907
##    (OOB) Requested performance error: 0.57400394
```

```r
# Calculate the C-index
get.cindex(rsf_fit$yvar[,1], rsf_fit$yvar[,2], rsf_fit$predicted.oob)
```

```
## [1] 0.5740039
```

Here, we get a C-index of 0.5740039, which indicates that the model performs better than random guessing.

RSF is particularly advantageous due to its ability to model complex interactions and non-linear relationships without assuming the functional form for the survival function. It is robust to overfitting, especially in high-dimensional settings. This makes RSF suitable for applications in fields like genomics, where the number of predictors is usually large relative to the number of observations.

Despite its strengths, RSF can be computationally intensive, particularly with large datasets, due to the need to grow a large number of trees and evaluate complex splitting criteria. The model also has a black-box nature common to many machine learning methods, making it difficult to discern the specific influence of individual predictors within the ensemble framework. Additionally, the choice of hyperparameters, such as the number of trees and node size, can significantly impact model performance and requires careful tuning.

7.1.2 Oblique Random Survival Forests

Oblique Random Survival Forests (ORSF) enhance the traditional RSF by using linear combinations of input variables (LCIV) at each node split, rather than single-predictor splits. The ORSF embeds a regularized

Cox PH model into the nonterminal nodes of the trees and then uses these models to construct LCIVs. This method allows ORSF to capture complex, multivariate interactions more effectively than axis-aligned splits [28].

Recently, the development of Accelerated Oblique Random Survival Forests (AORSF) has further improved the capabilities of ORSF. AORSF introduces techniques such as Newton-Raphson scoring to significantly speed up the computation process. Additionally, AORSF incorporates methods for assessing the importance of individual predictors, addressing the challenge of interpretability inherent in ORSF due to its use of linear combinations [29]. As the most recent version of this method, we demonstrate AORSF using the `aorsf` package [30], where the hyperparameters are selected through a grid search with 10-fold cross-validation with the C-indec (8.1) as the performance evaluation metric:

```r
library(aorsf)
set.seed(712)
# Fit the accelerated ORSF
aorsf_fit <- orsf(data = sim_data$data,
                  formula = Surv(y, failed) ~ .,
                  n_tree = 2000,
                  n_split = 5,
                  n_retry = 0,
                  mtry = 1,
                  split_rule = "logrank")

aorsf_fit
```

```
## ---------- Oblique random survival forest
##
##      Linear combinations: Accelerated Cox regression
##           N observations: 500
##                 N events: 376
##                  N trees: 2000
##       N predictors total: 4
##    N predictors per node: 1
##  Average leaves per tree: 6.597
## Min observations in leaf: 5
##       Min events in leaf: 1
##           OOB stat value: 0.59
##            OOB stat type: Harrell’s C-index
##      Variable importance: anova
##
## -----------------------------------------
```

ORSFs often provide higher prediction accuracy compared to traditional RSF as well as regression methods. Furthermore, ORSF can reduce the number of splits needed, potentially leading to more compact and efficient models [28].

However, ORSF can be more computationally demanding than RSF. It also shares the black-box nature of other ensemble methods, making it even less interpretable than its axis-based counterparts in terms of understanding the influence of individual predictors. Moreover, the complexity of the model may also lead to overfitting if not appropriately regularized [28] [29].

7.1.3 Conditional Inference Survival Forests

Conditional Inference Survival Forests (CISF) are an extension of the conditional inference framework to survival analysis. Unlike traditional survival forests, CISF uses permutation tests to select the best split ensuring that the selection process is unbiased with respect to the number of predictor variables [31]. For the sake of brevity, we will not demonstrate the use of CISF. For more information, refer to the `ICcforest` package documentation [32].

The method has been demonstrated to improve prediction performance in comparison to Random Survival Forests and traditional Cox models, especially when dealing with complex datasets involving multiple covariates and interactions [33]. However, CISF can be computationally intensive due to the need for permutation testing at each node, which may limit its applicability to very large datasets [31].

Choosing between RSF, ORSF, and CISF depends on the specific needs and characteristics of the data. If a method that can handle high-dimensional data with complex interactions and non-linear relationships is required, RSF is a robust choice, though it may be computationally intensive. CISF is preferable for unbiased variable selection and interpretability, especially in high-dimensional settings. ORSF is ideal for higher prediction accuracy when one can manage the computational demands.

7.2 Deep Learning

Deep Learning is a subset of machine learning that uses neural networks with multiple layers to model complex patterns in data, It has become more popular due to its ability to handle large and high-dimensional datasets. Survival analysis started using deep learning because traditional statistical models, like the Cox Proportional Hazards model, often assume linear relationships between features and outcomes, which may not capture complex interactions present in high-dimensional data such as medical images or genomic data.

7.2.1 DeepSurv

DeepSurv is a feed-forward deep neural network (DNN) for survival analysis that extends the Cox Proportional Hazards model. It is the first DNN designed specifically for survival analysis. DeepSurv replaces the linear predictor $\mathbf{x}_i^T\beta$ of the Cox model with a neural network that maps the covariates to a risk score. It models the log-hazard function as a neural network, where the network's output is used to estimate the hazard ratio. The model is trained by maximizing the partial likelihood function, similar to the traditional Cox model, but with the flexibility of a neural network to capture complex. non-linear relationships between covariates and the survival outcome [34].

Using the `dnn` package [35], we first need to build the neural network that will be used as a component for the `deepSurv` framework. The Rectified Linear Unit (ReLU) is used for the hidden layers to facilitate faster training and avoid the vanishing gradient problem. For the output layer, we have chosen a sigmoid activation function because it aligns with the needs of traditional survival analysis by effectively handling binary classification tasks. In cases involving competing risks or multi-state models, a softmax activation function might be a more appropriate output layer [36].

