Machine Learning in Survival Analysis

Getting Started

Main Title

Standard blurb goes here

%%Placeholder for Half title

Series page goes here (if applicable); otherwise blank

Machine Learning in Survival Analysis

Raphael Sonabend, Andreas Bender

Imprint page here; PE will provide text

Table of contents

Preface

> "Everything happens to everybody sooner or later if there is time enough" - George Bernard Shaw
>
> "... but in this world nothing can be said to be certain, except death and taxes." - Benjamin Franklin

A logical consequence of Bernard Shaw's quote is that if there is time enough, then everybody will have experienced a given event at some point. This is one of the central assumptions to survival analysis (specifically to single-event analysis, but we'll get to that later). As nothing can be certain (except death and taxes), machine learning can be used to predict the probability people will experience the event and *when*. This is exactly the problem that this book tackles.

With immortality only being a theoretical concept, there is never 'time enough', hence survival analysis assumes that the event of interest is guaranteed to occur within an object's lifetime. This event could be a patient entering remission after a cancer diagnosis, the lifetime of a lightbulb after manufacturing, the time taken to finish a race, or any other event that is observed over time. Survival analysis differs from other fields of Statistics in that uncertainty is explicit encoded in the survival problem; this uncertainty is known as 'censoring'. For example, say a model is being built to predict when a marathon runner will finish a race and to learn this information the model is fed data from every marathon over the past five years. Across this period, there will be many runners who never finish their race. Instead, these runners are said to be 'censored' and the model uses all information up until the point of censoring (dropping out the race), and learns that they ran for at least as long as their censoring time (the time they dropped out). Censoring is unique to survival analysis and without the presence of censoring, survival analysis is mathematically equivalent to regression.

This book covers survival analysis in the most common right-censoring setting for independent censoring, as well as discussing competing risk frameworks for dependent censoring - these terms will all be covered in the introduction of the book.

A note from Raphael: I wrote my PhD thesis about machine learning applications to survival analysis as I was interested in understanding why more researchers were not using machine learning models for survival analysis. Since then I've had the pleasure to work with, and advise, researchers across different sectors, including pharmaceutical companies, governmental agencies, funding organisations, and research institutions. I hope that this book continues to help researchers discover machine learning survival analysis and to navigate the nuances and complexities it presents.

A note from Andreas: FIXME.

Overview

This textbook is intended to fill a gap in the literature by providing a comprehensive introduction to machine learning in the survival setting. If you are interested in machine learning or survival analysis separately then you might consider James et al. (2013), Hastie, Tibshirani, and Friedman (2001), Bishop (2006) for machine learning and Collett (2014), J. D. Kalbfleisch and Prentice (1973) for survival analysis. This book serves as a complement to the above examples and introduces common machine learning terminology from simpler settings such as regression and classification, but without diving into the detail found in other sources, instead focusing on extension to the survival analysis setting.

This book may be useful for Masters or PhD students who are specialising in machine learning in survival analysis, machine learning practitioners looking to work in the survival setting, or statisticians who are familiar with survival analysis but less so with machine learning. The book could be read cover-to-cover, but this is not advised. Instead it may be preferable to dip into sections of the book as required and use the 'signposts' that direct the reader to sections of the book that are relevant to each other.

The book is split into five parts:

Part I: Survival Analysis and Machine Learning The book begins by introducing the basics of survival analysis and machine learning and unifying terminology between the two to enable meaningful description of 'machine learning in survival analysis' (MLSA). In particular, the survival analysis 'task' and survival 'prediction types' are defined.

Part II: Evaluation The second part of the book discusses one of the most important parts of the machine learning workflow, model evaluation. In the simplest case, without evaluation there is no way to know if predictions from a trained machine learning model are any good. Whether one uses a Kaplan-Meier estimator, a complex neural network, or anything in between, there is no guarantee any of these methods will actually make useful predictions for a given dataset. This could be because the dataset is inherently difficult for any model to be trained on, perhaps because it is very 'noisy', or because a model is simply ill-suited to the task, for example using a Cox Proportional Hazards model when its key assumptions are violated. Evaluation is therefore crucial to trusting any predictions made from a model.

The measures in Part II are presented in different classes that reflect the prediction types identified in Part I. In-sample measures, which evaluate the quality of a model's 'fit' to data, are not included as this book primarily focuses on external validation of predictive machine learning models. Readers who are interested in this are are directed to Collett (2014) and Hosmer Jr, Lemeshow, and May (2011) for discussion on residuals; Choodari-Oskooei, Royston, and Parmar (2012a) and Patrick Royston and Sauerbrei (2004) for R^2 type measures; and Volinsky and Raftery (2000), Hurvich and Tsai (1989), and Liang and Zou (2008) for information criterion measures.

In each chapter, the measure class is introduced, particular metrics are listed, and commentary is provided on how and when to use the measures. Recommendations for choosing measures are discussed in Chapter 21.

Part III: Models Part III is a deep dive into machine learning models for solving survival analysis problems. This begins with 'classical' models that may not be considered 'machine learning' and then continues by exploring different classes of machine learning models including random forests, support vector machines, gradient boosting machines, neural networks, and other less common classes. Each model class is introduced in the simpler regression setting and then extensions to survival analysis are discussed. Differences between

model implementations are not discussed, instead the focus is on understanding how these models are built for survival analysis - in this way readers are well-equipped to independently follow individual papers introducing specific implementations.

Part IV: Reduction Techniques The next part of the book introduces reduction techniques in survival analysis, which is the process of solving the survival analysis task by using methods from other fields. In particular, chapters focus on demonstrating how any survival model can be used in the competing risks setting, discrete time modelling, Poisson methods, pseudovalues (reduction to regression), and other advanced modelling methods.

Part V: Extensions and Outlook The final part of the book provides some miscellaneous chapters that may be of use to readers. The first chapter lists common practical problems that occur when running survival analysis experiments and solutions that we have found useful. The next lists open-source software at the time of writing for running machine learning survival analysis experiments. The final chapter is our outlook on survival analysis and where the field may be heading.

Exercises are provided at the end of the book so you can test yourself as you go along.

Citing this book

Whilst this book remains a work in progress you can cite it as

```
Sonabend. R, Bender. A. (2025). Machine Learning in Survival Analysis.
https://www.mlsabook.com.
```

```
@book{MLSA2025
    title = {Machine Learning in Survival Analysis},
    editor = {Raphael Sonabend, Andreas Bender},
    url = {https://www.mlsabook.com},
    year = {2025}
}
```

Please see the front page of the book website (https://www.mlsabook.com) for full licensing details.

We hope you enjoy reading this book.

Raphael and Andreas

Authors

Raphael Sonabend is the CEO and Co-Founder of OSPO Now, a company providing virtual open-source program offices as a service. They are also a Visiting Researcher at Imperial College London. Raphael holds a PhD in statistics, specializing in machine learning applications for survival analysis. They created the R packages `mlr3proba`, `survivalmodels`, and the Julia package `SurvivalAnalysis.jl`. Raphael co-edited and co-authored *Applied Machine Learning Using mlr3 in R* (Bischl et al. 2024).

Andreas Bender is...

Acknowledgments

We would like to gratefully acknowledge our colleagues that reviewed the content of this book, including: Lukas Burk, Dr Cesaire Fouodo, Prof. Dr. Matthias Schmid.

Symbols and Notation

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

The most common symbols and notation used throughout this book are presented below; in rare cases where different meanings are intended within the book, this will be made clear.

Fonts, matrices, vectors

A lower-case letter in normal font, x, refers to a single, fixed observation. When in bold font, a lower-case letter, $\mathbf{x}$, refers to a vector of fixed observations, and an upper-case letter, $\mathbf{X}$, represents a matrix. Calligraphic letters, $\mathcal{X}$, are used to denote sets.

A matrix will always be defined with its dimensions using the notation, $\mathbf{X} \in \mathcal{X}^{n \times p}$, or if for example $\mathcal{X}$ is the set of Reals, it may be written as "$\mathbf{X}$ is a $n \times p$ Real-valued matrix", analogously for integer-valued matrices etc. By default, a 'vector' will refer to a column vector, which may be thought of as a matrix with n rows and one column, and may be represented as:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Vectors are usually defined using transpose notation, for example the vector above may instead be written as $\mathbf{x}^\top = (x_1 \; x_2 \cdots x_n)$ or $\mathbf{x} = (x_1 \; x_2 \cdots x_n)^\top$. Vectors may also be defined in a shortened format as $\mathbf{x} \in \mathcal{X}^{n \times 1}$ or more simply $\mathbf{x} \in \mathcal{X}^n$, which implies a column vector of length n with elements as represented above.

A letter in normal font with one subscript refers to a single element from a vector. For example, given $\mathbf{x} \in \mathcal{X}^n$, the ith element is denoted x_i. Given a matrix $\mathbf{X} \in \mathcal{X}^{n \times p}$, a bold-face lower-case letter with a single subscript refers to the row of a matrix, for example the ith row would be $\mathbf{x}_i = (x_{i;1} \; x_{i;2} \cdots x_{i;p})^\top$. Whereas a column is referenced with a semi-colon before the subscript, for example the jth column would be $\mathbf{x}_{;j} = (x_{1;j} \; x_{2;j} \cdots x_{n;j})^\top$. Two subscripts can be used to reference a single element of a matrix, for example $x_{i;j} \in \mathcal{X}$ would be the element in the ith row and jth column of $\mathbf{X}$.

Functions

Typically, a 'hat', $\hat{x}$, will refer to the prediction or estimation of a variable, x, with bold-face used again to represent vectors. A 'bar', $\bar{x}$, refers to the sample mean of $\mathbf{x}$. Capital letters in normal font, X, refer to scalar or vector random variables, which will be made clear from context. $\mathbb{E}(X)$ is the expectation of the random variable X. We write $A \perp\!\!\!\perp B$, to denote that A and B are independent, i.e., that $P(A \cap B) = P(A)P(B)$.

A function f, will either be written as a formal map of domain to codomain, $f : \mathcal{X} \to \mathcal{Y}; (x, y) \mapsto f(x, y)$ (which is most useful for understanding inputs and outputs), or more simply and commonly as $f(x, y)$. Given a random variable, X, following distribution ζ (mathematically written $X \sim \zeta$), then f_X denotes the probability density function, and analogously for other distribution defining functions such as the cumulative distribution function, survival function, etc. In the survival analysis context (Chapter 4), a subscript "0" refers to a "baseline" function, for example, S_0 is the baseline survival function.

Finally, exp, refers to the exponential function, $f(x) = e^x$, and log refers to the natural logarithm $\ln(x) = \log_e(x)$.

Variables and acronyms

Common variables and acronyms used in the book are given in Table 0.1 and Table 0.2 respectively.

Table 0.1: Common variables used throughout the book.

Variable	Definition
$\mathbb{R}, \mathbb{R}_{>0}, \mathbb{R}_{\geq 0}, \bar{\mathbb{R}}$	Set of Reals, positive Reals (excl. zero), non-negative Reals (incl. zero), and Reals including $\pm\infty$.
$\mathbb{N}_{>0}$	Set of Naturals excluding zero.
$(\mathbf{X}, \mathbf{t}, \boldsymbol{\delta})$	Survival data where $\mathbf{X} \in \mathbb{R}^{n \times p}$ is a real-valued matrix of n observations (rows) and p features (columns), $\mathbf{t} \in \mathbb{R}^n$ is a vector of observed outcome times, and $\boldsymbol{\delta} \in \mathbb{R}^n$ is a vector of observed outcome indicators.
$\boldsymbol{\beta}$	Vector of model coefficients/weights, $\boldsymbol{\beta} \in \mathbb{R}^p$.
$\boldsymbol{\eta}$	Vector of linear predictors, $\boldsymbol{\eta} = (\eta_1 \ \eta_2 \cdots \eta_n)^\top$, where $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$ and $\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$.
$\mathcal{D}, \mathcal{D}_{train}, \mathcal{D}_{test}$	Dataset, training data, and testing data.

Table 0.2: Common acronyms used throughout the book.

Acronym	Definition
AFT	Accelerated Failure Time
cdf	Cumulative Distribution Function
chf	Cumulative Hazard Function
CPH	Cox Proportional Hazards

Acronym	Definition
GBM	Gradient Boosting Machine
GLM	Generalised Linear Model
IPC(W)	Inverse Probability of Censoring (Weighted)
ML	Machine Learning
pdf	Probability Density Function
PH	Proportional Hazards
(S)SVM	(Survival) Support Vector Machine
t.v.i.	Taking Values In

1

Introduction

> ! Major changes expected!
>
> **This page is a work in progress and major changes will be made over time.**

TODO

- Mention somewhere that SA can be used to solve T-year prediciton problems (i.e., see if we can get classif users over to SA).
- Also SA can be used for censoring/truncation

Writing after a global pandemic, applications of survival analysis are more relevant than ever. Predicting the time from onset of COVID-19 symptoms to hospitalisation, or the time from hospitalisation to intubation, or intubation to death, are all time-to-event predictions that are at the centre of survival analysis. As well as morbid applications, survival analysis predictions may be concerned with predicting the time until a customer cancels their gym membership, or the lifetime of a lightbulb; any event that is guaranteed (or at least very likely) to occur can be modelled by a survival analysis prediction. As these predictions can be so sensitive, for example a model predicting when a child should be taken off breathing support (Data Study Group Team 2020), the best possible predictions, evaluated to the highest standard, are a necessity. In other fields of predictive modelling, machine learning has made incredible breakthroughs (such as AlphaFold), therefore applying machine learning to survival analysis is a natural step in the evolution of an important field.

Survival analysis is the field of Statistics focusing on modelling the distribution of an event, which may mean the time until the event takes place, the risk of the event happening, the probability of the event occurring at a single time, or the event's underlying probability distribution. Survival analysis ('survival') is a unique field of study in Statistics as it includes the added difficulty of 'censoring'. Censoring is best described through example: a study is conducted to determine the mortality rate of a group of patients after diagnoses with a particular disease. If a patient dies during this study then their outcome is 'death' and their time of death can be recorded. However if a patient drops-out of the study before they die, then their time of death (though guaranteed to occur) is unknown and the only available information is the time at which they left the study. This patient is now said to be *censored* at the time they drop out. The censoring mechanism allows as much outcome information (time and event) to be captured as possible for all patients (observations).

Machine learning (ML) is the field of Statistics primarily concerned with building models to either predict outputs from inputs or to learn relationships from data (Hastie, Tibshirani, and Friedman 2001; James et al. 2013). This book is limited to the former case, or more specifically

supervised learning, as this is the field in which the vast majority of survival problems live. Relative to other areas of supervised learning, development in survival analysis has been slow – the majority of developments in machine learning for survival analysis have only been in the past decade (see chapters (**?@sec-review**)-(Part II)). This appears to have resulted in less interest in the development of machine learning survival models (**?@sec-review**), less rigour in the evaluation of such models (Part II), and fewer off-shelf/open-source implementations (R. Sonabend, Király, et al. 2021). This book seeks to set the foundations for clear workflows, good practice, and precise results for 'machine learning survival analysis'.

1.1 Why is this book needed?

Firstly, whilst there are many books dedicated to regression and classification as machine learning problems (the 'bibles' of machine learning focus entirely on regression and classification only (Bishop 2006; Hastie, Tibshirani, and Friedman 2001; James et al. 2013)), there is a deficit of books covering the survival analysis setting. By writing this book we hope to fill this gap and enable more practitioners to use cutting-edge methods in survival analysis. Survival analysis has important applications in healthcare, finance, engineering and more, all fields that directly impact upon individual lives on a day-to-day basis, and should perhaps be considered as important as classification and regression. The result of this gap in interest, is the erroneous assumption that one field can be directly applied to another. For example there is evidence of researchers treating censoring as a nuisance to be ignored and using regression models instead (Schwarzer, Vach, and Schumacher 2000). Censoring is indeed a challenge and may contribute to making survival analysis less accessible than other fields, but this need not be the case; a clear unification of terminology and presentation of methods may help make 'machine learning survival analysis' more accessible. Added accessibility could lead to more academics (and non-academics) engaging with the field and promoting good standards of practice, as well as developing more novel models and measures.

Where survival models have been developed, these have skewed towards 'ranking models', which predict the relative risk of an event occurring (Chapter 6). In many applications these predictions are sufficient, for example in randomised control trials if assessing the increased/decreased risk of an event after treatment. However, there are many use-cases where predicting an individual's survival probability distribution is required. Take, for example, an engineer calculating the lifetime of a plane's engine.[1] There are three important reasons to replace a jet engine at the optimal time:

- financial: jet engines are very expensive and replacing one sooner than required is a waste of money;
- environmental: an engine being replaced too early is a waste of potential usage;
- safety: if the engine is replaced too late then there is a risk to passengers.

Now consider examples for the three possible 'prediction types' the engineer can make:

i. A 'relative risk prediction': This engine is twice as likely to fail as another.
ii. A 'survival time prediction': The engine is expected to fail in 30 days.
iii. A 'survival distribution prediction': The lifetime of the engine is distributed according to the probability distribution ζ.

[1] In this engineering context, survival analysis is usually referred to as reliability analysis.

The first prediction type is not useful as the underlying relative risk may be unknown and the engineer is concerned with the individual lifetime. The second prediction type provides a useful quantity for the engineer to work with however there is no uncertainty captured in this prediction. The third prediction type can capture the uncertainty of failure over the entirety of the positive Reals (though usually only a small subset is possible and useful). With this final prediction type, the engineer can create safe decisions: 'replace the engine at time τ, where τ is the time when the predicted probability of survival drops below 60%, $S(\tau) = 0.6$'. There are ethical, economic, and environmental reasons for a good survival distribution prediction and this book considers a distribution prediction to be the most important prediction type.

Evaluating predictions from survival models is of the utmost importance. This is especially important as survival models are often deployed in the public domain, particularly in healthcare. Physical products in healthcare, such as new vaccines, undergo rigorous testing and research in randomised control trials before being publically deployed; the same level of rigour should be expected for the evaluation of survival models that are used in life-and-death situations. Evaluation measures for regression and classification are well-understood with important properties, however survival measures have not undergone the same treatment. For example many survival models are still being evaluated solely with concordance indices that have been repeatedly criticised (Gönen and Heller 2005; Rahman et al. 2017; Schmid and Potapov 2012).

1.2 Reproducibility

This book includes simulations and figures generated in R, the code for any figures or experiments in this book are freely available at https://github.com/mlsa-book/MLSA under an MIT licence and all content on this website is available under CC BY 4.0.

Further reading

- (P. Wang, Li, and Reddy 2019) provides a light-touch but comprehensive survey of machine learning models for survival analysis.

Part I

Survival Analysis and Machine Learning

2

MLSA From Start to Finish

TODO (150-200 WORDS)

> **!** Page coming soon!
>
> **We are working on this page and it will be available soon!**

3

Machine Learning

TODO (150-200 WORDS)

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

This chapter covers core concepts in machine learning. This is not intended as a comprehensive introduction and does not cover mathematical theory nor how to run machine learning models using software. Instead, the focus is on introducing important concepts and to provide basic intuition for a general machine learning workflow. This includes the concept of a machine learning task, data splitting (resampling), model training and prediction, evaluation, and model comparison. Recommendations for more comprehensive introductions are given at the end of this chapter, including books that cover practical implementation in different programming languages.

3.1 Basic workflow

This book focuses on *supervised learning*, in which predictions are made for outcomes based on data with observed dependent and independent variables. For example, predicting someone's height is a supervised learning problem as data can be collected for features (independent variables) such as age and sex, and an observable outcome (dependent variable), which is height. Alternatives to supervised learning include *unsupervised learning*, *semi-supervised learning*, and *reinforcement learning*. This book is primarily concerned with *predictive* survival analysis, i.e., making future predictions based on (partially) observed survival outcomes, which falls naturally within the supervised learning domain.

The basic machine learning workflow is represented in Figure 3.1. Data is split into training and test datasets. A learner is selected and is trained on the training data, inducing a fitted model. The features from the test data are passed to the model which makes predictions for the unseen outcomes (Box 1). The outcomes from the test data are passed to a chosen measure with the predictions, which evaluates the performance of the model (Box 2). The process of repeating this procedure to test different training and test data is called *resampling* and running multiple resampling experiments with different models is called *benchmarking*. All these concepts will be explained in this chapter.

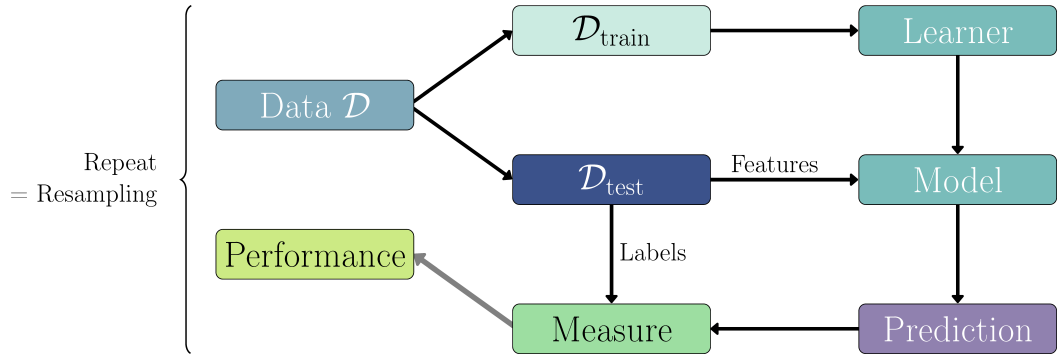


Figure 3.1: Basic machine learning workflow with data splitting, model training, predicting, and evaluating. Image from Foss and Kotthoff (2024) (CC BY-NC-SA 4.0).

3.2 Tasks

A machine learning task is the specification of the mathematical problem that is to be solved by a given algorithm. For example, "predict the height of a male, 13 year old child", is a machine learning task. Tasks are derived from datasets and one dataset can give rise to many tasks across any machine learning domain. The dataset described by columns: 'age', 'weight', 'height', 'sex', 'diagnosis', 'time of death', 'clinician notes', could give rise to any of the following tasks (and more):

- Predict age from weight, height, and sex - supervised regression task
- Predict sex from age and diagnosis - supervised classification task
- Predict time of death from all other features - supervised survival task
- Categorise observations into clusters - unsupervised clustering
- Learn to speak like a clinician depending on client diagnosis - natural language processing, likely with reinforcement learning

As this book is focused on supervised learning, only the first three of these is covered in this chapter and beyond. The specification of a task is vital for interpreting predictions from a model and its subsequent performance. This is particularly true when separating between determinisitc and probabilistic predictions, as discussed later in the chapter.

Formally, let $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^{n \times p}$ be a matrix with p features for n observations and let $y \in \mathcal{Y}$ be a vector of labels (or *outcomes* or *targets*) for all observations. A dataset is then given by $\mathcal{D} = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n))$ where it is assumed $\mathcal{D} \overset{i.i.d.}{\sim} (\mathbb{P}_{xy})^n$ for some unknown distribution $\mathbb{P}$.

A machine learning task is the problem of learning the unknown function $f : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{Y}$ specifies the nature of the task, for example classification, regression, or survival.

3.2.1 Regression

Regression tasks make continuous predictions, for example someone's height. Regression may be deterministic, in which case a single continuous value is predicted, or probabilistic, where a probability distribution over the Reals is predicted. For example, predicting an individual's height as 165cm would be a deterministic regression prediction, whereas predicting their

height follows a $\mathcal{N}(165, 2)$ distribution would be probabilistic.

Formally, a deterministic regression task is specified by $f_{Rd} : \mathcal{X} \to \mathcal{Y} \subseteq \mathbb{R}^n$, and a probabilistic regression task by $f_{Rp} : \mathcal{X} \to \mathcal{S}$ where $\mathcal{S} \subset \text{Distr}(\mathcal{Y})$ and $\text{Distr}(\mathcal{Y})$ is the space of distributions over $\mathcal{Y}$.

In machine learning, deterministic regression is much more common than probabilistic and hence the shorthand 'regression' is used to refer to deterministic regression (in contrast to statistical modeling, where regression usually implies probabilistic regression).

3.2.2 Classification

Classification tasks make discrete predictions, for example whether it will rain, snow, or be sunny tomorrow. Similarly to regression, predictions may be deterministic or probabilistic. Deterministic classification predicts which category an observation falls into, whereas probabilistic classification predicts the probability of an observation falling into each category. Predicting it will rain tomorrow is a deterministic prediction whereas predicting $\hat{p}(rain) = 0.6; \hat{p}(snow) = 0.1; \hat{p}(sunny) = 0.3$ is probabilistic.