```r
library(dnn)
# Build a deep neural network model
dnn_model <- dNNmodel(N = 4,                      # Number of training sample
                      units = c(128, 64, 32, 1),  # Number of nodes in each layer
                      optimizer = "adam",
                      input_shape = 4,            # Number of covariates
                      activation = c("relu", "relu", "relu", "sigmoid")
                      )
```

Next, we use this DNN in the `deepSurv` function [35] with the arguments found through a grid search for hyperparameter optimization.

```
set.seed(721)
# Fit the deepSurv model
deepSurv_fit <- deepSurv(Surv(y, failed) ~ . ,
                         data = sim_data$data,
                         model = dnn_model,
                         epochs = 300,
                         batch_size = 32,
                         lr_rate = 1e-04,
                         alpha = 0.25,
                         lambda = 0.01)
deepSurv_fit
```

```
## Summary of predicted values, risk score  and martingale residuals:
##             Min. 1st Qu. Median    Mean 3rd Qu.  Max.
## predictors  1.00   1.000  1.000  1.0000   1.000 1.000
## risk        2.72   2.718  2.718  2.7183   2.718 2.718
## residuals  -3.76  -0.379  0.144 -0.0102   0.692 0.921
## Concordance index: 0.5901
##
## for n =  500 observation(s).
##
## Distribution of baseline survival time for the training data:
## Call: survfit(formula = y0 ~ 1, se.fit = se.fit, conf.int = conf.int)
##
##        n events median 0.95LCL 0.95UCL
## [1,] 500    376    158     150     163
## log Lik. -1951.467
```

This model offers many benefits by modelling complex, non-linear relationships in survival data and providing personalized risk predictions. It is adept at handling high-dimensional data and automatic feature learning, which often results in better prediction compared to traditional Cox models. However, its computational complexity and tendencies to overfitting are a challenge, especially with small datasets. Moreover, the black-box nature of deep learning models can hinder interpretability, making it difficult to understand how covariates influence survival outcomes [34].

7.2.2 DeepHit

DeepHit is a deep learning model designed for survival analysis, particularly good at handling competing risks. Unlike traditional models that often assume a specific stochastic process, DeepHit learns the distribution of survival times directly through a DNN, which allows for flexibility in modelling the relationship between the covariates and survival times, without making strong parametric assumptions. DeepHit consists of a shared sub-network and multiple cause specific sub-networks, which enables it to handle datasets with both single and multiple competing risks effectively. The model is trained using a loss function that accounts for both survival times and relative risks [37].

DeepHit shares the same challenges as other deep learning models, such as computational complexity, tendency for overfitting, especially with small datasets, and difficult interpretations due to the model's black-box nature.

As DeepHit's strength lies in its ability to handle competing risks we omit an example due to lack of suitable data. Nevertheless, the DeepHit model can be implemented in R using the `survivalmodels` package [38].

7.2.3 DeepAFT

The deepAFT model is a new approach that integrates deep learning techniques with the AFT model (5.2) to predict survival outcomes, without assuming a specific distribution for the error term. It uses multiple hidden layer feed-forward artificial neural networks (ANNs) to capture the nonlinear effects of input variables. The model effectively handles censoring through imputation, using the imputed data to directly predict survival functions, restricted mean survival time, and median survival time. deepAFT demonstrates predictive performance comparable to other deep learning methods, even when the AFT assumption is violated [39].

We address the risk of overfitting, often associated with more computationally intensive deep learning models for smaller datasets by using a less complex model. We have opted to include only a single hidden layer with 32 nodes in our model architecture. We use the dnn package [35]:

```
set.seed(723)
# New DNN model
dnn_model_2 <- dNNmodel(N = 2,
                        units = c(32, 1),
                        optimizer = "adam",
                        input_shape = 4,
                        activation = c("relu", "sigmoid")
                        )
# Fit the deepAFT model
deepAFT_fit <- deepAFT(Surv(y, failed) ~ . ,
                       model = dnn_model_2,
                       data = sim_data$data,
                       method = "ipcw"
                       )
deepAFT_fit
```

```
## Deep AFT model with ipcw method
##
## Summary of predicted values of mu, location exp(mu) and martingale residuals:
##              Min. 1st Qu. Median     Mean 3rd Qu.   Max.
## predictors  3.48    3.54  3.673  3.74655    3.89  4.455
## locations  32.36   34.52 39.352 43.73508   49.11 86.065
## residuals  -4.10   -0.45  0.164 -0.00292    0.69  0.998
## Concordance index: 0.6195
##
## for n =  500 observation(s).
##
## Distribution of T0 = T/exp(mu) for the training data:
## Call: survfit(formula = y0 ~ 1, se.fit = se.fit, conf.int = conf.int)
##
##        n events median 0.95LCL 0.95UCL
## [1,] 500    376   1.38    1.33    1.46
## log Lik. -353.2975
```

The deepAFT model offers several advantages over traditional survival analysis methods. Its ability to model nonlinear relationships without assuming a specific error distribution makes it highly flexible and adaptable to various types of data. This flexibility allows deepAFT to provide more accurate predictions, especially in complex datasets where traditional models like the Cox regression may fail. Additionally, deepAFT's integration of deep learning techniques enables it to handle high-dimensional data effectively. However, this model often requires substantial computational resource allocation and user expertise in deep learning.

Furthermore, while deepAFT addresses censoring through imputation and IPCW, the accuracy of these techniques depends on the assumptions made about the censoring mechanism [39].

When choosing between DeepSurv, DeepHit, and DeepAFT for survival analysis, the decision should be based on the specific characteristics of the data and the requirements of the model. DeepSurv is suitable for datasets where the Cox proportional hazards assumption holds, as it models the log-risk function and is effective for personalized treatment recommendations. If the data involves competing risks or multiple events, DeepHit is a better choice because it can handle complex relationships between covariates without relying on the proportional hazards assumption. DeepAFT is useful for modeling survival times directly as it adapts the AFT model to deep learning frameworks.

8 Model Evaluation and Comparison

Model evaluation and comparison are critical steps in survival analysis to ensure the reliability and effectiveness of our statistical models. These processes help us assess model fit, compare different models, and select the most appropriate one for our questions. While general model evaluation techniques such as Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are applicable to survival models, there are also specific methods tailored for survival analysis. These specialized techniques account for the unique characteristics of survival data, such as censoring and time-dependent covariates.

8.1 C-Index

The most popular model evaluation metric in survival analysis is the *Harrell's Concordance Index*, also known as the *C-index*. It measures the model's discriminative ability, or the ability of a survival model to correctly rank the predicted survival times relative to the actual survival outcomes. We define the predicted risk score for the ith observation as $\hat{\eta}_i = \mathbf{x}_i^T\hat{\beta}$. Then, if $\hat{\eta}_j > \hat{\eta}_i$, our model predicts that the jth observation has a bigger hazard than the ith observation. Using this predicted risk, we define the *C-index* as the proportion of concordant pairs of observations (i.e. pairs for which the order of events is predicted correctly):

$$C = \frac{\sum_{i,j:y_i>y_j} I(\hat{\eta}_j > \hat{\eta}_i)\delta_j}{\sum_{i,j:y_i>y_j} \delta_j},$$

where y_i is the observed time and δ_i is the status indicator for the ith observation [1]. The *C-index* ranges from 0 to 1, with $C = 0.5$ indicating that the model is as good at predicting survival outcomes as random guessing.

As it is the most popular way to assess the discrimination of the model in survival analysis, we can see the concordance index displayed in most model summaries. With Random Survival Forests, we have shown how it can be calculated from the model output. However, the `survival` package also contains the `concordance` function for easy computation of the C-index of the model [4]:

```r
# C-index of our Cox PH fit
concordance(cph_fit)
```

```
## Call:
## concordance.coxph(object = cph_fit)
##
## n= 500
## Concordance= 0.6103 se= 0.01667
## concordant discordant     tied.x     tied.y    tied.xy
##      55005      35119          0       1763          0
```

Similarly, the package `glmnet` includes the function `Cindex` [14] for the calculation of the concordance index of a regularized Cox PH model.

8.2 Brier Score

The *Brier score* is a widely used metric for evaluating the accuracy of predictions in survival analysis. It measures the mean squared difference between the observed survival status and the predicted probability of survival at a specific time point. Specifically, for a given time t, the *Brier score* is given by

$$\text{BS}(t) = \frac{1}{n}\sum_{i=1}^{n}(\hat{S}(t|\mathbf{x}_i) - \delta_i)^2,$$

where $\hat{S}(t|\mathbf{x}_i)$ is the predicted survival probability for the individual at time t and δ_i is the censoring indicator. The Brier score ranges from 0 to 1, with lower values indicating better predictive accuracy [40].

The *Integrated Brier score* (IBS) is an extension of the Brier score that provides a summary measure of prediction accuracy over a range of time points, rather than at a single time point. It is defined as the integral of the Brier score over a specified time interval, typically from the start of the study up to a maximum follow-up time $t_{\max}$:

$$\text{IBS} = \frac{1}{t_{\max}}\int_0^{t_{\max}} \text{BS}(t)dt.$$

We can calculate the Integrated Brier score using the `pec` package [41]:

```r
library(pec)
# Calculate the Brier Score
pec(object = cph_fit,
    data = sim_data$data,
    times = seq(0, 100, by = 1))
```

```
##
## Prediction error curves
##
## Prediction models:
##
## Reference     coxph
## Reference     coxph
##
## Right-censored response of a survival model
##
## No.Observations: 500
##
## Pattern:
##                 Freq
##  event          376
##  right.censored 124
##
## IPCW: cox model
##
## No data splitting: either apparent or independent test sample performance
##
## Cumulative prediction error, aka Integrated Brier score  (IBS)
##  aka Cumulative rank probability score
##
## Range of integration: 0 and time=100 :
##
```