Formally, a deterministic classification task is given by $f_{Cd} : \mathcal{X} \to \mathcal{Y} \subseteq \mathbb{N}_0$, and a probabilistic classification task as $f_{Cp} : \mathcal{X} \to \mathcal{Y} \subseteq [0, 1]^k$ where k is the number of categories an observation may fall into. Practically this latter prediction is estimation of the probability mass function $\hat{p}_Y(y) = P(Y = y)$. If only two categories are possible, these reduce to the *binary classification* tasks: $f_{Bd} : \mathcal{X} \to \{0, 1\}$ and $f_{Bp} : \mathcal{X} \to [0, 1]$ for deterministic and probabilistic binary classification respectively.

Note that in the probabilistic binary case it is common to write the task as predicting $[0, 1]$ not $[0, 1]^2$ as the classes are mutually exclusive. The class for which probabilities are predicted is referred to as the *positive class*, and the other as the *negative class*.

3.3 Training and predicting

The terms *algorithm*, *learner*, and *model* are often conflated in machine learning. A *learner* is a description of a learning algorithm, prediction algorithm, parameters, and hyperparameters. The *learning algorithm* is a mathematical strategy to estimate the unknown mapping from features to outcome as represented by a task, $f : \mathcal{X} \to \mathcal{Y}$. During *training*, data, $\mathcal{D}$, is fed into the learning algorithm and induces the *model* $\hat{f}$. Whereas the learner defines the framework for training and prediction, the model is the specific instantiation of this framework after training on data.

After training the model, new data, $\mathbf{x}^*$, can be fed to the *prediction algorithm*, which is a mathematical strategy that uses the model to make predictions $\hat{\mathbf{y}} = \hat{f}(\mathbf{x}^*)$. Algorithms can vary from simple linear equations with coefficients to estimate, to complex iterative procedures that differ considerably between training and predicting.

Algorithms usually involve parameters and hyperparameters. Parameters are learned from data whereas hyperparameters are set beforehand to guide the algorithms. Model *parameters* (or *weights*), $\boldsymbol{\theta}$, are coefficients to be estimated during model training. Hyperparameters, $\boldsymbol{\lambda}$, control how the algorithms are run but are not directly updated by them. Hyperparameters can be mathematical, for example the learning rate in a gradient boosting machine (Chapter 14), or structural, for example the depth of a decision tree (Chapter 12). The number

of hyperparameters usually increases with learner complexity and affects its predictive performance. Often hyperparameters need to be tuned (Section 3.5) instead of manually set. Computationally, storing $(\hat{\boldsymbol{\theta}}, \boldsymbol{\lambda})$ is sufficient to recreate any trained model.

Box 1 (Ridge regression)

Let $f : \mathcal{X} \to \mathcal{Y}$ be the regression task of interest with $\mathcal{X} \subseteq \mathbb{R}$ and $\mathcal{Y} \subseteq \mathbb{R}$. Let $(\mathbf{x}, \mathbf{y}) = ((x_1, y_1), \ldots, (x_n, y_n))$ be data such that $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ for all $i = 1, ..., n$. Say the **learner** of interest is a regularized linear regression model with **learning algorithm**:

$$(\hat{\beta}_0, \hat{\beta}_1) := \underset{\beta_0, \beta_1}{\arg\min} \left\{ \sum_{i=1}^{n} \left(y_i - (\beta_0 + \beta_1 x_i)\right)^2 + \gamma \beta_1^2 \right\}.$$

and **prediction algorithm**:

$$\hat{f}(\phi) = \hat{\beta}_0 + \hat{\beta}_1 \phi$$

The **hyperparameters** are $\lambda = (\gamma \in \mathbb{R}_{>0})$ and the **parameters** are $\boldsymbol{\theta} = (\beta_0, \beta_1)^\top$. Say that $\gamma = 2$ is set and the learner is then trained by passing $(\mathbf{x}, \mathbf{y})$ to the learning algorithm and thus estimating $\hat{\boldsymbol{\theta}}$ and $\hat{f}$. A **prediction**, can then be made by passing new data $x^* \in \mathcal{X}$ to the fitted model: $\hat{y} := \hat{f}(x^*) = \hat{\beta}_0 + \hat{\beta}_1 x^*$.

3.4 Evaluating and benchmarking

To understand if a model is 'good', its predictions are evaluated with a *loss function*. Loss functions assign a score to the discrepancy between predictions and true values, $L : \mathcal{Y} \times \mathcal{Y} \to \bar{\mathbb{R}}$. Given (unseen) real-world data, $(\mathbf{X}^*, \mathbf{y}^*)$, and a trained model, $\hat{f}$, the loss is given by $L(\hat{f}(\mathbf{X}^*), \mathbf{y}^*) = L(\hat{\mathbf{y}}, \mathbf{y}^*)$. For a model to be useful, it should perform well in general, meaning its generalization error should be low. The *generalization error* refers to the model's performance on new data, rather than just the data encountered during training and development.

A model should only be used to make predictions if its generalization error was estimated to be acceptable for a given context. If a model were to be trained and evaluated on the same data, the resulting loss, known as the *training error*, would be an overoptimistic estimate of the true generalization error (James et al. 2013). This occurs as the model is making predictions for data it has already 'seen' and the loss is therefore not evaluating the model's ability to generalize to new, unseen data. Estimation of the generalization error requires *data splitting*, which is the process of splitting available data, $\mathcal{D}$, into *training data*, $\mathcal{D}_{train} \subset \mathcal{D}$, and *testing data*, $\mathcal{D}_{test} = \mathcal{D} \setminus \mathcal{D}_{train}$.

The simplest method to estimate the generalization error is to use *holdout resampling*, which is the process of partitioning the data into one training dataset and one testing dataset, with the model trained on the former and predictions made for the latter. Using 2/3 of the data for training and 1/3 for testing is a common splitting ratio (Kohavi 1995). For independent and identically distributed (iid) data, it is generally best practice to partition the data randomly. This ensures that any potential patterns or information encoded in the ordering of the data are removed, as such patterns are unlikely to generalize to new, unseen

data. For example, in clinical datasets, the order in which patients enter a study might inadvertently encode latent information such as which doctor was on duty at the time, which could theoretically influence patient outcomes. As this information is not explicitly captured in measured features, it is unlikely to hold predictive value for future patients. Random splitting breaks any spurious associations between the order of data and the outcomes.

When data is not iid, for example spatially correlated or time-series data, then random splitting may not be advisable, see Hornung et al. (2023) for an overview of evaluation strategies in non-standard settings.

Holdout resampling is a quick method to estimate the generalization error, and is particular useful when very large datasets are available. However, hold-out resampling has a very high variance for small datasets and there is no guarantee that evaluating the model on one hold-out split is indicative of real-world performance.

k-fold cross-validation (CV) can be used as a more robust method to better estimate the generalization error (Hastie, Tibshirani, and Friedman 2001). k-fold CV partitions the data into k subsets, called *folds*. The training data comprises of $k-1$ of the folds and the remaining one is used for testing and evaluation. This is repeated k times until each of the folds has been used exactly once as the testing data. The performance from each fold is averaged into a final performance estimate (Figure 3.2). It is common to use $k = 5$ or $k = 10$ (Leo Breiman and Spector 1992; Kohavi 1995). This process can be repeated multiple times (*repeated k-fold CV*) and/or k can even be set to n, which is known as *leave-one-out cross-validation*.

Cross-validation can also be stratified, which ensures that a variable of interest will have the same distribution in each fold as in the original data. This is important, and often recommended, in survival analysis to ensure that the proportion of censoring in each fold is representative of the full dataset (Casalicchio and Burk 2024; Herrmann et al. 2021).

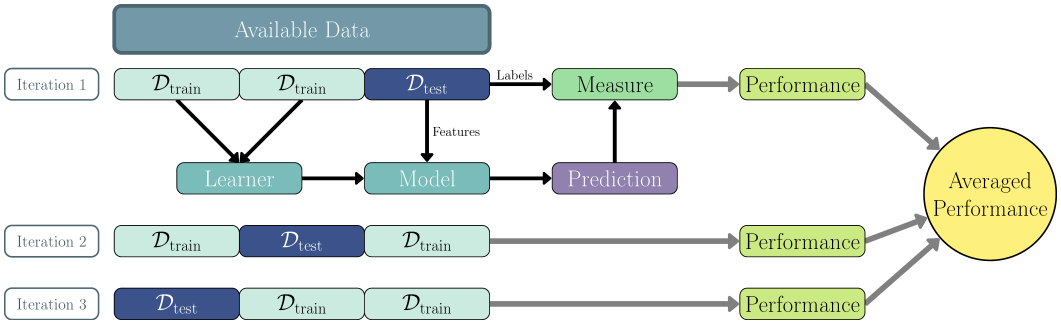


Figure 3.2: Three-fold cross-validation. In each iteration a different dataset is used for predictions and the other two for training. The performance from each iteration is averaged into a final, single metric. Image from Casalicchio and Burk (2024) (CC BY-NC-SA 4.0).

Repeating resampling experiments with multiple models is referred to as a *benchmark experiment*. A benchmark experiment compares models by evaluating their performance on *identical* data, which means the same resampling strategy and folds should be used for all models. Determining if one model is actually better than another is a surprisingly complex topic (Benavoli et al. 2017; Demšar 2006; Dietterich 1998; Nadeau and Bengio 2003) and is out of scope for this book, instead any benchmark experiments performed in this book are purely for illustrative reasons and no results are expected to generalize outside of these experiments. A common heuristic is to suggest one model outperforms another if it performs better across all folds in a repeated cross-validation benchmark experiment, however this is

just a heuristic and without robust hypothesis testing results should be interpreted with caution.

Box 2 (Evaluating ridge regression)

Let $\mathcal{X} \subseteq \mathbb{R}$ and $\mathcal{Y} \subseteq \mathbb{R}$ and let $(\mathbf{x}^*, \mathbf{y}^*) = ((x_1^*, y_1^*), \ldots, (x_m^*, y_m^*))$ be data previously unseen by the model trained in Box 1 where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ for all $i = 1, ..., m$. Predictions are made by passing $\mathbf{x}^*$ to the fitted model yielding $\hat{\mathbf{y}} = (\hat{y}_1, \ldots \hat{y}_m)$ where $\hat{y}_i := \hat{f}(x_i^*) = \hat{\beta}_0 + \hat{\beta}_1 x_i^*$.
Say the mean absolute error is used to evaluate the model, defined by

$$L(\boldsymbol{\phi}, \boldsymbol{\varphi}) = \frac{1}{n} \sum_{i=1}^{n} |\phi_i - \varphi_i|$$

where $(\boldsymbol{\phi}, \boldsymbol{\varphi}) = ((\phi_1, \varphi_1), \ldots, (\phi_n, \varphi_n))$.
The model's predictive performance is then calculated as $L(\hat{\mathbf{y}}, \mathbf{y}^*)$.

3.5 Hyperparameter Optimization

Section 3.3 introduced model hyperparameters, which control how training and prediction algorithms are run. Setting hyperparameters is a critical part of model fitting and can significantly change model performance. *Tuning* is the process of using internal benchmark experiments to automatically select the optimal hyper-parameter configuration. For example, the depth of trees, m_r in a random forest (Chapter 12) is a potential hyperparameter to tune. This hyperparameter may be tuned over a range of values, say $[1, 15]$ or over a discrete subset, say $\{1, 5, 15\}$, for now assume the latter. Three random forests with 1, 5, and 15 tree depth respectively are compared in a benchmark experiment. The depth that results in the model with the optimal performance is then selected for the hyperparameter value going forward. *Nested resampling* is a common method to reduce bias that could occur from using overlapping data for tuning, training, or testing (R. Simon 2007). Nested resampling is the process of resampling the training set again for tuning and then the optimal model is refit on the entire training data (Figure 3.3).

3.6 Conclusion

Key takeaways

- Machine learning tasks define the predictive problem of interest;
- Regression tasks make predictions for continuous outcomes, such as the amount of rain tomorrow;
- Classification tasks make predictions for discrete outcomes, such as the predicted weather tomorrow;
- Both regression and classification tasks may make determiistic predictions (a single

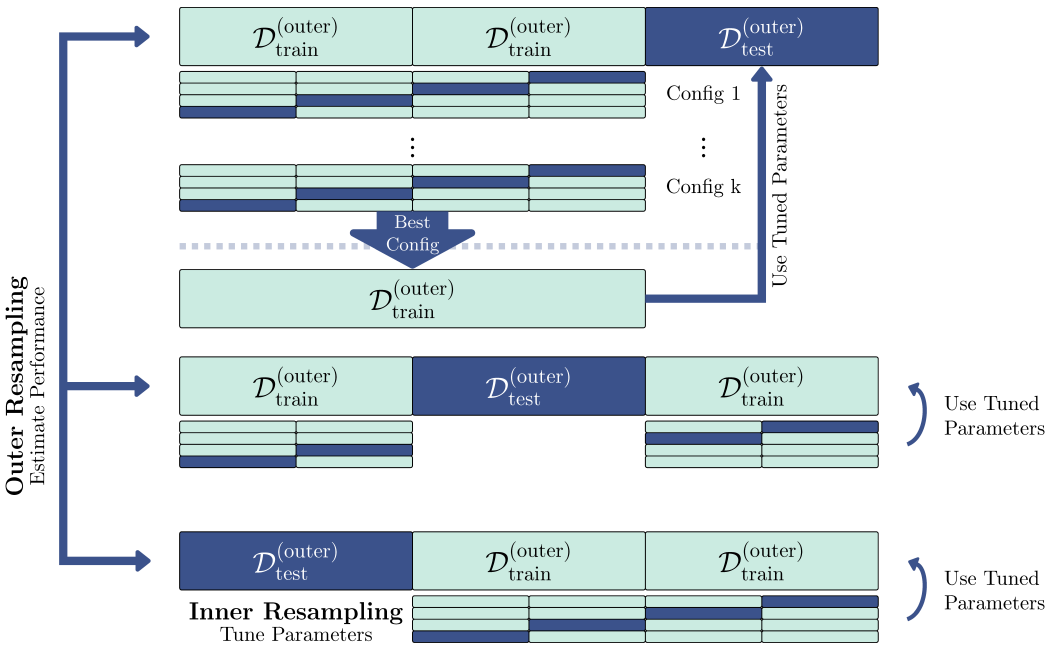


Figure 3.3: An illustration of nested resampling. The large blocks represent three-fold CV for the outer resampling for model evaluation and the small blocks represent four-fold CV for the inner resampling for hyperparameter optimization. The light blue blocks are the training sets and the dark blue blocks are the test sets. Image and caption from Becker, Schneider, and Fischer (2024) (CC BY-NC-SA 4.0).

number or category), or probabilistic predictions (the probability of a number or category);
- Models have parameters that are fit during training and hyperparameters that are set or tuned;
- Models should be evaluated on resampled data to estimate the generalization error to understand future performance.

Further reading

- *The Elements of Statistical Learning* (Hastie, Tibshirani, and Friedman 2001), *An Introduction to Statistical Learning* (James et al. 2013), and *Pattern Recognition and Machine Learning* (Bishop 2006) for comprehensive introductions and overviews to machine learning.
- *Applied Machine Learning Using mlr3 in R* (Bischl et al. 2024) and *Tidy Modeling* (Kuhn and Silge 2023) for machine learning in R
- *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow* (Géron 2019) for machine learning in Python.
- Bischl et al. (2012) for discussions about more resampling strategies including bootstrapping and subsampling.

4

Survival Analysis

TODO (150-200 WORDS)

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

Survival Analysis is concerned with data where the outcome is the time until an event takes place (a 'time-to-event'). Because the collection of such data takes place in the temporal domain (it takes time to observe a duration), the event of interest is often unobservable, for example because it did not occur by the end of the data collection period. In survival analysis terminology this is referred to as *censoring*.

This chapter defines basic terminology and mathematical definitions in survival analysis, which are used throughout this book. Building upon this chapter, Chapter 5 introduces event-history analysis, which is a generalisation to settings with multiple, potentially competing or recurrent events, including multi-state outcomes. Concluding this part of the book, Chapter 6 defines different prediction tasks in survival analysis that are used by models and measures to implement machine learning methods.

While these definitions and concepts are not new to survival analysis, it is imperative they are understood to build successful models. Evaluation functions (Part II) can identify if one model is better suited than another to minimize a given objective function, however they cannot identify if the objective function itself was specified correctly, which depends on the assumptions about the data generating process. Evaluating models with the wrong objective function yields meaningless results. Hence, it is of utmost importance for machine learning practitioners to be able identify and specify the survival problem present in their data correctly to ensure models are correctly fit and evaluated.

4.1 Quantifying the Distribution of Event Times

This section introduces functions that can be used to fully characterise a probability distribution, particular focus is given to functions that are important in survival analysis.

Note that we can generally distinguish between event taking place in discrete time or continuous time. For example, consider the time a politician serves in parliament. If we consider the number of election cycles they stay in parliament, it would constitute discrete time, as time can only take values, $1, 2, 3, \ldots$, that is $Y \in \mathbb{N}_{>0}$. On the other hand, the time

an individual stays in hospital is usually determined as the difference between the admission date-time and discharge date-time, which would constitute a continuous time $Y \in \mathbb{R}_{\geq 0}$.

In practice the differences are often blurred as time-measurement will naturally be discretized at some level and precision beyond some resolution is often not of interest (hospital length of stay might be interesting up to days or hours, but not minutes and seconds). Also discrete-time methods are often applied to continuous time data and vice versa. It is nevertheless important to make the distinction as it informs mathematical treatment and definition of the different quantities introduced below.

4.1.1 Continuous Time

For now, assume a continuous, positive, random variable Y taking values in (t.v.i.) $\mathbb{R}_{\geq 0}$. A standard representation of the distribution of Y is given by the probability density function (pdf), $f_Y : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, and cumulative distribution function (cdf), $F_Y : \mathbb{R}_{\geq 0} \to [0, 1]; (\tau) \mapsto P(Y \leq \tau)$.

In survival analysis, it is most common to describe the distribution of event times Y via the *survival function* and *hazard function* (or *hazard rate*) rather than the pdf or cdf. The survival function is defined as

$$S_Y(\tau) = P(Y > \tau) = \int_\tau^\infty f_Y(u) \, \mathrm{d}u, \tag{4.1}$$

which is the probability that an event has not occurred by $\tau \geq 0$ and thus the complement of the cdf: $S_Y(\tau) = 1 - F_Y(\tau)$. By definition, $S_Y(0) = 1$ and $S(\tau) \to 0$ for $\tau \to \infty$.

The hazard function is given by

$$\begin{aligned} h_Y(\tau) &= \lim_{\mathrm{d}\tau \searrow 0} \frac{P(\tau \leq Y < \tau + \mathrm{d}\tau | Y \geq \tau)}{\mathrm{d}\tau} \\ &= \lim_{\mathrm{d}\tau \searrow 0} \frac{P(Y \in [\tau, \tau + \mathrm{d}\tau) | Y \geq \tau)}{\mathrm{d}\tau} \\ &= \frac{f_Y(\tau)}{S_Y(\tau)} \end{aligned} \tag{4.2}$$

where $\mathrm{d}\tau$ denotes a time-interval. The hazard rate is often interpreted as the instantaneous risk of observing an event at τ, given that the event has not been observed before τ. This is not a probability and h_Y can be greater than one.

The cumulative hazard function (chf) can be derived from the hazard function by

$$H_Y(\tau) = \int_0^\tau h_Y(u) \, \mathrm{d}u, \tag{4.3}$$

and relates to the survival function via

$$H_Y(\tau) = \int_0^\tau h_Y(u) \, \mathrm{d}u = \int_0^\tau \frac{f_Y(u)}{S_Y(u)} \, \mathrm{d}u = \int_0^\tau -\frac{S_Y'(u)}{S_Y(u)} \, \mathrm{d}u = -\log(S_Y(\tau))$$

These last relationships are particularly important, as many methods estimate the hazard rate, which is then used to calculate the cumulative hazard and survival probability

$$S_Y(\tau) = \exp(-H_Y(\tau)) = \exp\left(-\int_0^\tau h_Y(u) \, \mathrm{d}u\right) \tag{4.4}$$

Unless necessary to avoid confusion, subscripts are dropped from S_Y, h_Y etc. going forward and instead these functions are referred to as S, h (and so on).

Usual regression techniques cannot be used to estimate these quantities as Y is only partially observed, due to different types of censoring and truncation, which are now described.

4.1.2 Discrete Time

Now consider a discrete, positive random variable $\tilde{Y}$ taking value in $\mathbb{N}_{>0}$ and $\tau \in \mathbb{N}_{>0}$ some time point in discrete time.

The discrete-time hazard rate

$$h^d_{\tilde{Y}}(\tau) = P(\tilde{Y} = \tau | \tilde{Y} \geq \tau). \tag{4.5}$$

Thus, in contrast to the continuous time hazard 4.2, the discrete time hazard is an actual (conditional) probability, rather than a rate and therefore might be easier to interpret.

The cumulative discrete time hazard is given by

$$H^d_{\tilde{Y}}(\tau) = \sum_{k=1}^{\tau} h^d_{\tilde{Y}}(k) \tag{4.6}$$

We then define the inverse probability

$$s^d_{\tilde{Y}}(\tau) := 1 - h^d_{\tilde{Y}}(\tau) = P(\tilde{Y} > \tau | \tilde{Y} \geq \tau).$$

It follows that the probability to survive beyond time point τ is given by

$$S^d_{\tilde{Y}} = P(\tilde{Y} > \tau) = \prod_{k \leq \tau} s^d_{\tilde{Y}}(\tau) = \prod_{k \leq \tau} (1 - h^d_{\tilde{Y}}(\tau)), \tag{4.7}$$

and the unconditional probability for an event at time τ is

$$P(\tilde{Y} = \tau) = S^d_{\tilde{Y}}(\tau - 1) h^d_{\tilde{Y}}(\tau). \tag{4.8}$$

When applied to continuous time Y, the follow-up is divided in J disjunct intervals $(a_0, a_1], \ldots, (a_{j-1}, a_j] \ldots, (a_{J-1}, a_J], j = 1, \ldots, J$ such that

$$Y \in (a_{j-1}, a_j] \Leftrightarrow \tilde{Y} = j$$

Thus,

$$h^d_{\tilde{Y}}(j) = P(Y \in (a_{j-1}, a_j] | Y > a_{j-1})$$

and

$$S^d_{\tilde{Y}}(j) = P(Y > a_j) = S_Y(a_j)$$

For more details on discrete time-to-event analysis consider (Tutz and Schmid 2016).

4.2 Single-event, right-censored data

The complexity of Survival Analysis compared to other fields arises from the fact that the outcome of interest is often only observed partially. In particular, the time-to-event is often unknown at the end of the observation period, as the event has not occurred yet.

Let,

- X taking values in $\mathbb{R}^p$ be the generative random variable representing the data *features/covariates/independent variables*.
- Y taking values in $\mathbb{R}_{\geq 0}$ be the (partially unobservable) *true survival time*.
- C taking values in $\mathbb{R}_{\geq 0}$ be the (partially unobservable) *true censoring time*.

In the presence of censoring C, it is impossible to fully observe the true outcome of interest, Y. Instead, the observable variables are defined by

- $T := \min\{Y, C\}$, the *outcome time* (realisations are referred to as the *observed outcome time*); and
- $\Delta := \mathbb{I}(Y = T) = \mathbb{I}(Y \leq C)$, the *event indicator* (also known as the *censoring* or *status* indicator).

Together (T, Δ) is referred to as the *survival outcome* or *survival tuple* and they form the dependent variables. The survival outcome provides a concise mechanism for representing the outcome time and indicating which outcome (event or censoring) took place.

A *survival dataset* is a $n \times p$, eal-valued matrix defined by $\mathcal{D} = ((\mathbf{x}_1,\ t_1,\ \delta_1) \cdots (\mathbf{x}_n, t_n, \delta_n))^\top$, where (t_i, δ_i) are realisations of the respective random variables (T_i, Δ_i) and $\mathbf{x}_i$ is a p-dimensional vector, $\mathbf{x}_i = (x_{i;1}\ x_{i;2} \cdots x_{i;p})^\top$ of features.

Finally, the following quantities are used frequently throughout this book and survival analysis literature more generally. Let $(t_i, \delta_i) \overset{i.i.d.}{\sim} (T, \Delta), i = 1, ..., n$, be observed survival outcomes. Then,

The **set of unique outcome times** is the set of time-points in which at least one observation experiences the event or is censored:

$$\mathcal{U}_O \subseteq \{t_i\}_{i \in \{1,\ldots,n\}}$$

The **set of unique event times** is the set of time-points in which at least one observation experiences the event (but not censored):

$$\mathcal{U}_D \subseteq \{t_i : \delta_i = 1\}_{i \in \{1,\ldots,n\}}$$

The **ordered, unique events times** may also be denoted by

$$t_{(k)},\ k = 1, \ldots, m \quad t_{(1)} < t_{(2)} < \cdots < t_{(m)}, \quad m \leq n$$

The **risk set at** τ, is the index-set of observation units at risk for the event just before τ

$$\mathcal{R}_\tau := \{i : t_i \geq \tau\} \tag{4.9}$$

where i is the index of an observation in the data. For right-censored data, $\mathcal{R}_0 = \{1, \ldots, n\}$ and $\mathcal{R}_\tau \subseteq \mathcal{R}_{\tau'}, \forall \tau > \tau'$. Note that in a continuous setting, 'just before' refers to an infinitesimally smaller time than τ, in practice as this is unobservable the risk set is defined at τ, hence an observation may both be at risk, and experience an event (or be censored) at τ.