```
##
## Integrated Brier score (crps):
##
##            IBS[0;time=100)
## Reference            0.154
## coxph                0.143
```

The Integrated Brier score is an essential tool in survival analysis, providing a comprehensive measure of model performance by evaluating both the accuracy of predictions (*calibration*) and the model's ability to capture the underlying risk structure (*discrimination*), whereas the C-index only assesses the discrimination.

9 Conclusion

We have provided a comprehensive overview of survival analysis techniques and their implementation in R, covering fundamental concepts, descriptive methods, and regression models. By exploring both traditional approaches and modern developments, we've demonstrated the versatility and power of survival analysis across various fields.

The integration of machine learning techniques with survival analysis has opened up new avenues for research and application. Machine learning methods, such as random survival forests and neural networks, are increasingly being used to handle complex, high-dimensional survival data. These approaches can capture non-linear relationships and interactions between variables, potentially leading to more accurate predictions of survival outcomes. Looking ahead, the field of survival analysis is awaiting further advancements. Future research may focus on developing more sophisticated methods for handling time-varying covariates and competing risks.

As big data continues to proliferate across industries, the intersection of survival analysis and machine learning is likely to become increasingly important. Researchers should stay up to date with emerging techniques and tools that can improve their ability to analyze complex survival data.

10 References

[1] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning with applications in r*, 2nd ed. in Springer texts in statistics. Springer, 2021. doi: 10.1007/978-1-0716-1418-1.

[2] J. Kropko and J. J. Harden, *Coxed: Duration-based quantities of interest for the cox proportional hazards model.* 2020. Available: https://CRAN.R-project.org/package=coxed

[3] J. J. Harden and J. Kropko, "Simulating duration data for the cox model," *Political Science Research and Methods*, vol. 7, no. 4, pp. 921–928, 2019, doi: 10.1017/psrm.2018.19.

[4] T. M. Therneau, *A package for survival analysis in r.* 2024. Available: https://CRAN.R-project.org/package=survival

[5] A. Kassambara, M. Kosinski, and P. Biecek, *Survminer: Drawing survival curves using 'ggplot2'.* 2021. Available: https://CRAN.R-project.org/package=survminer

[6] E. Bollschweiler, "Benefits and limitations of kaplan–meier calculations of survival chance in cancer surgery," *Langenbeck's Archives of Surgery*, vol. 388, no. 4, pp. 239–244, 2003, doi: 10.1007/s00423-003-0410-6.

[7] Ø. Borgan, "Nelson–aalen estimator," in *Wiley StatsRef: Statistics reference online*, John Wiley & Sons, Ltd, 2014. doi: https://doi.org/10.1002/9781118445112.stat06045.

[8] W. Nelson, "Theory and applications of hazard plotting for censored failure data," *Technometrics*, vol. 14, no. 4, pp. 945–966, 1972, doi: 10.1080/00401706.1972.10488991.

[9] J. P. Klein and M. L. Moeschberger, *Survival analysis: Techniques for censored and truncated data*, 2nd ed. in Statistics for biology and health. New York, NY: Springer, 2003. doi: 10.1007/b97377.

[10] S. original by Kenneth Hess and R. port by R. Gentleman, *Muhaz: Hazard function estimation in survival analysis.* 2021. Available: https://CRAN.R-project.org/package=muhaz

[11] D. R. Cox, "Regression models and life-tables," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–202, Jan. 1972, doi: 10.1111/j.2517-6161.1972.tb00899.x.

[12] Y. Huang, J. Li, M. Li, and R. R. Aparasu, "Application of machine learning in predicting survival outcomes involving real-world data: A scoping review," *BMC Medical Research Methodology*, vol. 23, no. 1, p. 268, 2023, doi: 10.1186/s12874-023-02078-1.

[13] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for cox's proportional hazards model via coordinate descent," *Journal of Statistical Software*, vol. 39, no. 5, pp. 1–13, 2011, doi: 10.18637/jss.v039.i05.