The **number of observations at risk at** τ is the cardinality of the risk set at τ,

$$n_\tau := \sum_i \mathbb{I}(t_i \geq \tau) = |\mathcal{R}_\tau|$$

Finally, the **number of events at** τ is defined by,

$$d_\tau := \sum_i \mathbb{I}(t_i = \tau, \delta_i = 1)$$

For truly continuous variables, one might expect only one event to occur at each observed event time: $d_{t_i} = 1, \forall i$. In practice, ties are often observed due to finite measurement precision, such that $d_\tau > 1$ occurs frequently in real-world datasets.

The quantities $\mathcal{R}_\tau$, n_τ, and d_τ underlie many models and measures in survival analysis. Several non-parametric and semi-parametric methods (Chapter 11) like the Kaplan-Meier estimator (see Section 4.3) are based on the ratio d_τ / n_τ.

Table 4.1 exemplifies an observed survival dataset, a subset of the `tumor` data (Bender and Scheipl 2018), which contains the time until death in days after operation ($\delta_i = 1$ if death occurred at the outcome time t_i and $\delta_i = 0$ otherwise).

In this example, the above quantities would be:

- $\mathcal{U}_0 = \{268, 397, 519, 1217, 2414\}$: with 1217 included only once
- $\mathcal{U}_D = \{268, 397, 1217\}$: with the inclusion of 1217 due to the event at t_1, not censoring at t_5
- $\mathcal{R}_{\tau=1217} = \{1, 3, 5\}$ (these subjects' outcome times are greater or equal to $\tau = 1217$ so they are at risk for the event at this time)
- $n_{\tau=1217} = |\mathcal{R}_{1217}| = 3$
- $d_{\tau=1217} = 1$: As only $i = 1$ experienced the event (and not censoring) at this time.

Table 4.1: Subset of the `tumor` (Bender and Scheipl 2018) time-to-event dataset. Rows are individual observations (ID), $\mathbf{x}_{;j}$ columns are features, t is observed time-to-event, δ is the event indicator.

id (i)	**age** ($\mathbf{x}_{;1}$)	**sex** ($\mathbf{x}_{;2}$)	**complications** ($\mathbf{x}_{;3}$)	**days** ($\mathbf{t}$)	**status** ($\boldsymbol{\delta}$)
1	71	female	no	1217	1
2	70	male	no	519	0
3	67	female	yes	2414	0
4	58	male	no	397	1
5	39	female	yes	1217	0
6	59	female	no	268	1

4.3 Kaplan-Meier estimator

Before we go further, we introduce a simple, non-parametric estimator for the survival function (4.1), the Kaplan-Meier estimator (Kaplan and Meier 1958). The estimator is useful for visualising survival data and is a popular baseline model to compare to the predictive performance of more complex methods. In machine learning terms it can be viewed as a "featureless" learner for survival analysis.

Using the quantities introduced in Section 4.2, the estimator is defined by:

$$\hat{S}_{KM}(\tau) = \prod_{k:t_{(k)} \leq \tau} \left(1 - \frac{d_{t_{(k)}}}{n_{t_{(k)}}}\right) \tag{4.10}$$

which is a step-function at the observed ordered event times $t_{(k)}, k = 1, \ldots, m$ with $\hat{S}_{KM}(\tau) = 1 \; \forall \tau < t_{(1)}$. It is usually estimated for all unique event times.

For illustration, Figure 4.1 shows the estimated survival probability obtained by applying the Kaplan-Meier estimator to the full `tumor` data, containing observations of $n = 776$ subjects. By definition, the survival function starts at $S(t) = 1$ at $t = 0$ and monotonically decreases towards $S(t) = 0$ for $t \to \infty$. Dotted lines indicated the median survival, defined as the time at which the survival function reaches $S(t) = 0.5$. In this example, the median survival time is approximately 1500 days, as indicated by the dashed lines. This means that 50% of the subjects are expected die within 1500 days after operation.

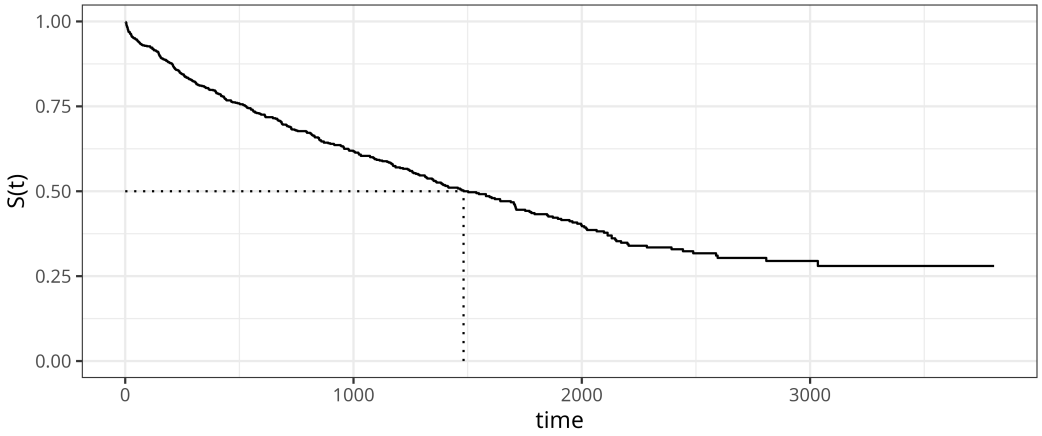


Figure 4.1: Kaplan-Meier estimate for the `tumor` data (Bender and Scheipl 2018). The estimated survival probabilities are given by the solid step-function. Dotted lines indicate the median survival time).

While the Kaplan-Meier estimator does not directly support estimation of covariate effects on the estimated survival probabilities, it is often used for descriptive analysis by applying the estimator to different subgroups (referred to as stratification in survival analysis). For example, Figure 4.2 shows $\hat{S}_{KM}$ separately for subjects aged 50 years or older and subjects younger than 50 years, respectively. Dashed lines again illustrate median survival times.

However, note that the median survival time does not exist for the younger age group, as their estimated survival function does not cross 0.5.

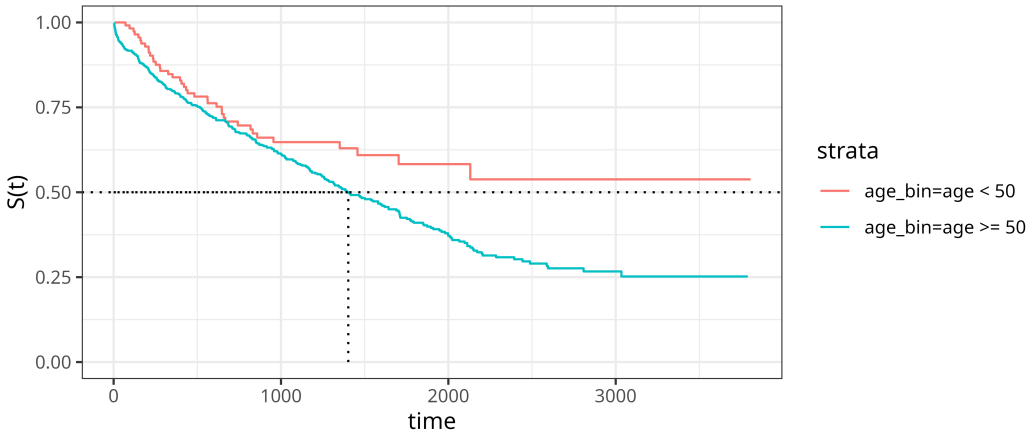


Figure 4.2: Kaplan-Meier estimate for the `tumor` data (Bender and Scheipl 2018) applied to subgroups of subjects of 50 or more years old and less than 50 years old, respectively. The estimated survival probabilities are given by the solid step-function. Dotted lines indicate the median survival time).

Sometimes the KM Estimator is also used to estimate the distribution of censoring times, for example to calculate inverse probability of censoring weights (Section 7.1.1). Throughout this book, we denote the Kaplan-Meier estimator applied to the observed censoring times $(t_i, 1 - \delta_i)$ as $\hat{G}_{KM}$.

4.4 Types of Censoring

Three types of censoring are commonly defined in survival analysis: right-censoring, left-censoring, and interval-censoring. The latter can be viewed as the most general case. Multiple types of censoring and/or truncation (Section 4.5) can occur in any given data set and it is vital to identify which types are present in order to correctly select and specify models and measures for the data.

Right-censoring

Right-censoring is the most commonly assumed form of censoring in survival data. It occurs when the event of interest was not experienced during the observation period, which may happen because it was no longer observable (for example, due to withdrawal from the study) or because the event did not happen until study end. The exact event time is unknown but it *is* known that the event is after the observed censoring time, hence *right*-censoring (imagine a timeline from left to right as in Figure 4.3).

Right-censoring can be further divided into Type-I, Type-II and random censoring. Type-I, or *administrative*, censoring occurs at the fixed, pre-defined end of an observation period τ_u, in which case the outcome is given by $(T_i = \min(Y_i, \tau_u), \Delta_i = \mathbb{I}(Y_i \leq \tau_u))$. Censored

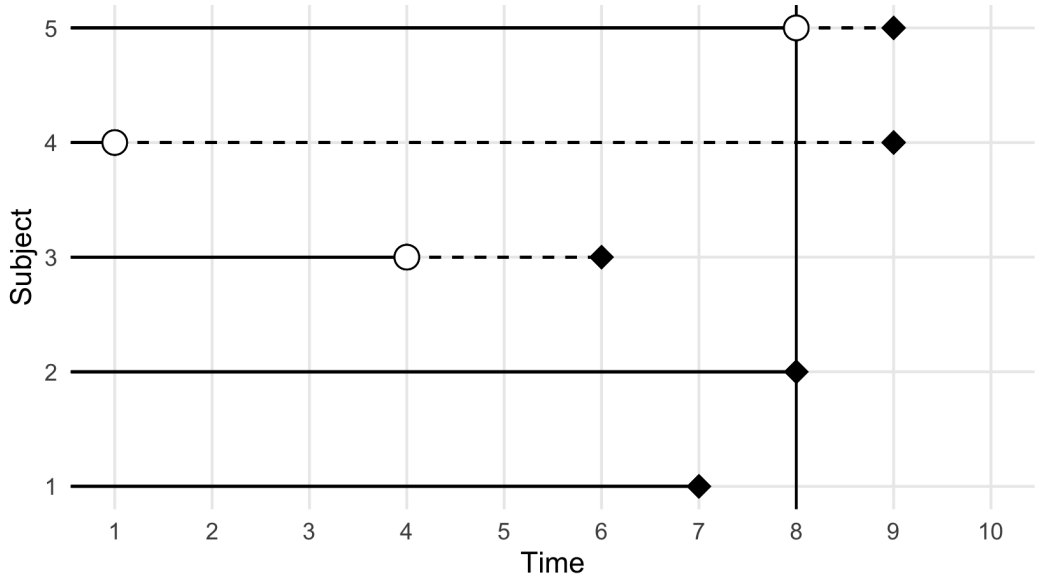


Figure 4.3: Dead and censored subjects (y-axis) over time (x-axis). Black diamonds indicate true death times and white circles indicate censoring times. Diamonds at the end of dashed lines are hypothetical (unknown in reality). Vertical line is the study end time. Subjects 1 and 2 die in the study time. Subject 3 is censored in the study and (unknown) dies within the study time. Subject 4 is censored in the study and (unknown) dies after the study. Subject 5 is censored at study end and (unknown) dies after the end of the study.

observations are therefore represented as $(\tau_u, 0)$. Type-II censoring also occurs when the observation period ends. However, in this case the study ends when a pre-defined number of subjects experienced the event of interest and hence τ_u is random.

Random censoring occurs when censoring times *randomly* follow an unknown distribution and one observes $(T_i = \min(Y_i, C_i), \Delta_i = \mathbb{I}(Y_i \leq C_i))$. Different types of right-censoring can, and sometimes do, co-occur in any given data set.

In practice, these different types of right-censoring are usually handled the same during modeling and evaluation and so this book refers to 'right-censoring' generally, which could occur from a combination of the above types.

Left- and Interval-censoring

Left-censoring occurs when the event is known to have happened at some unknown time before observation time, interval-censoring occurs when the event is known to have happened within some time span, but not the exact time.

Consider a survey about phone use where participants are asked: "How old were you when you used a smart phone for the first time?". The possible answers are:

- exact age of first use
- didn't use a smart phone yet
- did or does use a phone but doesn't remember age of first time use
- did or does use a phone, remembers a specific age range

The first case represents an exactly observed event time, the second case the familiar right-censoring, as the event may occur later in life, but it is unknown when. The third case is referred to as left-censoring, we know the event occurred before the interview, but don't know when. The fourth case is an example of interval-censoring, as we know the event occurred within some age span, but not the exact age.

Interval- and left-censoring also often occurs in medical contexts. For example, some guidelines suggest annual screenings for skin cancer (starting from a certain age). However, the initial age at which individuals do screenings and regularity of check ups varies widely. If cancer was detected between two screenings, the observation is interval-censored. If cancer is detected at first screening, the observation is left-censored (unless the 'age' of the cancer can be narrowed down based on size and other characteristics, in which case it would become interval-censored).

Censoring Notation

In the presence of left- or interval-censoring the usual representation of survival outcomes as (t_i, δ_i) is not sufficient to denote the different types of observations in the data set. Instead, we represent the data as intervals in which the event occurs. Formally, let Y_i the random variable for time until the event of interest and L_i, R_i random variables that define an interval $(L_i, R_i]$ with realisations l_i, r_i. Let further t_i the time of observation (for example age at interview in the phone use example) or last-follow up time for subject i. Then the event time of subject i is

- left-censored if $Y_i \in (L_i = 0, R_i = t_i]$;
- right-censored at t_i if $Y_i \in (L_i = t_i, R_i = \infty)$;
- interval-censored $Y_i \in (L_i = l_i, R_i = r_i]$, $l_i < r_i \leq t_i$
- exactly observed if $Y_i \in (L_i = t_i, R_i = t_i]$

In the cancer screening example from above it holds that $r_i = t_i$ and l_i the last check-up before r_i. In the phone use example, the participants might specify any age range between 0 and t_i.

To make this more concrete, consider phone use example data, where t_i is the age at interview. In practice, such data is often stored by creating two variables representing the left (l_i) and right border (r_i) of the respective intervals (t_i is not really needed here to define the outcome, but included for illustration).

id	t_i	l_i	r_i
1	13	13	∞
2	17	15	15
3	16	14	16
4	16	13	15
5	18	0	18

Here, the first subject is right-censored at 13 years ($l_i = t_i = 13, r_i = \infty$), the second subject remembered exactly ($l_i = r_i = 15 < t_i = 17$), the third subject remembers that it was after 14, but not exact age ($l_i = 14 < r_i = 16 = t_i$), fourth subject remembers use after 13 years and latest at 15 years of age ($l_i = 13 < r_i = 15 < t_i$), and the fifth subject uses a smart phone currently at age 18, but cannot specify further ($l_i = 0 < r_i = 18 = t_i$).

From the example above, it is clear, that right- and left-censoring are special cases of interval-censoring. However, if only right- or left-censoring is present, the likelihood and estimation simplifies (see Section 4.6). Also note that in case of left- and interval-censoring the event is known to have occurred, while for right-censoring the event didn't occur during time under observation. For left- and right-censoring one might be tempted to consider it an event $\delta_i = 1$ without exact time, while right-censoring would be consider a non-event $\delta_i = 0$. However, technically it is assumed that the event will always occur, if we wait long enough (for right-censored data in the interval $(t_i, \infty]$). Censoring therefore means having imprecise information about the time of event rather than information about the event occurring or not occurring.

Dependent vs. Informative Censoring

Censoring may be defined as *uninformative* if $Y \perp\!\!\!\perp C$ and *informative* otherwise. However, these definitions can be misleading as the term 'uninformative' could imply that C is independent of both X *and* Y, and not just Y. To avoid misinterpretation, the following definitions are used in this book:

- If $C \perp\!\!\!\perp X$, censoring is *feature-independent*, otherwise censoring is *feature-dependent*.
- If $C \perp\!\!\!\perp Y$, censoring is *event-independent*, otherwise censoring is *event-dependent*.
- If $(C \perp\!\!\!\perp Y)|X$, censoring is conditionally independent of the event given covariates, or *conditionally event-independent*.
- If $C \perp\!\!\!\perp (X, Y)$, censoring is *uninformative*, otherwise censoring is *informative*.

Uninformative censoring can generally be well-handled by models as the true underlying distribution of survival times is not affected by censoring. In fact, in this case one could

even use regression models after removing censored observations (if they do not form a high proportion of the data).

In reality, censoring is rarely non-informative as reasons for drop-out or missingness in outcomes tend to be related to the study of interest. Event-dependent censoring is a tricky case that, if not handled appropriately (by a competing-risks framework), can easily lead to poor model development. Imagine a study is interested in predicting the time between relapses of stroke but a patient suffers a brain aneurysm due to some separate neurological condition. There is a high possibility that a stroke may have occurred if the aneurysm had not. A survival model is unlikely to distinguish the censoring event (aneurysm) from the event of interest (stroke) and will confuse predictions.

In practice, the majority of models and measures assume that censoring is conditionally event-independent and hence censoring patterns can be predicted/estimated based on the covariates. For example, if studying the survival time of ill pregnant patients in hospital, then dropping out of the study due to pregnancy is clearly dependent on how many weeks pregnant the patient is when the study starts (for the sake of argument assume no early/late pregnancy due to illness).

4.5 Censoring vs. Truncation

A common confusion is to conflate censoring and truncation, which is problematic as the methods to handle them differ substantially. Outside of time-to-event settings, truncation usually refers to truncating (or removing) an entire subject from a dataset. As discussed in Chapter 1, truncation in survival analysis refers to partially truncating a period of time and is quite common in the more general event history setting (Chapter 5).

While censored observations have incomplete information about the time-to-event, they are still part of the data set. Whereas truncation leads to observations not entering the data set (at least not at time 0). This will usually introduce bias that needs to be accounted for.

Left-truncation

Left-truncation often occurs when inclusion into a study is conditional on the occurrence of another event. Left-truncation plays an important role when modeling recurrent events or multi-state data (Chapter 5), thus the concept will be introduced in more detail.

By example, consider a study from the 18th century (Broström 1987), when childhood and maternal mortality were relatively high. The goal of the study was to establish the effect of a mother's death on the survival of the infant. Since each death was reported to the authorities, an infant was added to the study if and when their mother died. To create a matched cohort, two other infants, whose mothers were alive, were matched into the study based on their age and other relevant features. Thus, groups of three infants within the study had identical features except for the status of the mother (alive or dead). Because of the study design, infants who died before their mothers could never enter into the study. A mother's death is thus referred to as left-truncation event and the infant's age at time of inclusion into the study is referred to as left-truncation time.

More formally, let t_i^L the subject-specific left-truncation time. Then we only observe subjects with $y_i > t_i^L$ and subjects with $y_i < t_i^L$ never enter the data.

This is illustrated in Figure 4.4. Continuing with the example above, say Infant 1 dies at t_1, while the mother dies at some later time point t_1^L, therefore infant 1 never enters the study. The mother of Infant 2 dies at t_2^L, at which point the infant is included in the study and experiences an event at t_2. Finally, say Infant 3 enters the study at t_3^L and is censored at 365 days when the study ends.

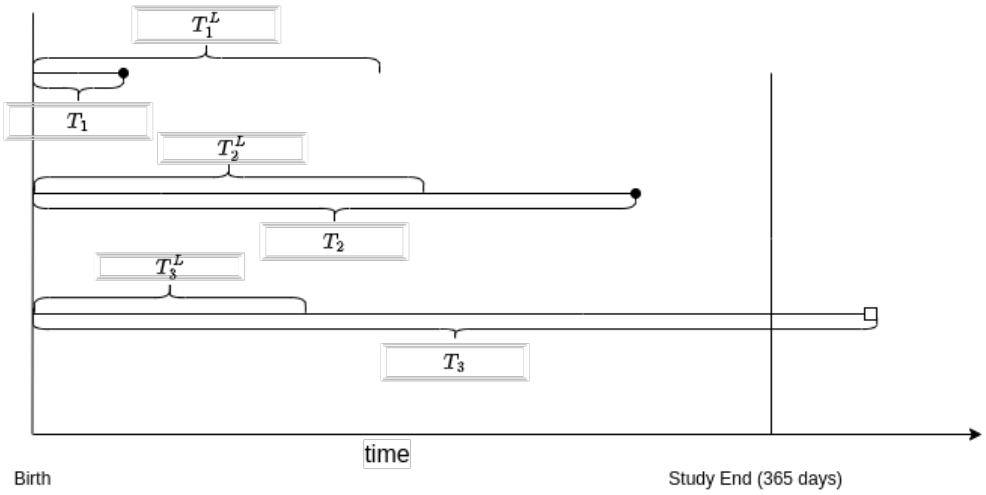


Figure 4.4: Illustration of left-truncation data. Subjects 2 and 3 enter the data set as the study entry condition (mother's death) occurs before the event of interest occurs after the left-truncation time (left-truncation time shorter than event time). Subject 1 on the other hand never enters the data as the event occurs before the study entry condition (left-truncation time longer than event time).

In this example, left-truncation biases the sample towards healthier or more robust infants, as frail infants on die earlier and thus on average before their mothers. This would bias the estimates if not properly taken into account.

Continuing the above example, Figure 4.5 shows the difference between estimated survival probabilities when left-truncation is ignored (left panel) and taken into account (right panel), respectively. It is clear that the survival probabilities were underestimated in both groups, but more so in the group of infants whose mother died (thereby underestimating the effect of the mothers' death on infant survival).

To understand this, consider an excerpt from the infants data in Table 4.3.

Table 4.3: Excerpt of the `infants` (Broström 2024) time-to-event dataset. Rows are individual observations (id), `group` indicates matched infants, t^L is the left-truncation time (time of inclusion into the study), t is the observed time, δ is the event indicator.

group	id	t^L	t	δ	mother
1	1	55	365	0	dead
1	2	55	365	0	alive
1	3	55	365	0	alive
2	4	13	76	1	dead
2	5	13	365	0	alive

group	id	t^L	t	δ	mother
2	6	13	365	0	alive
4	7	2	16	1	dead
4	8	2	365	0	alive
4	9	2	365	0	alive

Now recall the calculation of the Kaplan-Meier estimate from Section 4.3. In Table 4.3, without stratifying according to mother's status, the first observed event time is $t_{(1)} = t_7 = 16$. Then ignoring left-truncation,

- $\mathcal{R}_{t_{(1)}} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $d_{t_{(1)}} = 1$
- $n_{t_{(1)}} = 9$
- $S(t_{(1)}) = 1 - \frac{1}{9} \approx 0.9$

In contrast, when we take left-truncation into account, subjects only enter the risk set for the event after their left-truncation time (we already know they survived until t_i^L so they are not at risk for the event before that time), thus

- $\mathcal{R}_{t_{(1)}} = \{4, 5, 6, 7, 8, 9\}$
- $d_{t_{(1)}} = 1$
- $n_{t_{(1)}} = 6$
- $S(t_{(1)}) = 1 - \frac{1}{6} \approx 0.8$

Thus, the definition of the KM estimator in 4.10 can still be used in case of left-truncation using the more general risk set definition

$$\mathcal{R}_\tau = \{i : t_i^L \ \leq \tau \leq t_i\} \tag{4.11}$$

Right-truncation

Right-truncation often occurs in retrospective sampling based on registry data, when data is queried for cases reported by a certain cut-off time (see for example Vakulenko-Lagun, Mandel, and Betensky (2020)). A common example is the estimation of the incubation period of an infectious disease, which is the time from infection to the disease onset. Only known, symptomatic (and/or tested) cases are entered into the database. At a time τ, one can only observe the subset of the infected population that has already experienced the disease, and not the population that is still incubating the disease, hence biasing the data to shorter incubation periods.

Formally, let t_i^r be the right-truncation time (here time from infection until the time at which the database is queried), then subjects only enter the data set when $t_i < t_i^r$. This is illustrated in Figure 4.6 using three subjects. All three subjects were infected during the observation period, however, the right-truncation time t_2^r of subject 2 is shorter than the incubation period t_2 for this subject, thus at the time of querying the data base, this subject will not be included in the sample, as $t_2 > t_2^r$.

Note the difference to right-censoring. If subject 2 was right-censored, the subject would be in our sample and the time of infection would be known - the time of disease onset would be censored. In case of right-truncation on the other hand, the subject is not included in the sample at time of data extraction, as subjects are only included in the registry after disease

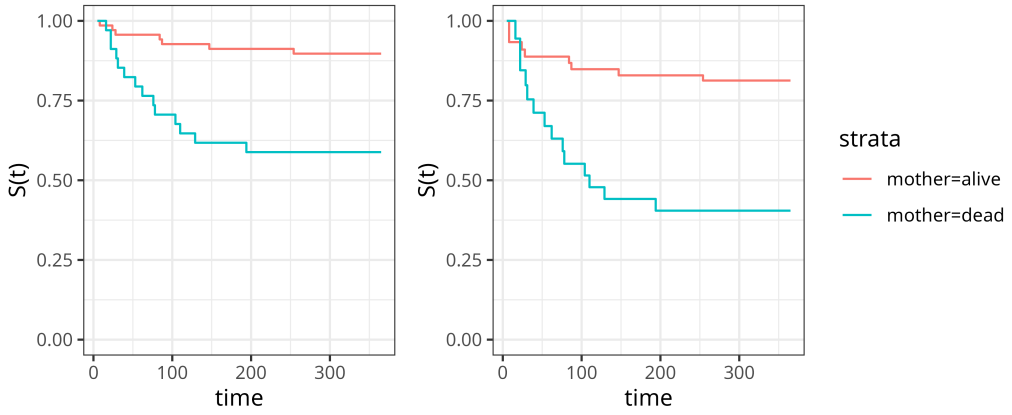


Figure 4.5: Left: Kaplan-Meier estimate of survival probabilities of infants depending on status of the mother ignoring left-truncation. Right: Kaplan-Meier estimate of survival probabilities adjusting for left-truncation.

onset. Overall this leads to a bias towards shorter incubation times and potentially feature values that lead to shorter incubation times.

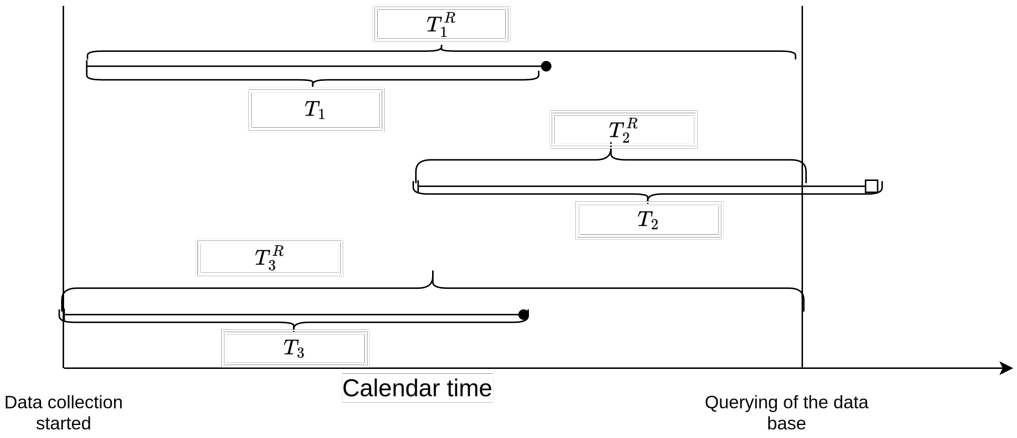


Figure 4.6: Illustration of right-truncation based on registry data. For subjects 1 and 3 the right-truncation time is longer than the incubation period, therefore they are included in the sample when the registry is queried. For subject 2 on the other hand the right-truncation time is shorter, therefore it's excluded from the sample.

As for left-truncation, ignoring right-truncation will lead to biased estimations. While for left-truncated data simple adjustments of the risk set work for non- and semi-parametric methods like the Kaplan-Meier and Cox-type estimators, this is not the case for right-truncated data. However, parametric methods can be employed (see Section 4.6) and generalised product limit estimators for right-truncated data exist (Michael G. Akritas and LaValley 2005).

4.6 Estimation

While details about estimation will be given later, when different models are introduced in Part III and Part IV of the book, it is worthwhile discussing some general concepts here, namely parametric and non-parametric approaches.

4.6.1 Parametric estimation

Consider for now uncensored data $(t_i, \delta_i = 1), i = 1, \ldots, n$. A standard approach would be to assume a suitable distribution for the event times $t_i \overset{iid}{\sim} F_Y(\boldsymbol{\theta}), \boldsymbol{\theta} = (\theta_1, \theta_2, \ldots)^\top$, and define the likelihood of the data as

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{n} f_Y(t_i|\boldsymbol{\theta}) \tag{4.12}$$

where f_Y is the pdf of F_Y.

The model parameters can then be obtained by maximizing the likelihood such that

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

However, in the presence of censoring 4.12 is incorrect, as the exact event time is only known for some subjects. For example, for right-censored data ($\delta_i = 0$) we only know that the event occurred after observed censoring time t_i. Thus the likelihood contribution for such data points is $P(Y_i > t_i) = S_Y(t_i)$, whereas for observed event times ($\delta_i = 0$) the likelihood contribution is $f_Y(t_i)$ as before.

Let now $\mathcal{O} = \{i : \delta_i = 1\}$ the index set of observed event times and $\mathcal{RC} = \{i : \delta_i = 0\}$ the index set of censored observations. The likelihood of this data can be written as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}) &\propto \prod_{i \in \mathcal{O}} f_Y(t_i|\boldsymbol{\theta}) \prod_{i \in \mathcal{RC}} S_Y(t_i|\boldsymbol{\theta}) \\
&= \prod_{i=1}^{n} f_y(t_i|\boldsymbol{\theta})^{\delta_i} S_Y(t_i|\boldsymbol{\theta})^{1-\delta_i} \\
&= \prod_{i=1}^{n} \frac{f_y(t_i|\boldsymbol{\theta})^{\delta_i}}{S_Y(t_i|\boldsymbol{\theta})^{\delta_i}} S_Y(t_i|\boldsymbol{\theta}) \\
&= \prod_{i=1}^{n} h_Y(t_i|\boldsymbol{\theta})^{\delta_i} S_Y(t_i|\boldsymbol{\theta}),
\end{aligned}$$

where the last equality follows from 4.2.

Similar adjustments to the likelihood can be made for other types of censoring and in the presence of truncation. Following Klein and Moeschberger (2003), we can define the individual likelihood contributions for the different types of censoring as

- observed event at t_i: $f_Y(t_i|\boldsymbol{\theta})$
- right-censoring: $P(Y_i > t_i|\boldsymbol{\theta}) = S_Y(t_i|\boldsymbol{\theta})$
- left-censoring: $P(Y_i < t_i|\boldsymbol{\theta}) = F_Y(t_i|\boldsymbol{\theta}) = 1 - S_Y(t_i|\boldsymbol{\theta})$

- interval-censoring: $P(l_i < Y_i \leq r_i|\boldsymbol{\theta}) = S_Y(l_i|\boldsymbol{\theta}) - S_Y(r_i|\boldsymbol{\theta})$

Depending on which of the above contributions occur in the data set, we can now construct our likelihood accordingly. Let $\mathcal{O}, \mathcal{RC}, \mathcal{LC}, \mathcal{IC}$ non-overlapping subsets of the observed data $\mathcal{D}$ for subjects with observed event times, right-censoring, left-censoring and interval-censoring, respectively. Assuming independence between observations and in absence of truncation, the likelihood for the observed data can be defined as

$$\mathcal{L}(\boldsymbol{\theta}) \propto \prod_{i \in \mathcal{O}} f_Y(t_i|\boldsymbol{\theta}) \prod_{i \in \mathcal{RC}} S_Y(t_i|\boldsymbol{\theta}) \prod_{i \in \mathcal{LC}} (1 - S_Y(t_i|\boldsymbol{\theta})) \prod_{i \in \mathcal{IC}} (S_Y(l_i|\boldsymbol{\theta}) - S_Y(r_i|\boldsymbol{\theta})) \tag{4.13}$$

In case of truncation, the adjustments are made to all observations, as we have to condition on the event occurring after/before the truncation time. The truncation adjusted likelihood contributions (assuming independence of truncation and event/censoring times) would thus be given by

- left-truncation:
  - event: $f(Y_i = t_i|Y_i \geq t_i^L, \boldsymbol{\theta}) = \frac{f_Y(t_i|\boldsymbol{\theta})}{S_Y(t_i^L|\boldsymbol{\theta})}$
  - left-censoring: $P(Y_i < t_i|Y_i \geq t_i^L, \boldsymbol{\theta}) = \frac{S_Y(t_i|\boldsymbol{\theta})}{S_Y(t_i^L|\boldsymbol{\theta})}$
  - interval-censoring: $P(l_i < Y_i \leq r_i|Y_i \geq t_i^L, \boldsymbol{\theta}) = \frac{S_Y(l_i|\boldsymbol{\theta}) - S_Y(r_i|\boldsymbol{\theta})}{S_Y(t_i^L|\boldsymbol{\theta})}$
- right-truncation:
  - event: $f(Y_i = t_i|Y_i \leq t_i^R, \boldsymbol{\theta}) = \frac{f_Y(t_i|\boldsymbol{\theta})}{F_Y(t_i^R|\boldsymbol{\theta})}$
  - left-censoring: $P(Y_i < t_i|Y_i \leq t_i^R, \boldsymbol{\theta}) = \frac{S_Y(t_i|\boldsymbol{\theta})}{F_Y(t_i^R|\boldsymbol{\theta})}$
  - interval-censoring: $P(l_i < Y_i \leq r_i|Y_i \leq t_i^R, \boldsymbol{\theta}) = \frac{S_Y(l_i|\boldsymbol{\theta}) - S_Y(r_i|\boldsymbol{\theta})}{F_Y(t_i^R|\boldsymbol{\theta})}$

Note that in in practice, data sets will often not contain all types of censoring or truncation, in which case 4.13 will contain only a subset of the product terms. This has been illustrated in 4.13 under absence of truncation and $\mathcal{LC} = \mathcal{IC} = \emptyset$.

4.6.2 Non-parametric estimation

As the name suggests, non-parametric estimation (and semi-parametric estimation) techniques do not make (strong) assumptions about the underlying distribution of event times.

A common principle for such techniques is to partition the follow-up into intervals or to define specific time-points during the follow-up and to estimate the continuous or discrete time hazards (4.5) for each interval/time-point. Then derive the respective estimates, for example the survival probability, based on the relationship between the hazard and other quantities (Section 4.1 or Section 4.1.2).

The Kaplan-Meier estimator introduced earlier (Section 4.3) is an example for this principle. There, time-points $t_{(k)},\ k = 1, \ldots, m$ are used to calculate the discrete time hazards

$$h^d(t_{(k)}) = P\left(Y \in (t_{(k-1)}, t_{(k)}]|Y > t_{(k-1)}\right) = \frac{d_{t_{(k)}}}{n_{t_{(k)}}}. \tag{4.14}$$

The survival probability and the definition of the Kaplan-Meier estimator (4.10) then follow from 4.7.

Similarly, the Nelson-Aalen estimator (Nelson (1972),O. Aalen (1978)) for the cumulative hazard is obtained via 4.6 as

$$H_{NA}(\tau) = \sum_{k:t_{(k)} \leq \tau} h^d(t_{(k)}) = \sum_{k:t_{(k)} \leq \tau} \frac{d_{t_{(k)}}}{n_{t_{(k)}}}. \tag{4.15}$$

Note that we use a continuous time valued τ for the definition, which implies that $H_{NA,e}(\tau) = H_{NA,e}(t_{(k)})\ \forall \tau \in [t_{(k)}, t_{(k+1)})$. In words: For time points between two unique event times we assume the previous value of the estimate. Similar to the Kaplan-Meier estimator, H_{NA} can deal with left-truncated data based on the general risk-set definition 4.11.

Based on 4.4 we can also define a survival probability estimator based on the Nelson-Aalen estimator of the cumulative hazard 4.15 as

$$S_{NA}(\tau) = \exp(-H_{NA}(\tau)). \tag{4.16}$$

This relationship is also used by Breslow (1972) to obtain survival probability estimates in the context of the Cox model (Chapter 11).

In the prescence of other censoring types there are a number of alternatives to the Kaplan-Meier estimator.

Interval censoring

For interval-censoring, the Kaplan-Meier estimator is replaced by the nonparametric maximum likelihood estimator (NPMLE) (Z. Zhang and Sun 2010), which estimates the likelihood function under the assumption of left- or interval-censoring data as in Section 4.6.1. The most common algorithm to compute this is the iterative Turnbull algorithm (Turnbull 1974). The intuition behind this algorithm is that an observation, i, should only contribute to the estimation at a given time-point, τ, if the event could have happened in the censoring interval, $\tau \in (L_i, R_i]$.

In summary:

Let $t_{(k)}, k = 1, ..., m$ be the ordered times at which to evaluate the NPMLE.

1. Make an initial guess for $\hat{S}^0(t_{(k)}), k = 1, ..., m$.

Where $\hat{S}^0$ is the initial guess and $\hat{S}^j$ is the estimated survival function at iteration j.

2. **While** $|\hat{S}^j(t_{(k)}) - \hat{S}^{j-1}(t_{(k)})| > \epsilon, \forall k = 1, ..., m$ **Do:**

a. Compute the probability of an event at each time-point:

$$p_{t_{(k)}} = \hat{S}^{j-1}(t_{(k-1)}) - \hat{S}^{j-1}(t_{(k)}), \quad k = 1, ..., m$$

b. Estimate the number of events at each time-point:

$$d_{t_{(k)}} = \sum_{i=1}^{n} \frac{w_{ik} p_{t_{(k)}}}{\sum_{l=1}^{m} w_{il} p_{t_{(l)}}}, \quad k = 1, ..., m$$

Where $w_{ik} = 1$ if the interval $(t_{(k-1)}, t_{(k)}]$ is contained within $(L_i, R_i]$, and $w_{ik} = 0$ otherwise; and L_i, R_i are the left- and right-censoring times for observations $i = 1, ..., n$.

c. Compute the cumulative number of events at or after $t_{(k)}$:

$$n_{t_{(k)}} = \sum_{l=k}^{m} d_{t_{(l)}}, \quad k = 1, ..., m$$

This serves a similar role to the usual 'number at risk' formula.n or after $t_{(k)}$.

d. Calculate $\hat{S}^j$ using the Kaplan-Meier formula (4.10), substituting the values in Steps 2b and 2c.

Giolo (2004) suggest using the Kaplan-Meier to find the initial guess in Step 1 and to use $\epsilon = 10^{-3}$. A similar algorithm can be used for left-censored data.

Left truncation

In the presence of left-truncation, the Kaplan-Meier is calculated using the left-truncated risk-set definition (McGough et al. 2021), which excludes individuals from the risk set until their left-truncation time has occurred (4.11). The number of individuals at risk at τ is then:

$$n_{\tau}^{L} = \sum_{i} \mathbb{I}(t_i^L \leq \tau \leq t_i)$$

where t_i, t_i^L are the outcome time and left truncation time of observation i respectively. This equation can be substituted into the Kaplan-Meier formula (4.10), with $d_{t_{(k)}}$ calculated in the usual manner.

4.7 Conclusion

Key takeaways

•

Further reading

- For practical discussion about survival models in the context of right-censoring and left-truncation see McGough et al. (2021)

5

Event-history Analysis

TODO (150-200 WORDS)

> ! Major changes expected!
>
> **This page is a work in progress and major changes will be made over time.**

In this chapter we take a more general view on time-to-event data. So far, we only considered a single potentially censored, outcome of interest. Here we explore more complex settings with multiple, potentially mutually exclusive events and recurrences of events. In this generalization, the observed data is sometimes referred to as *event-history data* and its analysis as *event-history analysis*.

One way to think about event history data is in terms of transitions between different states, as illustrated in Figure 5.1. Usually, a subject starts out in an initial state 0 (for example, 'healthy') and from there transitions to different states. States from which further transitions are possible are called *transient* (displayed as circles), otherwise a state is called *terminal* or *absorbing* (displayed as squares).

In the *single event* setting (Figure 5.1, upper left panel), a subject can only transition to one state (the event of interest). This setting was the focus of Chapter 4. There, the censoring event was considered independent of the event of interest. In the *competing risks* setting (Figure 5.1, upper right panel, Section 5.2), a subject could transition to any of the q mutually exclusive states, thus the subject is initially *at risk* for a transition to multiple states. Once one of them occurs, the process is considered to have concluded (for the modeling purposes).

In the *recurrent events setting* (Figure 5.1 lower left panel), the same event can be observed multiple times on the same subject (for example recurrent respiratory infections during one year). Two different ways to represent recurrent events are shown: (top) reset the status to 0 after occurrence of an event or (bottom) consider the 1st, 2nd, etc. recurrences of the event as separate states. A detail omitted in the graph: Often recurrent event processes also have a competing, absorbing event. In this more complex setting, but also in general, recurrent events are often represented as multi-state process, which we discuss next. Therefore we forgo detailed discussion of this setting in this book and refer to Cook and Lawless (2007) for a detailed account specific to recurrent events analysis.

In the most general case, the *multi-state* setting (Figure 5.1 lower right panel, Section 5.3), there are multiple transient and terminal states with potential back transitions (for example, moving between different stages of an illness with the possibility of (partial) recovery and death as terminal event).

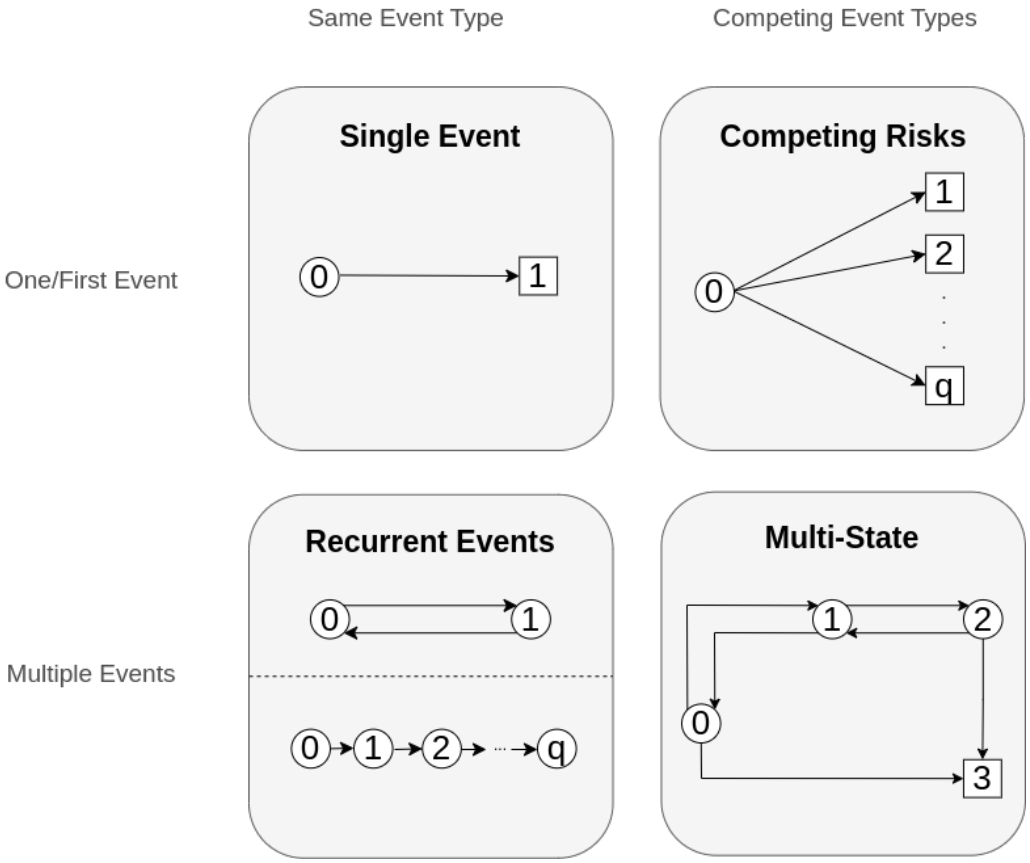


Figure 5.1: Illustration of different types of time-to-event processes. Transient states are displayed as circles, absorbing states are displayed as squares. Top, left: Standard single-event setting with transition from initial state 0 to state 1; Top, right: Competing risks setting with q competing events. The follow-up ends once one of the $\{1, \ldots, q\}$ events is observed or the study ends; Bottom, left: Recurrent events setting with multiple occurrences of the same event. Bottom, right: Multi-state setting where subjects can transition between multiple transient states with possible back-transitions or to absorbing states.

Note that the concepts discussed in Section 4.4 and Section 4.5 are still relevant here, as, dependent on the specific process, any transition between two states could be subject to different types of censoring and truncation. In particular, remaining in one of the transient states until the end of follow-up constitutes right-censoring with respect to all possible transitions from that state and left-truncation is particularly important as subjects enter the risk sets for a transition at different time points in context of recurrent events and multi-state settings.

5.1 A process point of view

In order to formalize the different settings more conveniently, we introduce the stochastic process

$$E(\tau) \in \{0, \ldots, q\},\ \tau \geq 0, \tag{5.1}$$

which indicates the state that is occupied at time τ.

Using this notation in the single-event setting we get $E(\tau) \in \{0, 1\}$ such that the hazard 4.2 could be written as

$$\begin{aligned} h(\tau) &= \lim_{\mathrm{d}\tau \searrow 0} \frac{P(Y \in [\tau, \tau + \mathrm{d}\tau) | Y \geq \tau)}{\mathrm{d}\tau} \\ &= \lim_{\mathrm{d}\tau \searrow 0} \frac{P\left(E(\tau + \mathrm{d}\tau) = 1 | E(\tau-) = 0\right)}{\mathrm{d}\tau}, \end{aligned} \tag{5.2}$$

where $\tau-$ indicates the time point immediately before τ.

This notation doesn't yield many advantages in the single-event setting, but will shorten notation later on, particularly in the multi-state setting.

5.2 Competing Risks

In contrast to single-event survival analysis, competing risks are concerned with the time to the first of multiple, mutually exclusive events.

Table 5.1. shows an excerpt of the `sir.adm` data (Allignol, Beyersmann, and Schumacher 2008) of patients on an intensive care unit (ICU). Time under observation (`time`) could end in one of three outcomes: 1 (discharge alive), 2 (death on ICU) or 0 (neither discharge nor death at the end of follow-up, which constitutes right-censoring at the end of study). The interest was in how pneumonia status (`pneumonia`) at admission to the ICU affects mortality.

Table 5.1: Subset of the `sir.adm` dataset (Allignol, Beyersmann, and Schumacher 2008). Each row represent one subject,`time` is the time under observation, `status` indicates the outcome observed (0: censored at the end of the study, 1: discharged alive from the ICU, 2: death in the ICU). `pneumonia` indicates whether a subject already had pneumonia at ICU admission.

time	status	pneumonia
8	0	no
8	0	no
31	1	yes
5	1	no
9	2	no
5	2	no

Contrast this data to the `tumor` data example in Section 4.3. There, patients were followed even after hospital discharge, thus loss to follow-up could be considered reasonably independent of the event of interest (death). In this study follow-up stopped once patients were discharged. As discharged patients are healthier compared to the ones who remain on ICU, assuming independence between the time until discharge and time until death is unrealistic. Analysis of this data and how the different assumptions (independent censoring vs. competing risks) affects the estimates is discussed in Section 5.2.2 and Section 5.2.4.

5.2.1 Notation and Definitions

In the competing risks setting, everyone starts out in the initial state 0 and can progress to one of the absorbing states $1, \ldots, q$. The goal is to characterize the process $E(\tau) \in \{0, \ldots, q\}$ in terms of transition hazards and probabilities.