[14] J. Friedman, T. Hastie, R. Tibshirani, N. Simon, B. Narasimhan, and J. Qian, *Glmnet: Lasso and elastic-net regularized generalized linear models.* 2023. Available: https://cran.r-project.org/web/packages/glmnet/index.html

[15] Y. Wu, "Elastic net for cox's proportional hazards model with a solution path algorithm," *Statistica Sinica*, vol. 22, pp. 27–294, 2012, doi: 10.5705/ss.2010.107.

[16] C. Jackson, "flexsurv: A platform for parametric survival modeling in R," *Journal of Statistical Software*, vol. 70, no. 8, pp. 1–33, 2016, doi: 10.18637/jss.v070.i08.

[17] J. D. Kalbfleisch and R. L. Prentice, *The statistical analysis of failure time data.* in Wiley series in probability and statistics. John Wiley & Sons, Inc., 2002. doi: 10.1002/9781118032985.

[18] C. H. Jackson, *Flexsurv: Flexible parametric survival and multi-state models.* 2023. Available: https://cran.r-project.org/web/packages/flexsurv/index.html

[19] T. Martinussen and T. H. Scheike, *Dynamic regression models for survival data.* in Statistics for biology and health. New York: Springer Verlag, 2006.

[20] T. H. Scheike, Z. Zhang, and M. Zhou, *Timereg: Flexible regression models for survival data.* 2022. Available: https://cran.r-project.org/web/packages/timereg/index.html

[21] Z. Zhang, J. Reinikainen, K. A. Adeleke, M. E. Pieterse, and C. G. Groothuis-Oudshoorn, "Time-varying covariates and coefficients in cox regression models," *Annals of Translational Medicine*, vol. 6, no. 7, p. 121, 2018, doi: 10.21037/atm.2018.02.12.

[22] P. C. Austin, D. S. Lee, and J. P. Fine, "Introduction to the analysis of survival data in the presence of competing risks," *Circulation*, vol. 133, no. 6, pp. 601–609, Feb. 2016, doi: 10.1161/CIRCULATIONAHA.115.017719.

[23] K. Monterrubio-Gómez, N. Constantine-Cooke, and C. A. Vallejos, “A review on competing risks methods for survival analysis.” 2022. Available: https://arxiv.org/abs/2212.05157

[24] B. Gray, *Cmprsk: Subdistribution analysis of competing risks.* 2024. Available: https://CRAN.R-project.org/package=cmprsk

[25] J. P. Fine and R. J. Gray, “A proportional hazards model for the subdistribution of a competing risk,” *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 496–509, 1999, doi: 10.1080/01621459.1999.10474144.

[26] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, “Random survival forests,” *The Annals of Applied Statistics*, vol. 2, no. 3, pp. 841–860, 2008, doi: 10.1214/08-AOAS169.

[27] H. S. Ishwaran and U. B. Kogalur, *randomForestSRC: Fast unified random forests for survival, regression, and classification (RF-SRC).* 2022. Available: https://cran.r-project.org/web/packages/randomForestSRC/index.html

[28] B. C. Jaeger *et al.*, “Oblique random survival forests,” *The Annals of Applied Statistics*, vol. 13, no. 3, pp. 1847–1883, 2019, doi: 10.1214/19-AOAS1261.

[29] B. C. Jaeger *et al.*, “Accelerated and interpretable oblique random survival forests.” 2022. Available: https://arxiv.org/abs/2208.01129

[30] B. C. Jaeger, S. Welden, K. Lenoir, and N. M. Pajewski, “Aorsf: An r package for supervised learning using the oblique random survival forest,” *Journal of Open Source Software*, vol. 7, no. 77, p. 4705, 2022, Available: https://doi.org/10.21105/joss.04705

[31] K. H. Torsten Hothorn and A. Zeileis, “Unbiased recursive partitioning: A conditional inference framework,” *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 651–674, 2006, doi: 10.1198/106186006X133933.

[32] W. Yao, H. Frydman, and J. S. Simonoff, *ICcforest: An ensemble method for interval-censored survival data.* 2020. Available: https://CRAN.R-project.org/package=ICcforest

[33] S. Lee *et al.*, “Predictive scores for identifying patients with type 2 diabetes mellitus at risk of acute myocardial infarction and sudden cardiac death,” *Endocrinology, Diabetes & Metabolism*, vol. 4, no. 3, p. e00240, 2021, doi: https://doi.org/10.1002/edm2.240.

[34] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger, “DeepSurv: Personalized treatment recommender system using a cox proportional hazards deep neural network,” *BMC Medical Research Methodology*, vol. 18, no. 1, p. 24, 2018, doi: 10.1186/s12874-018-0482-1.

[35] Bingshu E Chen, *A package of deep nerual network tools for probability models.* 2022. Available: https://CRAN.R-project.org/package=dnn

[36] A. Ghatak, *Deep learning applications: Theory and practices.* Singapore: Springer, 2020. doi: 10.1007/978-981-13-5850-0.

[37] C. Lee, W. Zame, J. Yoon, and M. van der Schaar, “DeepHit: A deep learning approach to survival analysis with competing risks,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018, doi: 10.1609/aaai.v32i1.11842.

[38] R. Sonabend, *Survivalmodels: Models for survival analysis.* 2024. Available: https://CRAN.R-project.org/package=survivalmodels

[39] P. A. Norman, W. Li, W. Jiang, and B. E. Chen, “deepAFT: A nonlinear accelerated failure time model with artificial neural network,” *Statistics in Medicine*, vol. 43, no. 19, pp. 3689–3701, 2024, doi: https://doi.org/10.1002/sim.10152.

[40] E. Graf, C. Schmoor, W. Sauerbrei, and M. Schumacher, “Assessment and comparison of prognostic classification schemes for survival data,” *Statistics in Medicine*, vol. 18, no. 17–18, pp. 2529–2545, 1999, doi: https://doi.org/10.1002/(SICI)1097-0258(19990915/30)18:17/18%3C2529::AID-SIM274%3E3.0.CO;2-5.

[41] T. A. Gerds and F. E. H. Jr., *Pec: Prediction error curves.* 2023. Available: https://cran.r-project.org/web/packages/pec/index.html

A Appendix

Name	Author(s)	Last Updated	Description	Main Functions
coxed	Jonathan Kropko, Jeffrey J. Harden	2022-10-12	Converts hazard functions from the Cox model into interpretable time-based metrics and simulates survival data with flexible baseline hazards.	coxed, sim.survdata
survival	Terry M. Therneau, Thomas Lumley	2024-06-05	Allows to define survival data as well as provides essential survival analysis tools, including Cox models, Kaplan-Meier estimators, and parametric survival models.	Surv, coxph, survfit, survdiff
survminer	Alboukadel Kassambara, Marcin Kosinski,Przemyslaw Biecek	2021-03-09	Enhances survival analysis visualization in R, providing tools for survival curves, forest plots, and customized plots.	ggsurvplot, ggforest
muhaz	Kenneth Hess, David Winsemius	2022-10-13	Estimates and plots the hazard function using kernel methods, piecewise exponential methods, and Kaplan-Meier estimates.	muhaz, pehaz, kphaz.fit
glmnet	Jerome Friedman, Trevor Hastie, Rob Tibshirani et al.	2023-08-22	Fits Cox models with elastic-net regularization, providing functions for plotting coefficients and survival curves.	glmnet, cv.glmnet
flexsurv	Christopher Jackson	2024-04-19	Flexible parametric survival models in R, supporting various distributions and custom hazard functions for complex survival analysis.	flexsurvreg, flexsurvspline
timereg	Thomas Scheike	2023-01-17	Provides tools for regression models on survival and event history data, including Cox models and additive hazard models.	aalen, cox.aalen, dynreg, prop.excess, prop.odds, timecox, two.stage
cmprsk	Bob Gray	2024-05-19	Tools for competing risks analysis and cumulative incidence function estimation using the proportional subdistribution hazards model.	crr,cuminc
randomForestSRC	Hemant Ishwaran, Udaya B. Kogalur	2024-06-25	Implements random forests for survival, regression, and classification with tools for model fitting, prediction, and evaluation.	rfsrc, rfsrc.fast
aorsf	Tyler B. M. Linder	2024-07-15	Provides tools for building oblique random survival forests, optimizing split rules for survival data, and enhancing model performance using ensemble techniques.	orsf
ICcforest	Weichi Yao, Halina Frydman, Jeffrey S. Simonoff	2022-10-12	Extends conditional inference trees and forests to handle interval-censored survival data.	ICtree, ICcforest
pec	Thomas A. Gerds	2023-06-19	Tools for evaluating prediction models in survival analysis using prediction error curves, cross-validation, and other performance metrics.	pec

Table 1: Summary of R packages used for survival analysis