In extension of 5.2, we define *cause-specific* hazards

$$h_e(\tau) = \lim_{\mathrm{d}\tau \to 0} \frac{P(E(\tau + \mathrm{d}\tau) = e \mid E(\tau-) = 0)}{\mathrm{d}\tau}. \tag{5.3}$$

Analogous to the single-event case, we can also define the cause-specific cumulative hazard

$$H_e(\tau) = \int_0^\tau h_e(u)\ du \tag{5.4}$$

As competing events are mutually exclusive at any time τ, it is possible to define the *all-cause hazard* which is the hazard of any event occurring as the sum of all cause-specific hazards

$$h(\tau) = \sum_{e=1}^{q} h_e(\tau), \tag{5.5}$$

as well as the *all-cause cumulative hazard*, which can be obtained either via the integral over the all-cause hazard (5.5) or as sum of cause-specific cumulative hazards (5.4):

$$H(\tau) = \sum_{e=1}^{q} H_e(\tau) = \sum_{e=1}^{q} \int_0^\tau h_e(u)\ du = \int_0^\tau \sum_{e=1}^{q} h_e(u)\ du \int_0^\tau h(u)\ du \tag{5.6}$$

The *all-cause survival probability* gives the probability that *none* of the events occurred before τ. This is usually not estimated directly but calculated from the cause specific hazards instead via 5.6 and 5.7:

$$S(\tau) = P(Y > \tau) = \exp(-H(\tau)) \tag{5.7}$$

Finally, the probability of experiencing an event e before time τ, *which is often referred to as Cumulative Incidence Function* (CIF), is given by

$$\begin{aligned} F_e(\tau) &= P(E(\tau) = e) \\ &= \int_0^\tau f_e(u)\ \mathrm{d}u = \int_0^\tau S(u-)h_e(u)\ \mathrm{d}u, \end{aligned} \tag{5.8}$$

where

- $S(u-)$ is the probability of surviving (not experiencing any of the competing events) until the time-point shortly before u
- $f_e(u)\mathrm{d}u = S(u-)h_e(u)\mathrm{d}u$ is the probability of experiencing event e at time point u (which follows analogously to 4.2).

Note that here we use the notation $S(u-)$ rather than $S(u)$ to make explicit that we want the probability to survive until the time point immediately before u. This doesn't make much difference in continuous time where $P(T > t) = P(T \geq t)$, but may be important in (discrete) approximations (as in Section 5.2.2).

$F_e(\tau)$ can be interpreted as the proportion of subjects who experienced event of type e until time τ. Because the events are mutually exclusive, it holds that

$$\sum_{e=1}^{q} F_e(\tau) + S(\tau) = F(\tau) + S(\tau) = 1,$$

where $F(\tau)$ is the probability that an event of any type occurring before τ and $S(\tau)$ the probability that no event occurs (5.7).

Note that all terms of 5.8 can be calculated from the individual hazards (5.3). Many estimation procedures for the CIF take this approach, consequently referred to as cause-specific hazards approach.

5.2.2 Non-parametric estimators

Non-parametric estimators for the cause-specific (cumulative) hazard (5.4) in the competing risks setting are derived analogous to the single event case (Section 4.6.2). The CIF then follows from 5.8.

First, recall from Section 4.2 the definitions of the unique ordered event times $t_{(k)}, k = 1 \ldots, m$, the risk-set at time $t_{(k)}$, $\mathcal{R}_{t_{(k)}}$, the number of events, $d_{t_{(k)}}$, and number of observations at risk $n_{t_{(k)}}$. Assume partitioning of the follow-up into m disjunct intervals $(t_{(k-1)}, t_{(k)}], k = 1, \ldots, m$, such that $Y \in (t_{(k-1)}, t_{(k)}] \Leftrightarrow \tilde{Y} = t_{(k)}$, with $\tilde{Y}$ defined as in Chapter 18..

An estimate for the *cause-specific* hazard is derived by updating the numerator in 4.14 to $d_{e,t_{(k)}}$ (the number of events of type e at time $t_{(k)}$):

$$h_e^d(t_{(k)}) := \frac{d_{e,t_{(k)}}}{n_{t_{(k)}}},\ e \in \{1, \ldots, q\}. \tag{5.9}$$

The Nelson-Aalen estimator (4.15) for the *cause-specific* cumulative hazard is then given by

$$H_{NA,e}(\tau) = \sum_{k:t_{(k)} \leq \tau} h_e^d(t_{(k)}) = \sum_{k:t_{(k)} \leq \tau} \frac{d_{e,t_{(k)}}}{n_{t_{(k)}}}, \tag{5.10}$$

which yields a step function for each e, with jumps at time points $t_{(k)}$.

The all-cause survival probability follows from 5.6 and 5.7 as

$$S_{NA}(\tau) = \exp\left(-\sum_{e=1}^{q} H_{NA,e}(\tau)\right).$$

Finally, the Aalen-Johansen (AJ) estimator (O. O. Aalen and Johansen (1978)) for the CIF follows via 4.8 as

$$F_{AJ,e}(\tau) = \sum_{k:t_{(k)} \leq \tau} S_{NA}(\tau-) h_e^d(\tau) = \sum_{k:t_{(k)} \leq \tau} S_{NA}(t_{(k-1)}) \frac{d_{e,t_{(k)}}}{n_{t_{(k)}}} \tag{5.11}$$

5.2.3 Application to mortality of ICU patients

For illustration of the AJ estimator and the interpretation of the CIFs consider the analysis conducted in Beyersmann, Allignol, and Schumacher (2012), based on the data from Table 5.1. Recall that one is interested in estimation of the mortality conditional on pneumonia status at admission, while accounting for discharge from the ICU as competing risk ($E \in \{$"discharge", "death"$\}$). While the AJ estimator cannot naturally incorporate feature information, it can be applied to subgroups of the data (here based on the pneumonia status). Note that this will yield different sets of unique event times in each group, thus the AJ can have jumps at different time-points for the two groups.

TODO: clean up text below

Figure 5.2 shows the AJ estimates of the CIFs for each event type (discharge/death) stratified by pneumonia status. Exemplary, the proportion of subjects with pneumonia being discharged until $\tau = 120$ days is approximately 75% ($\hat{F}_{\text{discharge}}(120) = P(Y \leq 120, E = \text{"discharge"}) \approx 0.75$), while approximately 25% died in the ICU ($\hat{F}_{\text{death}}(120) = P(Y \leq 120, E = \text{"death"}) \approx 0.25$). For patients without pneumonia we have $\hat{F}_{\text{discharge}}(120) \approx 0.91$ and $\hat{F}_{\text{death}} \approx 0.09$. In this example, $\hat{F}_{\text{discharge}} + \hat{F}_{\text{death}} \approx 1$ for both pneumonia groups, as only 14 of 747 patients were censored (neither discharge nor death) at the end of the follow-up.

5.2.4 Independent Censoring vs. Competing Risks

It is worth spending some time to consider the difference between independent right-censoring and competing risks. Note that for the estimation of the hazard (5.9), occurrences of competing events are implicitly assumed right-censored (as $d_{e,t_{(k)}}$) only counts events of type e and $n_{t_{(k)}}$ contains the same subjects that would remain if events of type $\tilde{e} \neq e$ were considered censored before $t_{(k)}$. Nevertheless, competing risks are taken into account in the definition of the AJ estimator (5.11), as the all-cause survival probability (5.7) depends on all cause-specific hazards.

In contrast, assume that in our analysis of the `sir.adm` data we would consider time of discharge as independent right-censoring. As we only have one other event (death), the data could be treated as single-event, right-censored data as in Chapter 4 and therefore analyzed

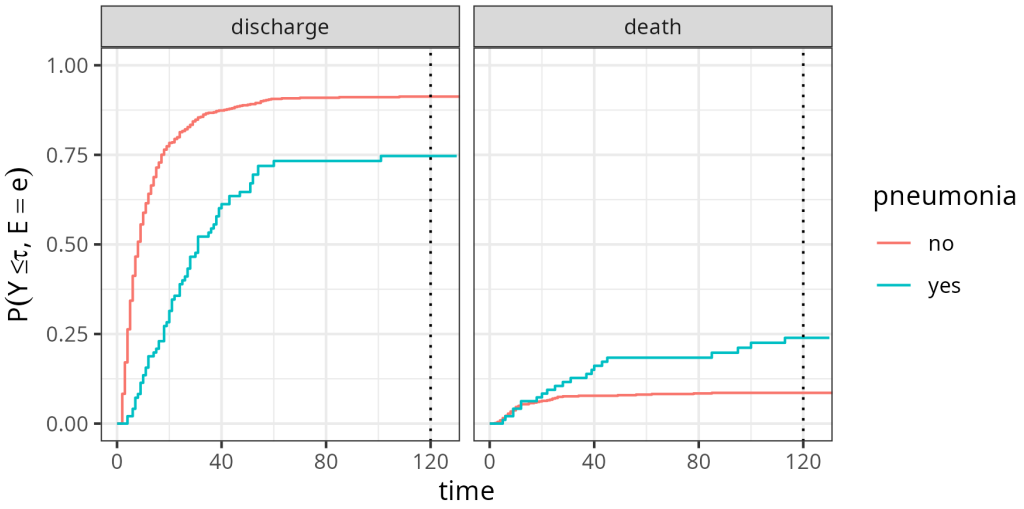


Figure 5.2: Aalen-Johansen estimator for the `sir.adm` data (Allignol, Beyersmann, and Schumacher 2008), stratified by pneumonia status at admission to the ICU. Left panel: Proportion of subjects discharged alive from the ICU. Right panel: Proportion of subjects who died in the ICU.

using the Nelson-Aalen estimator (4.16). The probability of death before some time-point τ could thus be obtained via $P(Y \leq \tau) = F(\tau) = 1 - S_{NA}(\tau)$.

Figure 5.3 shows the estimates obtained under the two assumptions. Solid lines indicate the probabilities under the competing risks assumption (identical to the right-hand side of Figure 5.2). Dashed lines are obtained under the independent right-censoring assumption. Clearly, the probabilities of dying at time $\tau = 120$ are greater when independent censoring is assumed ($\approx$ 75% vs. $\approx$ 25% in the pneumonia group and $\approx$ 62% vs. $\approx$ 13% in the no pneumonia group).

5.3 Multi-state Models

The multi-state process can be considered the most general type of time-to-event process, as other types (single-event, competing risks, recurrent events) can be viewed as special cases. Multi-state modeling allows realistic depiction of complex processes where subjects can start in different states and transition back and forth between them.

For illustration consider the `prothr` dataset (de Wreede, Fiocco, and Putter 2011) of liver cirrhosis patients from a randomized clinical trial with possible transitions depicted in Figure 5.4. Patients may have normal (state 0) or abnormal (state 1) levels of prothrombin (a protein important for blood clotting, produced by the liver) at the beginning of the trial. Some patients where treated with prednisone (which suppresses immune response and reduces inflammation) and others received a placebo. Death (state 2) constitutes an absorbing state.

The goal of the trial was to investigate if treatment (prednisone) slows down or reverses

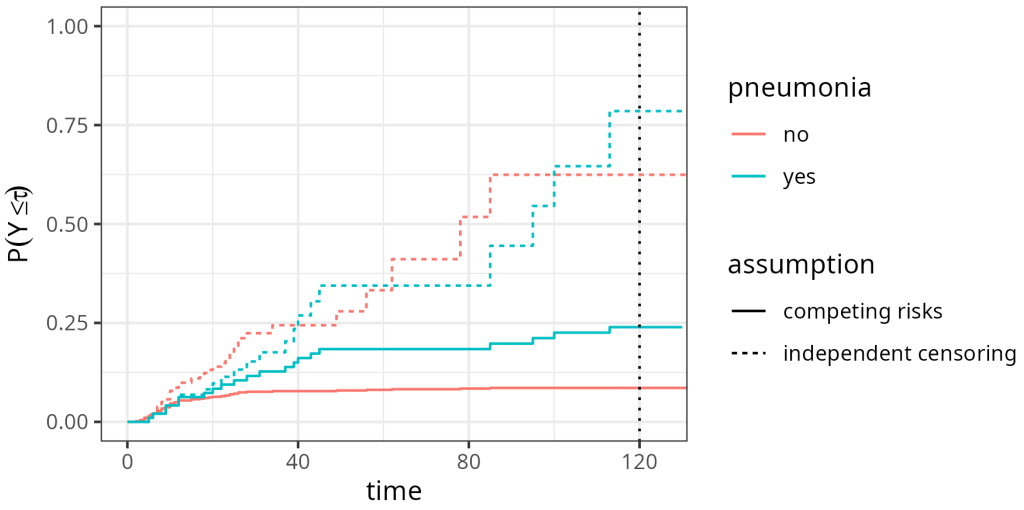


Figure 5.3: Estimation of the probability of dying in the ICU conditional on pneumonia status at admission. Dashed lines give the probabilities under assumption of right-censoring. Solid lines give the probabilities when taking into account discharge as competing risk.

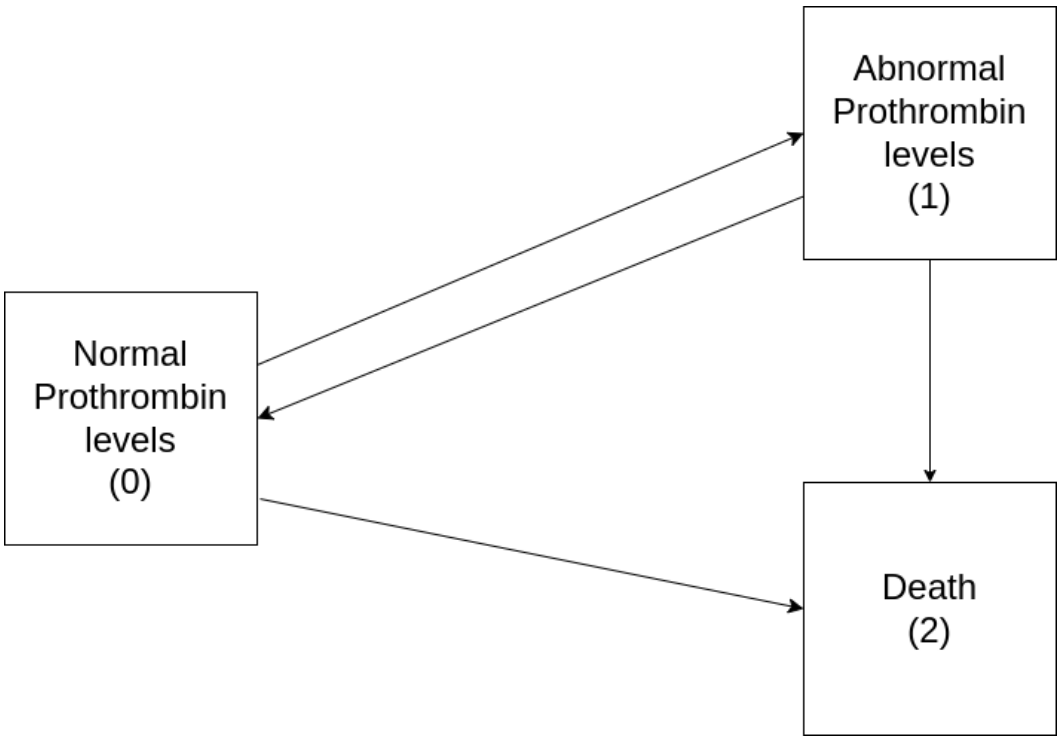


Figure 5.4: Transition graph for the liver cirrhosis patients.

disease progression (transitions $0 \to 1$ and $1 \to 0$) and reduces mortality (transitions $0 \to 2$ and $1 \to 2$).

Table 5.2 shows a subset of the data set and contains for each subject (`id`)one row for each transition for which the subject was at risk for. In this example, this includes transitions that were possible, but didn't happen (counterfactual transitions). The columns `from` and `to` indicate the initial state and the possible end state. `tstart` indicates the time at which the subject entered the risk set for said transitions and `tstop` the time point at which the subjects exited the `from` state (or were censored for any transition). The variable `status` indicates whether the transition was actually made (`status = 1`) or not (`status = 0`). This is necessary, as all possible transitions are listed, so we need an indicator for which transition actually occurred. If `status=0` for all possible transitions, the subject is censored for further transitions. Finally, `treatment` indicates whether a patient was assigned the treatment or placebo group.

Concretely, subject `id=1` already had abnormal prothrombin levels at the beginning of the trial, thus started in state 1 with possible transitions $1 \to 0$ and $1 \to 2$. In this case, the patient died, thus transition $1 \to 2$ was realized after 151 days, while the transition $1 \to 0$ is a 'counterfactual' transition that could have happened in the time-span between `tstart=0` and `tend=151`, but didn't. Patient `id=8` also started in state 1, but made a back transition to normal prothrombin levels after 211 days at which time they entered the risk set for transitions $0 \to 1$ and $0 \to 2$. Neither of the transitions occurred, as `status=0` for both transitions, which means the subject remained in status 0 until the end of their follow-up at 2770 days (that is was right-censored at 2770 days). Finally, subject `id=46` started in state 0 (normal prothrombin levels), transitioned to state 1 (abnormal levels) after 415 days and then died (transition $1 \to 2$) two days later. This also illustrates the importance of left-truncation (Section 4.5) in multi-state processes. For example, subjects `id=1` and `id=8` are at risk for the transitions $1 \to 0$ and $1 \to 2$ from the beginning of the trial (`tstart = 0`). Subject `id=46` on the other hand starts in state 0 and only enters state 1 (and thus the risk set for the transitions $1 \to 0$ and $1 \to 2$) after 415 days (`tstart = 415`). Other subjects in the data may never enter the risk set for these transitions by remaining in state 0 until the end of follow up or by directly transitioning to state 2. The fact that subjects enter the risk sets for different transitions at different time points technically constitutes left-truncation and thus should be taken into account accordingly (Section 5.3.4).

Table 5.2: Subset of the `prothr` dataset (de Wreede, Fiocco, and Putter 2011).

id	from	to	trans	tstart	tstop	status	treatment
1	1	0	3	0	151	0	Placebo
1	1	2	4	0	151	1	Placebo
8	1	0	3	0	211	1	Prednisone
8	1	2	4	0	211	0	Prednisone
8	0	1	1	211	2770	0	Prednisone
8	0	2	2	211	2770	0	Prednisone
46	0	1	1	0	415	1	Prednisone
46	0	2	2	0	415	0	Prednisone
46	1	0	3	415	417	0	Prednisone
46	1	2	4	415	417	1	Prednisone

5.3.1 Notation and Definitions

In the competing risks setting, we characterized the data generating process by cause-specific transitions hazards $h_e(\tau)$ (5.3). Implicitly these are transition from starting state 0 to end state e, however, since everyone starts in stats in state 0 this information is ignored. In the multi-state setting on the other hand, subjects can be in different states at different time points. Transitions between different states are therefore characterized by transition-specific hazards, denoted by $h_{\ell\to e}(\tau)$ or short $h_{\ell e}(\tau)$, where ℓ is the starting state and e the end state, $\ell, e \in \{0, \ldots, q\}$ (ignoring details such as the starting state must be a transient state and not all states are reachable from each starting state).

Let $E(\tau) \in \{0, \ldots, q\}$ be the state process as before (5.1). Then, the transition-specific hazard can be defined as

$$h_{\ell e}(\tau) = \lim_{\mathrm{d}\tau\to 0} \frac{P(E(\tau + \mathrm{d}\tau) = e | E(\tau-) = \ell)}{\mathrm{d}\tau} \tag{5.12}$$

Transition hazards 5.12 indicate the relative risk to enter state e at time τ given occupation of state ℓ at $\tau-$, which is the instant before τ.

Analogous to the competing risks setting, we can define the transition specific cumulative hazards

$$H_{\ell e}(\tau) = \int_0^\tau h_{\ell e}(u)du \tag{5.13}$$

The probability to transition from state ℓ to e between two time-points depends on all transitions possible from ℓ and potentially the transitions that have taken place in the past. Thus, other quantities of interest in the multi-state setting are the transition probabilities $P_{\ell e}(\zeta, \tau) := P(E(\tau) = e | E(\zeta) = \ell)$, that is the probability to transition from state ℓ to state e between time points $\zeta < \tau$. Implicitly, this notation assumes that the process is Markovian: the transition probability depends only on the state at ζ and not any additional past states. Extensions do exist that relax this assumption, for example by including information about the past, but are not relevant for now.

Transition probabilities of a multi-state process are often summarized in a matrix

$$\mathbf{P}(\zeta, \tau) := \begin{pmatrix} P_{00}(\zeta,\tau) & \cdots & P_{0q}(\zeta,\tau) \\ \vdots & \ddots & \vdots \\ P_{q0}(\zeta,\tau) & \cdots & P_{qq}(\zeta,\tau) \end{pmatrix}, \tag{5.14}$$

where rows indicate starting states and columns indicate end states. Some of the elements of $\mathbf{P}$ might be zero or one depending on the specific process, presence of absorbing states and possible pathways between states. As subjects can only be in one of the $q+1$ states at τ, rows sum to 1:

$$\sum_{e=0}^{q} P_{\ell e} = 1, \forall \ell \in \{0, \ldots, q\}. \tag{5.15}$$

5.3.2 Instantaneous transition probabilities

In this section we briefly recap how the transition probabilities can be expressed as the product (integral) of instantaneous transition probabilities

$$p_{\ell e}(\tau) = P\big(E(\tau + \mathrm{d}\tau) = e | E(\tau-) = \ell\big) \tag{5.16}$$

which can be recognized as the nominator of 5.12, or intuitively as the transition probability between two subsequent time points.

For illustration, consider what is often referred to as an illness-death model depicted in Figure 5.5 (similar to Figure 5.4, but without back-transition), where subjects can transition from healthy state 0 to absorbing state death (2) either directly or via intermediate state 1.

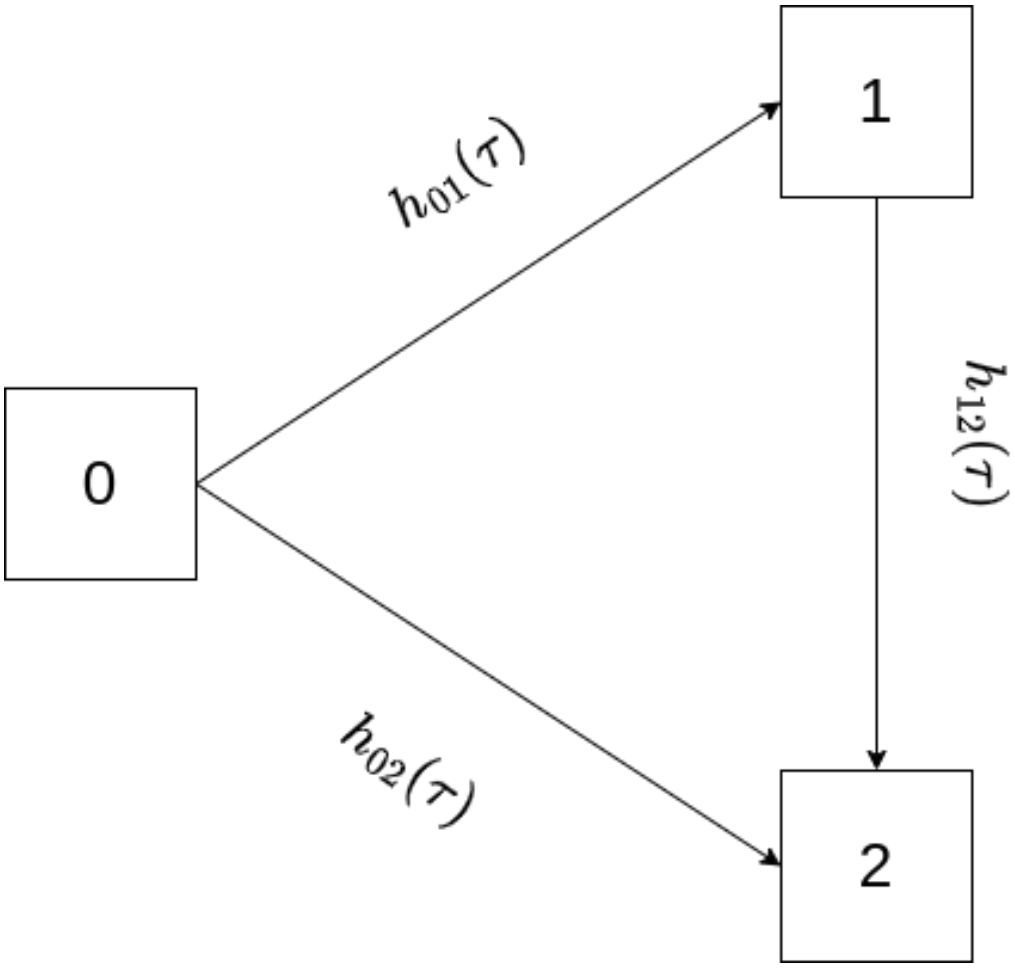


Figure 5.5: An *illness-death model* where subjects can transition from healthy state (0) to death (2) directly or via intermediate illness state (1)

In this example, back-transitions are not possible, therefore the lower triangle of the matrix is filled with zeros and $P_{22}(\zeta, \tau) = 1, \forall\ \zeta < \tau$ by virtue of being an absorbing state. The matrix of transition probabilities is thus given as

$$\mathbf{P}(\zeta, \tau) = \begin{pmatrix} P_{00}(\zeta, \tau) & P_{01}(\zeta, \tau) & P_{02}(\zeta, \tau) \\ 0 & P_{11}(\zeta, \tau) & P_{12}(\zeta, \tau) \\ 0 & 0 & 1 \end{pmatrix} \tag{5.17}$$

First, assume that data is collected in discrete time, that is $\zeta, \tau \in \{0, 1, 2, \ldots\}, \zeta < \tau$ and transitions only occur at these discrete time points and not in between. Say we are interested in transition probability $P_{02}(4, 6)$, that is the probability to transition from state 0 to state 2

between time points $\zeta = 4$ and $\tau = 6$, given we are in state 0 at time $\zeta = 4$. This is possible in the three ways depicted in Figure 5.6.

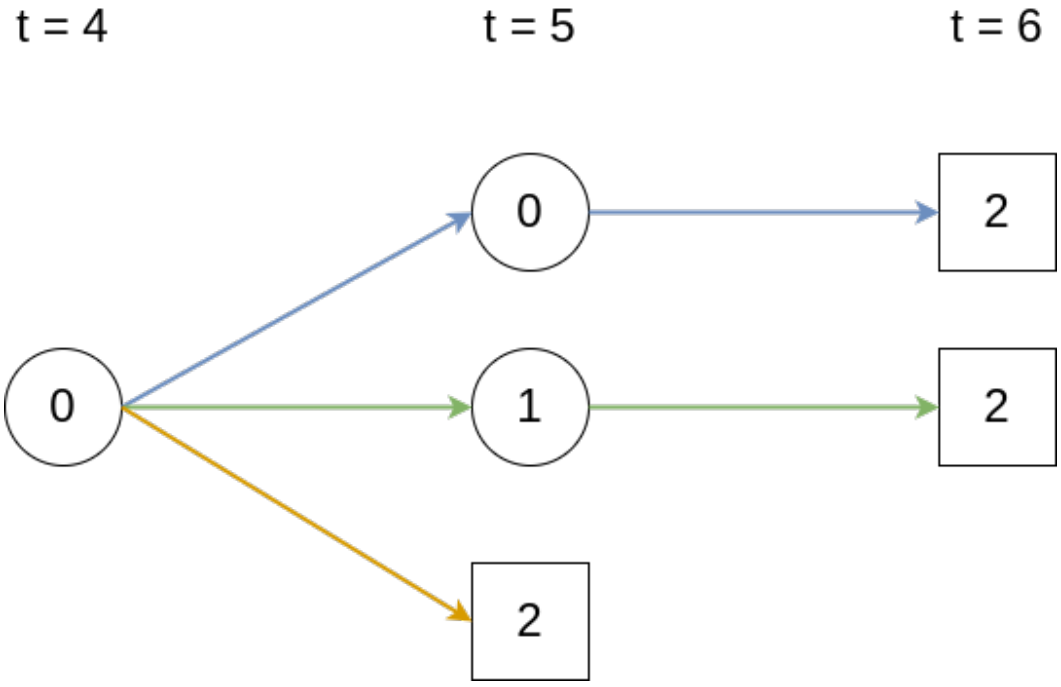


Figure 5.6: Possible paths to transition from state 0 to state 2 between time points 4 and 6

Thus, $P_{02}(4,6) = p_{00}(5)p_{02}(6) + p_{01}(6)p_{12}(6) + p_{02}(5)$, where $p_{\ell e}(\tau) = P(E(\tau) = e | E(\tau - 1) = \ell)$ are the probabilities for transitions $\ell \to e$ between two subsequent discrete time points. Thus, in discrete time, the matrix of transition probabilities can be represented as a finite matrix product

$$\mathbf{P}(\zeta, \tau) = \prod_{j=\zeta+1}^{\tau} \begin{pmatrix} p_{00}(j) & p_{01}(j) & p_{02}(j) \\ 0 & p_{11}(j) & p_{12}(j) \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.18}$$

For the concrete example we thus have

$$\begin{aligned} \mathbf{P}(4,6) &= \begin{pmatrix} P_{00}(4,6) & P_{01}(4,6) & P_{02}(4,6) \\ 0 & P_{11}(4,6) & P_{12}(4,6) \\ 0 & 0 & 1 \end{pmatrix} = \prod_{j=5}^{6} \begin{pmatrix} p_{00}(j) & p_{01}(j) & p_{02}(j) \\ 0 & p_{11}(j) & p_{12}(j) \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} p_{00}(5) & p_{01}(5) & p_{02}(5) \\ 0 & p_{11}(5) & p_{12}(5) \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_{00}(6) & p_{01}(6) & p_{02}(6) \\ 0 & p_{11}(6) & p_{12}(6) \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} p_{00}(5)p_{00}(6) & p_{00}(5)p_{01}(6) + p_{01}(5)p_{11}(6) & p_{00}(5)p_{02}(6) + p_{01}(5)p_{12}(6) + p_{02}(5) \\ 0 & p_{11}(5)p_{11}(6) & p_{11}(5)p_{12}(6) + p_{12}(5) \\ 0 & 0 & 1 \end{pmatrix}, \end{aligned}$$

where the quantity of interest, $P_{02}(4,6)$ is given in the top right corner, but other transition

probabilities are also readily available. For example, the probability to transition from state 1 to 2 between time points $\zeta = 4$ and $\tau = 6$ is given as $P_{12}(4,6) = p_{11}(5)p_{12}(6) + p_{12}(5)$.

Returning to the continuous time setting where transitions can occur at any time point, ideas from the discrete time setting still hold. Imagine dividing the interval $(\zeta, \tau]$ into J intervals such that $\zeta = t_0 < t_1 < \cdots < t_j < \cdots t_J = \tau$, assuming that no events occur between time points $t_j \in \mathbb{R}_+, j = 1, \ldots, J$. Then 5.18 still holds when replacing $p_{\ell e}(j)$ with $p_{\ell e}(t_j)$.

Increasing the number of intervals to infinity or equivalently, reducing the interval size to infinitesimally small intervals du where only one transition can be observed, leads to a product integral over the instantaneous transition probabilities $p_{\ell e}(u)$ (5.16), such that

$$\mathbf{P}(\zeta, \tau) = \lim_{\mathrm{d}u \to 0} \prod_{u \in (\zeta, \tau]} \begin{pmatrix} p_{00}(u) & \cdots & p_{0q}(u) \\ \vdots & \ddots & \vdots \\ p_{q0}(u) & \cdots & p_{qq}(u) \end{pmatrix} \tag{5.19}$$

5.3.3 Instantaneous transition probabilities and hazards

In context of survival analysis we want to express the transition matrix (5.19) and thus instantaneous transition probabilities $p_{\ell e}(u)$ in terms of (cumulative) hazards. To do so, we use, somewhat informally, the following relationships

- From equations 5.12 and 5.13 we can equate the instantaneous transition probabilities to increments of the cumulative hazard (that is the increase in the cumulative hazard within a (fixed, infinitesimally) small interval $\mathrm{d}u$): $dH_{\ell e}(u) = h_{\ell e}(u)du = P\big(E(u + du) = e | E(u-) = \ell\big) = p_{\ell e}(u)$,
- because of relationship 5.15, diagonal elements (transitions into the same state) are set to $dH_{\ell\ell}(u) := -\sum_{e \neq \ell} dH_{\ell e}(u)$ such that $1 + dH_{\ell\ell}(u) = 1 - \sum_{e \neq \ell} dH_{\ell e}(u) = 1 - \sum_{e \neq \ell} p_{\ell e}(u) = p_{\ell\ell}(u)$.

Consequently, 5.19 can be rewritten as

$$\begin{aligned} \mathbf{P}(\zeta, \tau) &= \lim_{\mathrm{d}u \to 0} \prod_{u \in (\zeta, \tau]} \begin{pmatrix} p_{00}(u) = 1 + dH_{00}(u) & \cdots & p_{0q}(u) = dH_{0q}(u) \\ \vdots & \ddots & \vdots \\ p_{q0}(u) = dH_{q0}(u) & \cdots & p_{qq}(u) = 1 + dH_{qq}(u) \end{pmatrix} \\ &= \lim_{\mathrm{d}u \to 0} \prod_{u \in (\zeta, \tau]} (\mathbf{I} + d\mathbf{H}(u)), \end{aligned}$$

where $\mathbf{I}$ is a $(q+1) \times (q+1)$ identity matrix and $d\mathbf{H}(u)$ is the matrix of increments of the cumulative hazard within infinitesimally small intervals. Thus, similar to the competing risks setting, relationship 5.19 implies that knowledge of the transition specific (cumulative) hazards is sufficient to fully specify the multi-state process.

As analytical solutions of 5.19 only exist for specific models, in practice, the transition probabilities are often once again approximated numerically via a finite matrix product on a discrete time grid $\zeta = t_0 < t_1 < \cdots < t_{J-1} < t_J = \tau$

$$\mathbf{P}(\zeta, \tau) \approx \prod_{j=1}^{J} (\mathbf{I} + \triangle \mathbf{H}_{\ell e}(t_j)), \tag{5.20}$$

where $\triangle H_{\ell e}(t_j) = H_{\ell e}(t_j) - H_{\ell e}(t_{j-1})$ is the increment of the cumulative hazards between to consecutive time points.

5.3.4 Non-parametric estimation of transition probabilities

From 5.20, it follows that transition probabilities can be estimated by first computing the transition-specific cumulative hazards, $H_{\ell e}(\tau)$. Similarly to the competing risks setting (Section 5.2.2), we can first define the transition specific hazards

$$h^d_{\ell e}(t_{(k)}) := \frac{d_{\ell e, t_{(k)}}}{n_{\ell; t_{(k)}}},$$

where

- $d_{\ell e, t_{(k)}}$ is the number of subjects that made the transition $\ell \to e$ at time $t_{(k)}$ and
- $n_{\ell; t_{(k)}}$ is the number of subjects in state ℓ immediately before $t_{(k)}$.

The cumulative transition-specific hazards follow as

$$H_{NA,\ell e}(\tau) = \sum_{k: t_{(k)} \leq \tau} h^d_{\ell e}(t_{(k)}) = \sum_{k: t_{(k)} \leq \tau} \frac{d_{\ell e, t_{(k)}}}{n_{\ell; t_{(k)}}},$$

and transition probabilities are obtained via 5.20 as

$$\mathbf{P}(\zeta, \tau) = \prod_{j=1}^{J} \big(\mathbf{I} + \triangle \mathbf{H}_{NA,\ell e}(t_{(j)})\big). \tag{5.21}$$

5.3.5 Application to liver cirrhosis patients

For illustration, consider again the `prothr` data set (Table 5.2), with possible transitions summarized in Figure 5.4. In contrast to the illness-death model in Figure 5.5, back transitions are possible and some subjects already start in the "abnormal" state at the beginning of the study.

Figure 5.7 shows the transition probabilities for the four possible transitions over time for subjects who received treatment and placebo, respectively. In this example back-transitions are possible, therefore, in contrast to the cumulative incidence functions in the competing risks setting, transition probabilities (to transient states) are not monotonously increasing over time. While the probabilities to transition from normal or abnormal state to death ($0 \to 2$, $1 \to 2$) increase over time for both groups, probabilities for transitions between the transient states (normal to abnormal and vice versa) increase in the beginning but eventually decreases over time. Overall, prednisone doesn't appear to have a strong protective effect. Although there appears to be a reduction in $0 \to 2$ transitions and an increase in $1 \to 0$ transitions between time points 1000 and 3000, this effect doesn't seem to persist until the end of the follow up.

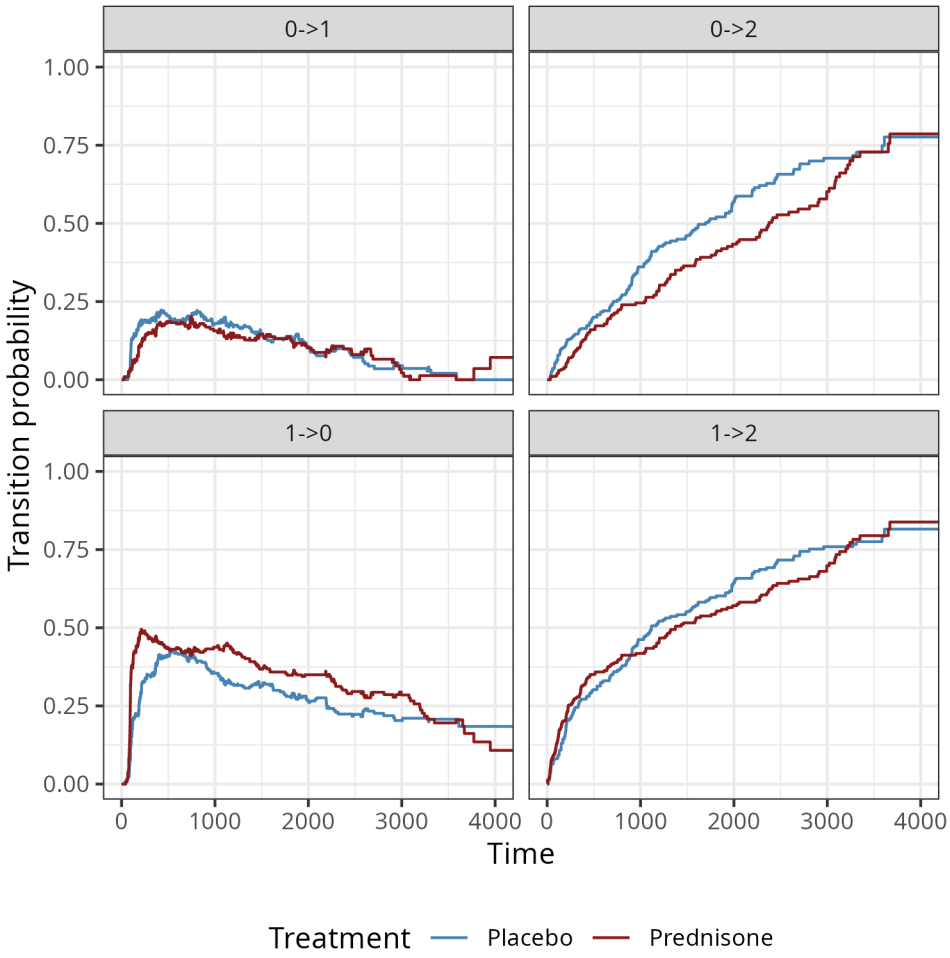


Figure 5.7: Estimated transition probabilities for the different transitions of the prothrombin data example .

6

Survival Task

This chapter introduces the different survival problems and formalises them as survival tasks. There are four prediction types in survival analysis: relative risks - predicting the risk of an event, survival times - predicting the time until an event happens, prognostic index - predicting a linear predictor to assess outcomes based on risk factors, and survival distributions - predicting the probability of an event taking place over time. These reduce to three formal survival tasks: deterministic (survival time), ranking (risks and prognostic index), and probabilistic (distribution). This separation of tasks helps create a taxonomy of survival models and losses that is used throughout the book.

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

This final section of this part of the book, brings everything together and considers the different prediction types that might be of interest in the survival analysis context and introduces the notion of a survival task more formally. Throughout this chapter let $\mathcal{X} \subseteq \mathbb{R}^{n \times p}$ be the feature space.

A general survival prediction problem is one in which (Section 3.1):

- a survival dataset, $\mathcal{D}$, is split for training, $\mathcal{D}_{train}$, and testing, $\mathcal{D}_{test}$;
- a survival model is fit on $\mathcal{D}_{train}$; and
- the model predicts a representation of the unknown true survival time, Y, given $\mathcal{D}_{test}$.

The process of fitting is model-dependent, and can range from non-parametric methods and maximum likelihood estimation of model parameters to machine learning approaches. The model fitting process is discussed on a high-level in Section 3.1 and concrete algorithms are discussed in Part III of this book. The different survival problems are separated by *prediction types* or *prediction problems*, which can also be thought of as predictions of different representations of Y. We consider 4 commonly used prediction types:

1. The *relative risk* of an individual experiencing an event: A single continuous ranking.
2. The *time until an event* occurs: A single continuous value.
3. The *prognostic index* for a model: A single continuous value.
4. The *survival distribution*: A probability distribution.

The first three of these are referred to as *deterministic* as they predict a single value whereas the fourth is *probabilistic* and returns a full survival distribution. Definitions of these are

expanded on below but first note that survival predictions differ from other fields in two respects:

- The observed data used for model training (observed times T, status indicator Δ) is different from the outcome of interest (event times Y). This differs from, say, standard regression in which the same object (a single continuous variable) is used for fitting and predicting.
- With the exception of the time-to-event prediction, all other prediction types do not predict the expectation $\mathbb{E}(Y)$, which is often of interest, but some other (related) quantity.

Survival prediction problems must be clearly separated as they are inherently incompatible. For example, it is not meaningful to compare a relative risk prediction from one model to a survival distribution prediction of another. Whilst these prediction types are separated above, they can be viewed as special cases of each other. Both (1.) and (2.) may be viewed as variants of (3.); and (1.), (2.), and (3.) can all be derived from (4.); this is elaborated on below and discussed fully in Chapter 16.

6.1 Predicting Risks

This is a common survival problem and is defined as predicting a continuous rank for an individual's relative risk of experiencing the event. For example, given three subjects, $\{i, j, k\}$, a relative risk prediction may predict the risk of event as $\{0.1, 0.5, 10\}$ respectively. From these predictions, the following types of conclusions can be drawn:

- Conclusions comparing subjects. For example, i is at the least risk; the risk of j is only slightly higher than that of i but the risk of k is considerably higher than j; the corresponding ranks for i, j, k, are $1, 2, 3$;
- Conclusions comparing risk groups. For example, thresholding the risks at 1.0 means that i and j are in a low-risk group whilst k is in a high-risk group.

Whilst many important conclusions can be drawn from these predictions, the values themselves have no meaning when not compared to other individuals. Interpretation of these rankings depends on the model class (for example, PH and AFT models have opposite interpretations, Chapter 11) and its parametrization or implementation in specific software. For some higher ranking implies higher risk whereas others may assume that higher ranking implies lower risk. In this book, a higher ranking will always imply a higher risk of event (as in the example above).

Predicting rankings is the primary form of the *survival ranking task*, defined by predicting a continuous value, $g : \mathcal{X} \to \mathcal{R}$ where $\mathcal{R} \subseteq \mathbb{R}$.

6.2 Predicting Survival Times

Predicting a time to event is the problem of predicting the expectation $\hat{y} = \mathbb{E}(Y|\mathbf{x})$. A time-to-event prediction is a special case of a ranking prediction as an individual with a longer survival time will have a lower overall risk: if $\hat{y}_i, \hat{y}_j$ and $\hat{r}_i, \hat{r}_j$ are survival time and ranking predictions for subjects i and j respectively, then $\hat{y}_i > \hat{y}_j \Rightarrow \hat{r}_i < \hat{r}_j$.

For practical purposes, the expected time-to-event would be the ideal prediction type as it is easy to interpret and communicate. However, this type of prediction is rare for multiple reasons. For one, an usuall loss based on $f(y_i)$ or some difference of true and predicted value, $y_i - \hat{y}_i$ is not, suitable for censored data, as y_i is not observed for some observations, so direct estimation/prediction of $\hat{y}_i = E(Y|\mathbf{x}_i)$ requires some imputation of censored observations (and evaluation on new data can also only be done on observed or imputed values).

Alternatively, one could derive the expectation by predicting the survival distribution while taking into account the censoring and obtain a time-to-event prediction by calculating expected values, but this brings its own challenges and pitfalls (see "Survival Distribution" below for details).

Predicting survival times is the *deterministic survival task*, defined by predicting a continuous value in the positive Reals and is specified by $g : \mathcal{X} \to \mathbb{R}_{\geq 0}$. See Section 21.1 for practical discussion around predicting in $\mathbb{R}_{\geq 0}$ vs. $\mathbb{R}_{>0}$ and continuous vs discrete time representations. Formally, whilst this is a special case of the ranking task with $\mathcal{R} \subseteq \mathbb{R}_{\geq 0}$, the distinction is important as a 'deterministic' prediction specifically refers to forecasting a single determined outcome with a meaningful interpretation, whereas the 'ranking' task is not a deterministic forecast of an event.

6.3 Prognostic Index Predictions

In medical terminology (which is often used in survival analysis), a prognostic index is a tool that predicts outcomes based on risk factors. Given covariates, $\mathbf{X} \in \mathbb{R}^{n \times p}$, and coefficients, $\boldsymbol{\beta} \in \mathbb{R}^p$, the *linear predictor* is defined as $\boldsymbol{\eta} := \mathbf{X}\boldsymbol{\beta}$. Applying some function g, which could simply be the identity function $g(x) = x$, yields a *prognostic index*, $g(\boldsymbol{\eta})$. A prognostic index can serve several purposes, including:

1. Scaling or normalization – simple functions to scale the linear predictor can better support interpretation and visualisation;
2. Capturing non-linear effects – for example the Cox PH model (Chapter 11) applies the transformation $g(\boldsymbol{\eta}) = \exp(\boldsymbol{\eta})$ to capture more complex relationships between features and outcomes;
3. Aiding in interpretability – in some cases this could simply be $g(\boldsymbol{\eta}) = -\boldsymbol{\eta}$ to ensure the 'higher value implies higher risk' interpretation.

A prognostic index is a special case of the survival ranking task, assuming that there is a one-to-one mapping between the prediction and expected survival times. Once again, it is assumed in this book that a higher value for the prognostic index implies higher risk of event.

6.4 Predicting Distributions

Predicting a survival distribution refers specifically to predicting the distribution of a subject's survival time, i.e., modelling the distribution of the event occurring over $\mathbb{R}_{\geq 0}$. Therefore, this is seen as the probabilistic analogue to the deterministic time-to-event prediction.

Distributional prediction can, in theory, target any of the quantities introduced in Section 4.1, but predicting $S(t)$ and/or $h(t)$ is most common. Hazard based approaches are particularly relevant for non- and semi-parametric estimation of the distribution, where no (or few) assumptions are made about the underlying distribution of event times.

As mentioned above, all prediction types can theoretically be derived from a survival distribution prediction. For example, a time-to-event prediction can be obtained via $E(Y|\mathbf{x}) = \int_0^\infty \hat{S}(t)$. However, for non-parametric methods the estimated cdf is often improper in the presense of censoring and thus integration requires extrapolation of the cdf (R. Sonabend, Bender, and Vollmer 2022). For parametric models, the distribution of event times is fully specified once the paramers of the assumed distribution have been estimated, however, if the parameters were estimated based on only a small subset of the possible domain of Y, this essentially still constitutes extrapolation and will in most cases yield implausible predictions. A popular alternative is therefore to estimate the *restricted mean survival time* (RMST; K. Han and Jung (2022); Andersen, Hansen, and Klein (2004)).

Predicting survival distributions is a type of *probabilistic survival task*, defined by predicting a conditional distribution over the positive Reals, $g : \mathcal{X} \to \mathcal{S}$ where $\mathcal{S} \subseteq \text{Distr}(\mathbb{R}_{\geq 0})$ is a convex set of distributions on $\mathbb{R}_{\geq 0}$.

6.5 Conclusion

Key takeaways

- There are three survival tasks: probabilistic, deterministic, and ranking;
- Probabilistic tasks predict a survival distribution, which is the probability of an event occurring over time;
- Deterministic tasks predict a survival time, which is a useful value but hard to estimate and evaluate in practice;
- Ranking tasks predict ranks that can be compared within cohorts to identify relative risks. Predicting a prognostic index is a special case of a ranking prediction.

Further reading

-

Part II

Evaluation

7

Discrimination

TODO (150-200 WORDS)

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

This chapter discusses 'discrimination measures', which evaluate how well models separate observations into different risk groups. A model is said to have good discrimination if it correctly predicts that one observation is at higher risk of the event of interest than another, where the prediction is 'correct' if the observation predicted to be at higher risk does indeed experience the event sooner.

In the survival setting, the 'risk' is taken to be the continuous ranking prediction. All discrimination measures are ranking measures, which means that the exact predicted value is irrelevant, only its relative ordering is required. For example given predictions $\{100, 2, 299.3\}$, only their rankings, $\{2, 1, 3\}$, are used by measures of discrimination.

This chapter begins with time-independent measures (Section 7.1), which measure concordance between pairs of observations at a single observed time point. The next section focuses on time-dependent measures (Section 7.2), which are primarily AUC-type measures that evaluate discrimination over all possible unique time-points and integrate the results for a single metric.

7.1 Time-Independent Measures

The simplest form of discrimination measures are concordance indices, which, in general, measure the proportion of cases in which the model correctly ranks a pair of observations according to their risk. These measures may be best understood in terms of two key definitions: 'comparable', and 'concordant'.

Definition 7.1 (Concordance)**.** Let (i, j) be a pair of observations with outcomes $\{(t_i, \delta_i), (t_j, \delta_j)\}$ and let $r_i, r_j \in \mathbb{R}$ be their respective risk predictions. Then (i, j) are called (F. E. J. Harrell et al. 1984; F. E. Harrell, Califf, and Pryor 1982):

- *Comparable* if $t_i < t_j$ and $\delta_i = 1$;
- *Concordant* if $r_i > r_j$.

Note that this book defines risk rankings such that a higher value implies higher risk of event and thus lower expected survival time (Chapter 6), hence a pair is concordant if $\mathbb{I}(t_i < t_j, r_i > r_j)$. Other sources may instead assume that higher values imply lower risk of event and hence a pair would be concordant if $\mathbb{I}(t_i < t_j, r_i < r_j)$.

Concordance measures then estimate the probability of a pair being concordant, given that they are comparable:

$$P(r_i > r_j | t_i < t_j \cap \delta_i) \tag{7.1}$$

These measures are referred to as time *independent* when r_i, r_j is not a function of time as once the observations are organized into comparable pairs, the observed survival times can be ignored. The time-dependent case is covered in Section 7.2.1.

While various definitions of a 'Concordance index' (C-index) exist (discussed in the next section), they all represent a weighted proportion of the number of concordant pairs over the number of comparable pairs. As such, a C-index value will always be between $[0, 1]$ with 1 indicating perfect separation, 0.5 indicating no separation, and 0 being separation in the 'wrong direction', i.e. all high risk observations being ranked lower than all low risk observations.

Concordance measures may either be reported as a value in $[0, 1]$, a percentage, or as 'discriminatory power', which refers to the percentage improvement of a model's discrimination above the baseline value of 0.5. For example, if a model has a concordance of 0.8 then its discriminatory power is $(0.8 - 0.5)/0.5 = 60\%$. This representation of discrimination provides more information by encoding the model's improvement over some baseline although is often confused with reporting concordance as a percentage (e.g. reporting a concordance of 0.8 as 80%). In theory this representation could result in a negative value, however this would indicate that $C < 0.5$, which would indicate serious problems with the model that should be addressed before proceeding with further analysis. Representing measures as a percentage over a baseline is a common method to improve measure interpretability and closely relates to the ERV representation of scoring rules (Section 9.4).

7.1.1 Concordance Indices

Common concordance indices in survival analysis can be expressed as a general measure:

Let $\hat{\boldsymbol{r}} = (\hat{r}_1\ \hat{r}_2 \cdots \hat{r}_n)^\top$ be predicted risks, $(\mathbf{t}, \boldsymbol{\delta}) = ((t_1, \delta_1)\ (t_2, \delta_2) \cdots (t_n, \delta_n))^\top$ be observed outcomes, let W be some weighting function, and let τ be a cut-off time. Then, the time-independent ('ind') *survival concordance index* is defined by,

$$C_{ind}(\hat{\mathbf{r}}, \mathbf{t}, \boldsymbol{\delta} | \tau) = \frac{\sum_{i \neq j} W(t_i) \mathbb{I}(t_i < t_j, \hat{r}_i > \hat{r}_j, t_i < \tau) \delta_i}{\sum_{i \neq j} W(t_i) \mathbb{I}(t_i < t_j, t_i < \tau) \delta_i} \tag{7.2}$$

The choice of W specifies a particular variation of the c-index (see below). The use of the cut-off τ mitigates against decreased sample size (and therefore high variance) over time due to the removal of censored observations (see Figure 7.1). For τ to be comparable across datasets, a common choice would be to set τ as the time at which 80%, or perhaps 90% of the data have been censored or experienced the event.

There are multiple methods for dealing with tied predictions and times. Strictly, tied times are incomparable given the definition of 'comparable' given above and hence are usually

ignored in the numerator. On the other hand, ties in the prediction are more problematic but a common method is to set a value of 0.5 for observations when $r_i = r_j$ (Therneau and Atkinson 2024). Specific concordance indices can be constructed by assigning a weighting scheme for W which generally depends on the Kaplan-Meier estimate of the survival function of the censoring distribution fit on training data, $\hat{G}_{KM}$, or the Kaplan-Meier estimate for the survival function of the survival distribution fit on training data, $\hat{S}_{KM}$, or both. Measures that use $\hat{G}_{KM}$ are referred to as Inverse Probability of Censoring Weighted (IPCW) measures as the estimated censoring distribution is utilised to weight the measure in order to compensate for removed censored observations. This is visualised in Figure 7.1 where $\hat{G}_{KM}$, $\hat{G}^{-2}_{KM}$, and $\hat{S}_{KM}$ are computed based on the `whas` dataset (Hosmer Jr, Lemeshow, and May 2011).

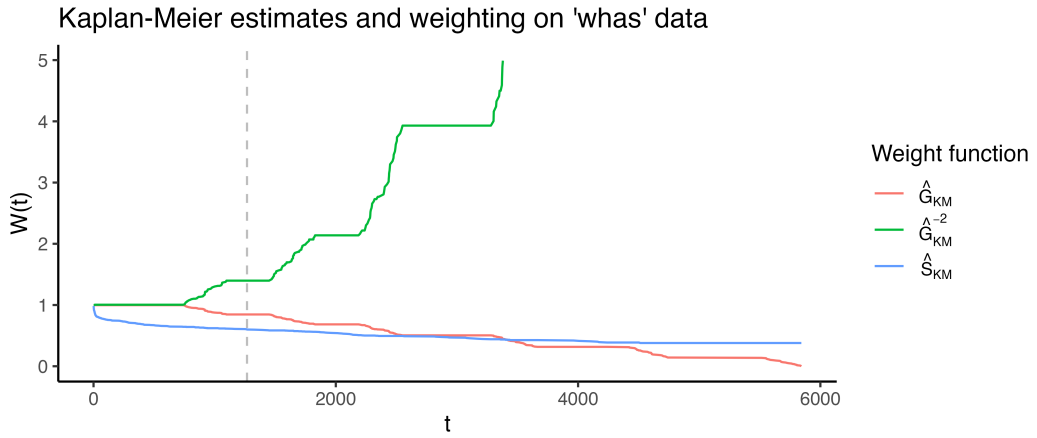


Figure 7.1: Weighting functions obtained on the `whas` dataset. x-axis is follow-up time. y-axis is outputs from one of three weighting functions: $\hat{G}_{KM}$, survival function based on the censoring distribution of the `whas` dataset (red), and $\hat{G}^{-2}_{KM}$ (green), $\hat{S}_{KM}$, marginal survival function based on original `whas` dataset (blue), . The vertical gray line at $t = \tau = 1267$ represents the point at which $\hat{G}(t) < 0.6$.

The following are a few of the weights that have been proposed for the concordance index:

- $W(t_i) = 1$: Harrell's concordance index, C_H (F. E. J. Harrell et al. 1984; F. E. Harrell, Califf, and Pryor 1982), which is widely accepted to be the most common survival measure and imposes no weighting on the definition of concordance. The original measure given by Harrell has no cut-off, $\tau = \infty$, however applying a cut-off is now more widely accepted in practice.
- $W(t_i) = [\hat{G}_{KM}(t_i)]^{-2}$: Uno's C, C_U (Uno et al. 2011).
- $W(t_i) = \hat{S}_{KM}(t_i)$ (Therneau and Atkinson 2024)
- $W(t_i) = \hat{S}_{KM}(t_i)/\hat{G}_{KM}(t_i)$ (Schemper, Wakounig, and Heinze 2009)

All methods assume that censoring is conditionally-independent of the event given the features (Section 4.4), otherwise weighting by $\hat{S}_{KM}$ or $\hat{G}_{KM}$ would not be applicable. It is assumed here that $\hat{S}_{KM}$ and $\hat{G}_{KM}$ are estimated on the training data and not the testing data (though the latter may be seen in some implementations, e.g. Therneau (2015)).

7.1.2 Choosing a C-index

With multiple choices of weighting available, choosing a specific measure might seem daunting. Matters are only made worse by debate in the literature, reflecting uncertainty in measure choice and interpretation. In practice, when a suitable cut-of τ is chosen, all these weightings perform very similarly (Rahman et al. 2017; Schmid and Potapov 2012). For example, Table 7.1 uses the `whas` data again to compare Harrell's C with measures that include IPCW weighting, when no cutoff is applied (top row) and when a cutoff is applied when $\hat{G}(t) = 0.6$ (grey line in Figure 7.1). The results are almost identical when the cutoff is applied but still not massively different without the cutoff.

Table 7.1: Comparing C-index measures (calculated on the `whas` dataset using a Cox model with three-fold cross-validation) with no cut-off (top) and a cut-off when $\hat{G}(t) = 0.6$ (bottom). First column is Harrell's C, second is the weighting $1/\hat{G}(t)$, third is Uno's C.

	$W = 1$	$W = G^{-1}$	$W = G^{-2}$
$\tau = \infty$	0.74	0.73	0.71
$\tau = 1267$	0.76	0.75	0.75

On the other hand, if a poor choice is selected for τ (cutting off too late) then IPCW measures can be highly unstable (Rahman et al. 2017), for example the variance of Uno's C drastically increases with increased censoring (Schmid and Potapov 2012).

In practice, all C-index metrics provide an intuitive measure of discrimination and as such the choice of C-index is less important than the transparency in reporting. 'C-hacking' (R. Sonabend, Bender, and Vollmer 2022) is the deliberate, unethical procedure of calculating multiple C-indices and to selectively report one or more results to promote a particular model or result, whilst ignoring any negative findings. For example, calculating Harrell's C and Uno's C but only reporting the measure that shows a particular model of interest is better than another (even if the other metric shows the reverse effect). To avoid 'C-hacking':

i) the choice of C-index should be made before experiments have begun and the choice of C-index should be clearly reported;
ii) when ranking predictions are composed (Chapter 16) from distribution predictions, the composition method should be chosen and clearly described before experiments have begun.

As the C-index is highly dependent on censoring within a dataset, C-index values between experiments are not directly comparable, instead comparisons are limited to comparing model rankings, for example conclusions such as "model A outperformed model B with respect to Harrell's C in this experiment".

7.2 Time-Dependent Measures

In the time-dependent case, where the metrics are computed based on specific survival times, the majority of measures are based on the Area Under the Curve, with one exception which is a simpler concordance index.

7.2.1 Concordance Indices

In contrast to the measures described above, Antolini's C (Antolini, Boracchi, and Biganzoli 2005) provides a time-dependent ('dep') formula for the concordance index. The definition of 'comparable' is the same for Antolini's C, however, concordance is now determined using the individual predicted survival probabilities calculated at the smaller event time in the pair:

$$P(\hat{S}_i(t_i) < \hat{S}_j(t_i) | t_i < t_j \cap \delta_i)$$

Note that observations are concordant when $\hat{S}_i(t_i) < \hat{S}_j(t_i)$ as at the time t_i, observation i has experienced the event and observation j has not, hence the expected survival probability for $\hat{S}_i(t_i)$ should be as close to 0 as possible (noting inherent randomness may prevent the perfect $\hat{S}_i(t_i) = 0$ prediction) but otherwise should be less than $\hat{S}_j(t_i)$ as j is still 'alive'. Once again this probability is estimated with a metric that could include a cut-off and different weighting schemes (though this is not included in Antolini's original definition):

$$C_{dep}(\hat{\mathbf{S}}, \mathbf{t}, \boldsymbol{\delta} | \tau) = \frac{\sum_{i \neq j} W(t_i) \mathbb{I}(t_i < t_j, \hat{S}_i(t_i) < \hat{S}_j(t_i), t_i < \tau) \delta_i}{\sum_{i \neq j} W(t_i) \mathbb{I}(t_i < t_j, t_i < \tau) \delta_i}$$

where $\hat{\boldsymbol{S}} = (\hat{S}_1 \ \hat{S}_2 \cdots \hat{S}_n)^\top$.

Antolini's C provides an intuitive method to evaluate the discrimination of a model based on distribution predictions without depending on compositions to ranking predictions.

7.2.2 Area Under the Curve

AUC, or AUROC, measures calculate the Area Under the Receiver Operating Characteristic (ROC) Curve, which is a plot of the *sensitivity* (or true positive rate (TPR)) against $1 -$ *specificity* (or true negative rate (TNR)) at varying thresholds (described below) for the predicted probability (or risk) of event. Figure 7.2 visualises ROC curves for two classification models. The blue line is a featureless baseline that has no discrimination. The red line is a decision tree with better discrimination as it comes closer to the top-left corner.

In a classification setting with no censoring, the AUC has the same interpretation as Harrell's C (Uno et al. 2011). AUC measures for survival analysis were developed to provide a time-dependent measure of discriminatory ability (Patrick J. Heagerty, Lumley, and Pepe 2000). In a survival setting it can reasonably be expected for a model to perform differently over time and therefore time-dependent measures are advantageous. Computation of AUC estimators is complex and as such there are limited accessible metrics available off-shelf.

The AUC, TPR, and TNR are derived from the *confusion matrix* in a binary classification setting. Let $y_i, \hat{y}_i \in \{0, 1\}$ be the true and predicted binary outcomes respectively for some observation i. The confusion matrix is then given by:

	$y_i = 1$	$y_i = 0$
$\hat{y}_i = 1$	TP	FP
$\hat{y}_i = 0$	FN	TN

where $TN := \sum_i \mathbb{I}(y_i = 0, \hat{y}_i = 0)$ is the number of true negatives, $TP := \sum_i \mathbb{I}(y_i = 1, \hat{y}_i = 1)$ is the number true positives, $FP := \sum_i \mathbb{I}(y_i = 0, \hat{y}_i = 1)$ is the number of false positives, and $FN := \sum_i \mathbb{I}(y_i = 1, \hat{y}_i = 0)$ is the number of false negatives. From these are derived

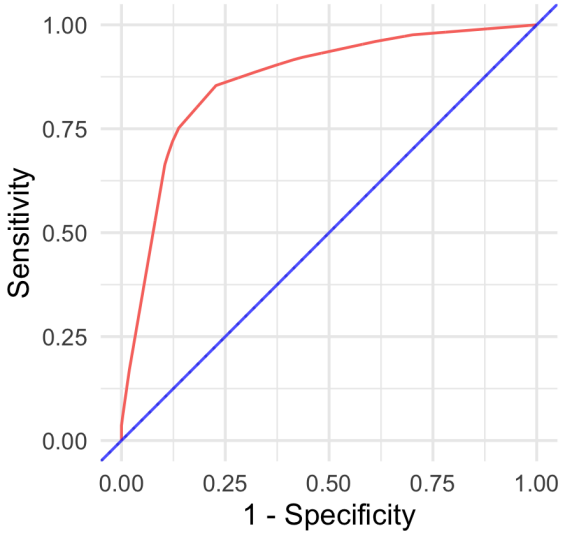


Figure 7.2: ROC Curves for a classification example. Red is a decision tree with good discrimination as it 'hugs' the top-left corner. Blue is a featureless baseline with no discrimination as it sits on $y = x$.

$$TPR := \frac{TP}{TP + FN}$$
$$TNR := \frac{TN}{TN + FP}$$

In classification, a probabilistic prediction of an event can be *thresholded* to obtain a deterministic prediction. For a predicted $\hat{p} := \hat{P}(b = 1)$, and threshold α, the thresholded binary prediction is $\hat{b} := \mathbb{I}(\hat{p} > \alpha)$. This is achieved in survival analysis by thresholding the linear predictor at a given time for different values of the threshold and different values of the time. All measures of TPR, TNR and AUC are in the range $[0, 1]$ with larger values preferred.

Weighting the linear predictor was proposed by Uno *et al.* (2007) (Uno et al. 2007) and provides a method for estimating TPR and TNR via

$$TPR_U(\hat{\boldsymbol{\eta}}, \mathbf{t}, \boldsymbol{\delta}|\tau, \alpha) = \frac{\sum_{i=1}^{n} \delta_i \mathbb{I}(k(\hat{\eta}_i) > \alpha, t_i \leq \tau)[\hat{G}_{KM}(t_i)]^{-1}}{\sum_{i=1}^{n} \delta_i \mathbb{I}(t_i \leq \tau)[\hat{G}_{KM}(t_i)]^{-1}}$$

and

$$TNR_U(\hat{\boldsymbol{\eta}}, \mathbf{t}|\tau, \alpha) \mapsto \frac{\sum_{i=1}^{n} \mathbb{I}(k(\hat{\eta}_i) \leq \alpha, t_i > \tau)}{\sum_{i=1}^{n} \mathbb{I}(t_i > \tau)}$$

where $\hat{\boldsymbol{\eta}} = (\hat{\eta}_1 \ \hat{\eta}_2 \cdots \hat{\eta}_n)^\top$ is a vector of estimated linear predictors, τ is the time at which to evaluate the measure, α is a cut-off for the linear predictor, and k is a known, strictly

increasing, differentiable function. k is chosen depending on the model choice, for example if the fitted model is PH then $k(x) = 1 - \exp(-\exp(x))$ (Uno et al. 2007). Similarities can be drawn between these equations and Uno's concordance index, in particular the use of IPCW. Censoring is again assumed to be at least random once conditioned on features. Plotting TPR_U against $1 - TNR_U$ for varying values of α provides the ROC.

The second method, which appears to be more prominent in the literature, is derived from Heagerty and Zheng (2005) (Patrick J. Heagerty and Zheng 2005). They define four distinct classes, in which observations are split into controls and cases.

An observation is a *case* at a given time-point if they are dead, otherwise they are a *control*. These definitions imply that all observations begin as controls and (hypothetically) become cases over time. Cases are then split into *incident* or *cumulative* and controls are split into *static* or *dynamic*. The choice between modelling static or dynamic controls is dependent on the question of interest. Modelling static controls implies that a 'subject does not change disease status' (Patrick J. Heagerty and Zheng 2005), and few methods have been developed for this setting (Kamarudin, Cox, and Kolamunnage-Dona 2017), as such the focus here is on *dynamic* controls. The incident/cumulative cases choice is discussed in more detail below.[1]

The TNR for dynamic cases is defined as

$$TNR_D(\hat{\mathbf{r}}, N|\alpha, \tau) = P(\hat{r}_i \leq \alpha | t_i > \tau)$$

where $\hat{\boldsymbol{r}} = (\hat{r}_1\ \hat{r}_2 \cdots \hat{r}_n)^\top$ is some prediction, which is usually deterministic, often the linear predictor $\hat{\eta}$, however could be a predicted survival probability – though in the latter case there is not necessarily an advantage over Antolini's C. Cumulative and incident versions of the TPR are respectively defined by

$$TPR_C(\hat{\mathbf{r}}, N|\alpha, \tau) = P(\hat{r}_i > \alpha | t_i \leq \tau)$$

and

$$TPR_I(\hat{\mathbf{r}}, N|\alpha, \tau) = P(\hat{r}_i > \alpha | t_i = \tau)$$

A 'true negative' occurs when the risk is below a certain threshold and the event has yet to occur, in contrast a 'true positive' occurs when the event has already occurred and therefore the risk is expected to be above the threshold. Estimation of these quantities depends on non-parametric estimators, such as the Kaplan-Meier and Akritas estimator (**?@sec-surv-models-uncond**), further details are not provided here.

The choice between the incident/dynamic (I/D) and cumulative/dynamic (C/D) measures primarily relates to the use-case. The C/D measures are preferred if a specific time-point is of interest (Patrick J. Heagerty and Zheng 2005) and is implemented in several applications for this purpose (Kamarudin, Cox, and Kolamunnage-Dona 2017). The I/D measures are preferred when the true survival time is known and discrimination is desired at the given event time (Patrick J. Heagerty and Zheng 2005).

Defining a time-specific AUC is now possible with

[1] All measures discussed in this section evaluate model discrimination from 'markers', which may be a *predictive* marker (model predictions) or a *prognostic* marker (a single covariate). This section always defines a marker as a ranking prediction, which is valid for all measures discussed here with the exception of one given at the end.

$$AUC(\hat{\mathbf{r}}, N|\tau) = \int_0^1 TPR(\hat{\mathbf{r}}, N|1 - TNR^{-1}(p|\tau), \tau)\ dp$$

Finally, integrating over all time-points produces a time-dependent AUC and as usual a cut-off is applied for the upper limit,

$$AUC^*(\hat{\mathbf{r}}, N|\tau^*) = \int_0^{\tau^*} AUC(\hat{\mathbf{r}}, N|\tau) \frac{2\hat{p}_{KM}(\tau)\hat{S}_{KM}(\tau)}{1 - \hat{S}^2_{KM}(\tau^*)}\ d\tau$$

where $\hat{S}_{KM}, \hat{p}_{KM}$ are survival and mass functions estimated with a Kaplan-Meier model on training data.

Since Heagerty and Zheng's paper, other methods for calculating the time-dependent AUC have been devised, including by Chambless and Diao (Chambless and Diao 2006), Song and Zhou (Song and Zhou 2008), and Hung and Chiang (Hung and Chiang 2010). These either stem from the Heagerty and Zheng paper or ignore the case/control distinction and derive the AUC via different estimation methods of TPR and TNR. Blanche *et al.* (2012) (Blanche, Latouche, and Viallon 2012) surveyed these and concluded ''regarding the choice of the retained definition for cases and controls, no clear guidance has really emerged in the literature'', but agree with Heagerty and Zeng on the use of C/D for clinical trials and I/D for 'pure' evaluation of the marker. Blanche *et al.* (2013) (Blanche, Dartigues, and Jacqmin-Gadda 2013) published a survey of C/D AUC measures with an emphasis on non-parametric estimators with marker-dependent censoring, including their own Conditional IPCW (CIPCW) AUC, which is not discussed further here as it cannot be used for evaluating predictions (R. E. B. Sonabend 2021).

Reviews of AUC measures have produced (sometimes markedly) different results (Blanche, Latouche, and Viallon 2012; L. Li, Greene, and Hu 2018; Kamarudin, Cox, and Kolamunnage-Dona 2017) with no clear consensus on how and when these measures should be used. The primary advantage of these measures is to extend discrimination metrics to be time-dependent. However, it is unclear how to interpret a threshold of a linear predictor and moreover if this is even the 'correct' quantity to threshold, especially when survival distribution predictions are the more natural object to evaluate over time.

7.3 Extensions

7.3.1 Competing risks

Discrimination measures are usually extended to the competing risk setting by evaluating cause-specific probabilities individually and then potentially summing or averaging over cause-specific measures (Geloven et al. 2022; C. Lee et al. 2018; Bender et al. 2021; Alberge et al. 2025).

To recap formulae from Chapter 5, given q possible events then the cause-specific hazard for an event is defined as h_e:

$$h_e(\tau) = \lim_{\Delta\tau \to 0} \frac{P(\tau \leq Y \leq \tau + \Delta\tau, E = e \mid Y \geq \tau)}{\Delta\tau},\ e = 1, \ldots, q.$$

where Y is the random variable representing the time-to-event and $E \in \{1, \ldots, k\}$ is the random variable with realizations e, which denotes one of k competing events that can occur at event time Y.

In a single-event setting, a pair of observations, (i, j), with outcomes $\{(t_i, \delta_i), (t_j, \delta_j)\}$, and predicted risk predictions, $r_i, r_j \in \mathbb{R}$, are called comparable if $t_i < t_j$ and $\delta_i = 1$; and concordant if also $r_i > r_j$. In a competing risks setting, for an event of interest e, the observations are (Geloven et al. 2022):

- *Comparable* if $t_i < t_j$ and $\delta_{ie} = 1$; and
- *Concordant* if $r_i > r_j$

Where $\delta_{ie} = 1$ is equivalent to $\mathbb{I}(Y_i \leq C_i \wedge E_i = e)$.

The usual definition of concordance measures then follow given this additional conditioning on the event of interest e. In practice, given that competing risks models often estimate cause-specific hazard functions, it is a variation of Antolini's time-dependent concordance measure that suits the competing risks setting best. Hence for an event e, the measure is given by:

$$C(\hat{\mathbf{h}}_e, \mathbf{t}, \boldsymbol{\delta}|\tau) = \frac{\sum_{i \neq j} W(t_i)\mathbb{I}(t_i < t_j, \hat{h}_{e_i}(t_i) > \hat{h}_{e_j}(t_i), t_i < \tau)\delta_{ie}}{\sum_{i \neq j} W(t_i)\mathbb{I}(t_i < t_j, t_i < \tau)\delta_{ie}}$$

where $\hat{\mathbf{h}}_e = (\hat{h}_{e_1}\ \hat{h}_{e_2} \cdots \hat{h}_{e_n})^\top$ are cause specific hazards for individual observations risk of event e.

7.3.2 Other censoring and truncation types

AUC and concordance indices are designed to rank observations, or at least evaluate how well a model discriminates between two risk groups. This is a substantial challenge when there is interval censoring in the data as the 'true' ranks of observations are unknown. Let i, j be two observations with interval censoring times $(l_i, r_j), (l_j, r_j)$ respectively. Given valid combinations (r > l) then these observations may coincide with one of six combinations:

1. $l_i < r_i < l_j < r_j$
2. $l_i < l_j < r_i < r_j$
3. $l_i < l_j < r_j < r_i$
4. $l_j < r_j < l_i < r_i$
5. $l_j < l_i < r_j < r_i$
6. $l_j < l_i < r_i < r_j$

Of these, only case (1) and (4) can be used in a standard concordance index as the intervals are non-overlapping. For all other cases, conditional probabilities have to be substituted or imputation used to estimate when in an interval the event takes place (Tsouprou 2015; Ying Wu and Cook 2020). After such estimation, interpretation of any metric is difficult, as it's unclear if the original prediction is being evaluated or the guesswork that went into the evaluation measure. In this case it is likely better to use more straightforward methods that do not require these extra steps.

When it comes to time-dependent AUC measures, the TNR and TPR can be extended to accommodate interval-censoring, by updating the equations to only include contributions

from observations when the event is guaranteed to have occurred ($r_i \leq \tau$) or not ($l_i > \tau$) (J. Li and Ma 2011):

$$TNR_D(\hat{\mathbf{r}}, N|\alpha, \tau) = P(\hat{r}_i \leq \alpha | l_i > \tau)$$

and

$$TPR_C(\hat{\mathbf{r}}, N|\alpha, \tau) = P(\hat{r}_i > \alpha | r_i \leq \tau)$$

In the presence of interval censoring, estimation depends on more complex estimators such as the nonparametric maximum likelihood estimator, which can be challenging to fit and splines could be used instead (Yuan Wu et al. 2020). However, splines themselves require modelling and any metric that requires modelling to estimate introduces significant difficulties in its interpretation. As a more simplistic (yet sometimes realistic on average) alternative, one could impute the outcome time as the midpoint between the interval margins, $t_i = (l_i + r_i)/2$ (Jacqmin-Gadda et al. 2016).

As left-censoring is a specialised case of interval-censoring (with $l_i = 0$) the above formulae can be used to handle left-censored data as well.

7.3.3 Truncation

When only right-censoring is present, the objective of concordance measures is to estimate the probability that the ranking of two observations' risk is correctly specified (7.1). The naive method to deal with left-truncation is to simply ignore it and use 7.2 as usual. Doing so yields an estimator, C_N, that converges in probability to

$$P(r_i > r_j | \delta_i, t_i < t_j, t_i \geq t_i^L, t_j \geq t_j^L) \tag{7.3}$$

where t_i^L, t_j^L are the left-truncation times for i and j respectively (which are 0 if the observation is not left-truncated). This is considered a 'naive' estimator as it may introduce bias (Hartman et al. 2022), to see why this is the case consider that the right-hand-side of 7.3 is true in one of two cases (ignoring ties):

1. $t_i < t_j^L < t_j$;
2. $t_j^L < t_i < t_j$.

In the first case, j enters the study after i has experienced the outcome and the observations never overlap in the data at the same time Hartman et al. (2022) demonstrate that evaluating discrimination in the presence of these 'nonoverlapping intervals' creates bias if truncation is dependent on the predictors (for the same reason, methods for estimation need to be adapted as well). Therefore, the data should be further restricted to overlapping intervals, yielding the estimator

$$C_{LT}(\hat{\mathbf{r}}, \mathbf{t}, \boldsymbol{\delta}, \mathbf{t}^L | \tau) = \frac{\sum_{i \neq j} \mathbb{I}(t_i < t_j, \hat{r}_i > \hat{r}_j, t_i < \tau, t_i \geq t_j^L)\delta_i}{\sum_{i \neq j} \mathbb{I}(t_i < t_j, t_i < \tau, t_i \geq t_j^L)\delta_i}$$

In contrast to 7.2, no generic weighting term is included, which is due to the complexity of estimating IPC weights in the context of left-truncation (Therneau and Atkinson 2024).

Interestingly, experiments have shown that naively applying Harrell's C (7.2 with $W = 1$) may be a better estimator than C_{LT} with lower bias (Hartman et al. 2022). Whilst more complex IPC measures have been proposed, they do not seem to be in common usage anywhere whereas C_N and C_{LT} are both used in published research and software (McGough et al. 2021; Therneau 2015). As has been seen throughout this book, left-truncation research is still nascent. No evidence of methods for right-truncation could be found.

8

Calibration

TODO (150-200 WORDS)

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

Calibration measures evaluate the 'average' quality of survival distribution predictions. In general there is a trade-off between discrimination and calibration. A model that makes perfect individual predictions (good discrimination) might be overfit to the data and make poor predictions on new, unseen data. Whereas a model that makes perfect population predictions (good calibration) might not be able to separate between individual observations. The literature around survival analysis calibration measures is scarce (Rahman et al. 2017), potentially due to the complexity of even defining calibration in a survival context (Van Houwelingen 2000). Though this meaning does become clearer as specific metrics are introduced. As with other measure classes, only measures that can generalize beyond Cox PH models are included here but note that several calibration measures for re-calibrating PH models have been discussed in the literature (Demler, Paynter, and Cook 2015; Van Houwelingen 2000).

Calibration measures can be grouped (Andres et al. 2018) into those that evaluate distributions at a single time-point, '1-Calibration' or 'Point Calibration' measures, and those that evaluate distributions at all time-points 'distributional-calibration' or 'probabilistic calibration' measures. A point-calibration measure will evaluate a function of the predicted distribution at a single time-point whereas a probabilistic measure evaluates the distribution over a range of time-points; in both cases the evaluated quantity is compared to the observed outcome, (t, δ).

8.1 Point Calibration

Point calibration measures can be further divided into metrics that evaluate calibration at a single time-point (by reduction) and measures that evaluate an entire distribution by only considering the event time. The difference may sound subtle but it affects conclusions that can be drawn. In the first case, a calibration measure can only draw conclusions at that one time-point, whereas the second case can draw conclusions about the calibration of the entire distribution. This is the same caveat as using prediction error curves for scoring rules (Section 9.3).

8.1.1 Calibration by Reduction

Point calibration measures are implicitly reduction methods as they use classification methods to evaluate a full distribution based on a single point only (Chapter 16). For example, given a predicted survival function $\hat{S}$, one could calculate the survival function at a single time point, $\hat{S}\tau$ and then use probabilistic classification calibration measures. Using this approach one may employ common calibration methods such as the Hosmer–Lemeshow test (Hosmer and Lemeshow 1980). Measuring calibration in this way can have significant drawbacks as a model may be well-calibrated at one time-point but poorly calibrated at all others (Haider et al. 2020). To mitigate this, one could perform the Hosmer–Lemeshow test (or other applicable tests) multiple times with multiple testing correction at many (or all possible) time points, however this would be less efficient and more difficult to interpret than other measures discussed in this chapter.

8.1.2 Houwelingen's α

As opposed to evaluating distributions at one or more arbitrary time points, one could instead evaluate distribution predictions at meaningful times. van Houwelingen proposed several measures (Van Houwelingen 2000) for calibration but only one generalises to all probabilistic survival models, termed here 'Houwelingen's α'. The measure assesses if the model correctly estimates the theoretical 'true' cumulative hazard function of the underlying data generating process, $H = \hat{H}$.

The statistic is derived by noting the closely related nature of survival analysis and counting processes. In brief, for an observation i one could define the counting process $N_i(\tau) = \mathbb{I}(T_i \leq \tau, \Delta_i = 1)$, which represents the number of events i has experienced at τ. Clearly in single-event survival analysis this will be 0 before the event has been experienced or 1 at or after the event. An important quantity in counting process is the 'intensity process', which is the instantaneous rate at which events occur *given* past information. This is related to, but distinct from the hazard rate. The core difference is that the intensity process incorporates real-world information via the individual risk indicator, $R_i(\tau) = \mathbb{I}(T_i \geq \tau)$. Hence the hazard and intensity are related via $\lambda_i(\tau) = R_i(\tau)h(\tau|\mathbf{x}_i)$. The intensity process can itself be thought of as the expected number of events at t, which is 0 if the event has already occurred at $R_i(\tau) = 0$ or equal to $h(\tau|\mathbf{x}_i)$ otherwise (Hosmer Jr, Lemeshow, and May 2011). Therefore the expected number of events between $[0, \tau]$ can be obtained by the cumulative intensity process

$$\Lambda_i(\tau) = \int_0^\tau \lambda_i(s)\ ds = \int_0^\tau R_i(s)\ h(s|\mathbf{x}_i)\ ds = \int_0^{\min\{\tau, t_i\}} h(s|\mathbf{x}_i)\ ds = H(\min\{\tau, t_i\}, \mathbf{x}_i) \tag{8.1}$$

A particularly useful quantity is the expected number of events for individual i in $[0, t_i]$, which can be seen from (8.1) reduces to $H(t_i|\mathbf{x}_i)$. In a perfect model one would expect $\hat{H}(t_i|\mathbf{x}_i) = 1$ if $\delta_i = 1$ and 0 otherwise. Hence summing over all observations, $\sum \hat{H}(t_i|\mathbf{x}_i)$ gives the total number of expected events in the dataset. Houwelingen's α exploits this result to provide a simple method for assessing calibration by comparing the actual and expected number of events:

$$H_\alpha(\boldsymbol{\delta}, \hat{\mathbf{H}}, \mathbf{t}) = \frac{\sum_i \delta_i}{\sum_i \hat{H}(t_i|\mathbf{x}_i)}$$

with standard error $SE(H_\alpha) = \exp(1/\sqrt{\sum_i \delta_i})$ (Van Houwelingen 2000). A slightly more useful metric is given by $H_\alpha(\boldsymbol{\delta}, \hat{\mathbf{H}}, \mathbf{t}) - 1$ which is lower-bounded at 0 and can therefore be minimized in automated tuning processes. A model is then well-calibrated if $H_\alpha = 0$.

The next metrics we look at evaluate models across a spectrum of points to assess calibration over time.

8.2 Probabilistic Calibration

Calibration over a range of time points may be assessed quantitatively or qualitatively, with graphical methods often favoured. Graphical methods compare the average predicted distribution to the expected distribution, which can be estimated with the Kaplan-Meier curve, discussed next.

8.2.1 Kaplan-Meier Comparison

The simplest graphical comparison compares the average predicted survival curve to the Kaplan-Meier curve estimated on the testing data. Let $\hat{S}_1, ..., \hat{S}_n$ be predicted survival functions, then the average predicted survival function is the mixture: $\bar{\hat{S}} = \frac{1}{n}\sum_{i=1}^{n} \hat{S}_i(\tau)$. This estimate can be plotted next to the Kaplan-Meier estimate of the survival distribution in a test dataset (i.e., the true data for model evaluation), allowing for visual comparison of how closely these curves align. An example is given in Figure 8.1, a Cox model (CPH), random survival forest, and relative risk tree, are all compared to the Kaplan-Meier estimator. This figure highlights the advantages and disadvantages of this method. The relative risk tree is clearly poorly calibrated as it increasingly diverges from the Kaplan-Meier. In contrast, the Cox model and random forest cannot be directly compared to one another, as both models frequently overlap with each other and the Kaplan-Meier estimator. Hence it is possible to say that the Cox and forests models are better calibrated than the risk tree, however it is not possible to say which of those two is better calibrated and whether their distance from the Kaplan-Meier is significant or not at a given time (when not clearly overlapping).

This method is useful for making broad statements such as "model X is clearly better calibrated than model Y" or "model X appears to make average predictions close to the Kaplan-Meier estimate", but that is the limit in terms of useful conclusions. One could refine this method for more fine-grained information by instead using predictions to create 'risk groups' that can be plotted against a stratified Kaplan-Meier (Austin, Harrell Jr, and Klaveren 2020), however this method is harder to interpret and adds even more subjectivity around how many risk groups to create and how to create them (Patrick Royston and Altman 2013; Austin, Harrell Jr, and Klaveren 2020). The next measure we consider includes a graphical method as well as a quantitative interpretation.

8.2.2 D-Calibration

Recall that calibration measures assess whether model predictions align with population-level outcomes. In probabilistic classification, this means testing if predicted probabilities align with observed frequencies. For example, among all instances where a model predicts a 70% probability of the event happening, approximately 70% of the corresponding observations should actually experience the event. In survival analysis, calibration is extended by examining

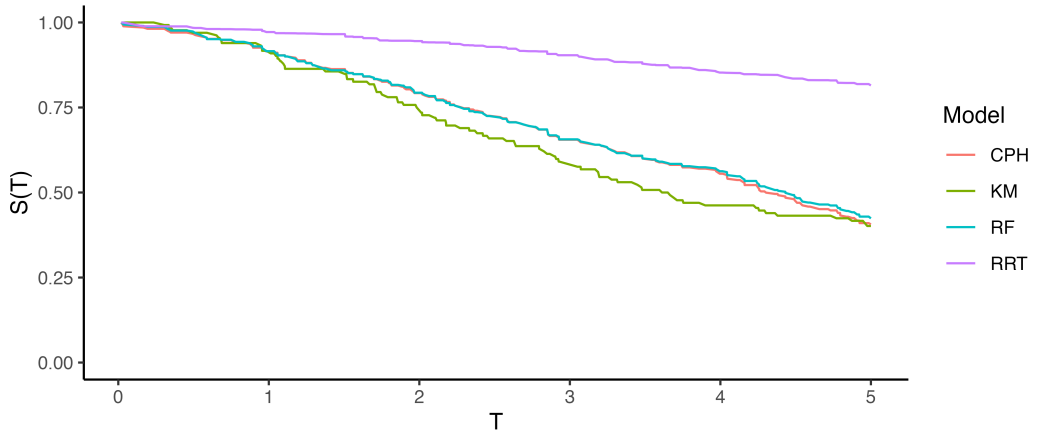


Figure 8.1: Comparing the calibration of a Cox PH (CPH), random forest (RF), and relative risk tree (RRT) to the Kaplan-Meier estimate of the survival function calculated on a test set. The calibration of RRT notably decreases over time whereas RF and CPH are closer to the Kaplan-Meier curve.

if predicted survival probabilities align with the actual distribution of event times. This is motivated by a well-known result: for any continuous random variable X, it holds that $S_X(X) \sim \mathcal{U}(0,1)$ (Angus 1994). This means that, regardless of whether the true outcome times, T, follow a Weibull, Gompertz, or any other continuous distribution, the survival probabilities evaluated at those times should be uniformly distributed in a well-calibrated model, $\hat{S}_i(T) \sim \mathcal{U}(0,1)$.

D-Calibration (Andres et al. 2018; Haider et al. 2020) leverages the fact that the event times, t_i, are i.i.d. randomly sampled event times from a distribution T, which justifies replacing T with t_i (Lemma B.2 Haider et al. 2020) and so a survival model is considered well-calibrated if the predicted survival probabilities at observed event times follow a standard Uniform distribution: $\hat{S}_i(t_i) \sim \mathcal{U}(0,1)$.

The χ^2 test-statistic is used to test if random variables follow a particular distribution:

$$\chi^2 := \sum_{g=1}^{G} \frac{(o_g - e_g)^2}{e_g}$$

where o_g, e_g are respectively the number of observed and expected events in groups $g = 1, ..., G$. In this case, the χ^2 statistic is testing if there is an even distribution of predicted survival probabilities across the $[0,1]$ range. In practice the test is simplified to instead compare if $\hat{S}_i(t_i) \sim \text{DiscreteUniform}(0,1)$. To do so the $[0,1]$ range is cut into G equal width bins. Now let n be the total number of observations, then, under the null hypothesis, the expected number of events in each bin is equal: $e_i = n/G$.

To calculate the observed number of events in each bin, first define which observations are in each bin. The observations in the gth bin are defined by the set:

$$\mathcal{B}_g := \{i = 1, \ldots, n : \lceil \hat{S}_i(t_i) \times G \rceil = g\}$$

where $i = 1, \ldots, n$ are the indices of the observations, $\hat{S}_i$ are predicted survival functions, t_i are observed outcome times, and $\lceil \cdot \rceil$ is the ceiling function. For example, if there are 5 bins then the bins are $\{[0, 0.2], (0.2, 0.4], (0.4, 0.6], (0.6, 0.8], (0.8, 1]\}$. So observation i would be in the fourth bin if $\hat{S}_i(t_i) = 0.7$ as $\lceil 0.7 \times 5 \rceil = \lceil 3.5 \rceil = 4$. Finally, the observed number of events is the number of observations in the corresponding set: $o_g = |\mathcal{B}_g|$.

The D-Calibration measure, or χ^2 statistic, is then defined by:

$$D_{\chi^2}(\hat{\mathbf{S}}, \mathbf{t}) := \frac{\sum_{i=1}^{B} (o_i - \frac{n}{G})^2}{n/G}$$

where $\hat{\mathbf{S}} = (\hat{S}_1 \ \hat{S}_2 \cdots \hat{S}_n)^\top$ and $\mathbf{t} = (t_1 \ t_2 \cdots t_n)^\top$.

This measure has several useful properties. Firstly, one can test the null hypothesis that a model is 'D-calibrated' by deriving a p-value from comparison to χ^2_{B-1}. Secondly, D_{χ^2} tends to zero as a model is increasingly well-calibrated, hence the measure can be used for model comparison. Finally, the theory lends itself to an intuitive graphical calibration method, known as reliability diagrams (Wilks 1990). A D-calibrated model implies:

$$p = \frac{\sum_i \mathbb{I}(t_i \leq \hat{F}_i^{-1}(p))}{n}$$

where p is some value in $[0, 1]$, $\hat{F}_i^{-1}$ is the ith predicted inverse cumulative distribution function, and n is again the number of observations. In words, the number of events occurring at or before each quantile should be equal to the quantile itself, for example 50% of events should occur before their predicted median survival time. Therefore, one can plot p on the x-axis and the right hand side of the above equation on the y-axis. A D-calibrated model should result in a straight line on $x = y$. This is visualized in Figure 8.2 for the same models as in Figure 8.1. This figure supports the previous findings that the relative risk tree is poorly calibrated in contrast to the Cox model and random forest but again no direct comparison between the latter models is possible.

Whilst D-calibration has the same problems as the Kaplan-Meier method with respect to visual comparison, at least in this case there are quantities to help draw more concrete solutions. For the models in Figure 8.2, it is clear that the relative risk tree is not D-calibrated with $p < 0.01$ indicating the null hypothesis of D-calibration (predicted survival probabilities follow $\mathcal{U}(0, 1)$) can be comfortably rejected. Whilst the D-calibration for the Cox model is smaller than that of the random forest, the difference is unlikely to be significant, as is seen in the overlapping curves in the figure.

8.3 Extensions

8.3.1 Competing risks

Numerical methods have been proposed for calibration in a competing risk setting, including using pseudo-measures (Schoop, Schumacher, and Graf 2011) and overall calibration ratios (Geloven et al. 2022). However, these are complex to implement and interpret and therefore graphical methods are more often used in practice (Monterrubio-Gómez, Constantine-Cooke, and Vallejos 2024).

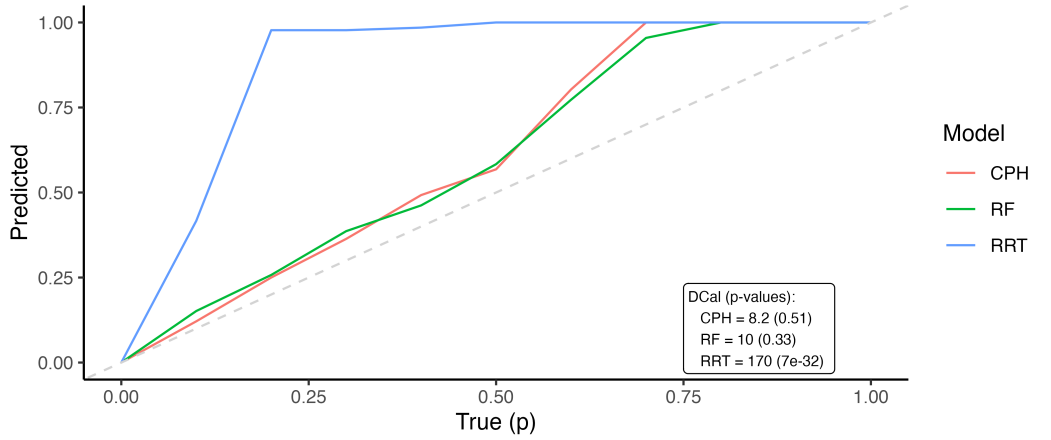


Figure 8.2: Comparing the D-calibration of a Cox PH (CPH), random forest (RF), and relative risk tree (RRT) to the expected distribution on y=x. As with Figure 8.1, the relative risk tree is clearly not D-calibrated (as supported by the figures in the bottom-right). The CPH and RF are closer to the y=x however neither follow it perfectly.

As there is not a *single* survival curve in the competing risks setting, instead one can estimate the 'true' CIF using the Aalen-Joahnson estimator (Section 5.2.2) and plot this against the average prediction in the same way as in Section 8.2.1.

As discussed earlier in this chapter, interpreting and using calibration measures is complex enough in the single event setting. Extending this to multiple plots, one for each event, makes interpretation even more complex. Using scoring rules to capture calibration performance in the competing risks setting is likely to be more straightforward (Section 9.5).

8.3.2 Other censoring and truncation types

Using graphical plots is simplest for measuring calibration when left-censoring, interval-censoring, and/or truncation are involved. In these contexts, predicted survival functions can be compared to the 'true' survival probability estimated with the Kaplan-Meier (Section 8.2.1) by using non-parametric estimators that can incorporate other censoring and truncation types as required. For example the NPMLE for interval censoring and the left-truncated risk set definition for left-truncation, see Section 4.6.2 for more.

Exercises

- Using real or simulated data, show that the sum of the cumulative hazard of a Kaplan-Meier estimator equals the number of events in the dataset.

9

Scoring Rules

TODO (150-200 WORDS)

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

Scoring rules evaluate probabilistic predictions and (attempt to) measure the overall predictive ability of a model in terms of both calibration and discrimination (Gneiting and Raftery 2007; Murphy 1973). In contrast to calibration measures, which assess the average performance across all observations on a population level, scoring rules evaluate the sample mean of individual predictions across all observations in a test set. As well as being able to provide information at an individual level, scoring rules are also popular as probabilistic forecasts are widely recognized to be superior to deterministic predictions for capturing uncertainty in predictions (A. P. Dawid 1984; A. Philip Dawid 1986). Formalisation and development of scoring rules has primarily been due to Dawid (A. P. Dawid 1984; A. Philip Dawid 1986; A. Philip Dawid and Musio 2014) and Gneiting and Raftery (Gneiting and Raftery 2007); though the earliest measures promoting "rational" and "honest" decision making date back to the 1950s (Brier 1950; Good 1952). Few scoring rules have been proposed in survival analysis, although the past few years have seen an increase in popularity in these measures. Before delving into these measures, we will first describe scoring rules in the simpler classification setting.

Scoring rules are pointwise losses, which means a loss is calculated for all observations and the sample mean is taken as the final score. To simplify notation, we only discuss scoring rules in the context of a single observation where $L_i(\hat{S}_i, t_i, \delta_i)$ would be the loss calculated for some observation i where $\hat{S}_i$ is the predicted survival function (from which other distribution functions can be derived), and (t_i, δ_i) is the observed survival outcome.

9.1 Classification Losses

In the simplest terms, a scoring rule compares two values and assigns them a score (hence 'scoring rule'), formally we'd write $L : \mathbb{R} \times \mathbb{R} \mapsto \bar{\mathbb{R}}$. In machine learning, this usually means comparing a prediction for an observation to the ground truth, so $L : \mathbb{R} \times \mathcal{P} \mapsto \bar{\mathbb{R}}$ where $\mathcal{P}$ is a set of distributions. Crucially, scoring rules usually refer to comparisons of true and

predicted *distributions*. As an example, take the Brier score (Brier 1950) defined by:

$$L_{Brier}(\hat{p}_i, y_i) = (y_i - \hat{p}_i(y_i))^2$$

This scoring rule compares the ground truth to the predicted probability distribution by testing if the difference between the observed event and the truth is minimized. This is intuitive as if the event occurs and $y_i = 1$, then $\hat{p}_i(y_i)$ should be as close to one as possible to minimize the loss. On the other hand, if $y_i = 0$ then the better prediction would be $\hat{p}_i(y_i) = 0$.

This demonstrates an important property of the scoring rule, *properness*. A loss is *proper*, if it is minimized by the correct prediction. In contrast, the loss $L_{improper}(\hat{p}_i, y_i) = 1 - L_{Brier}(\hat{p}_i, y_i)$ is still a scoring rule as it compares the ground truth to the prediction probability distribution, but it is clearly improper as the perfect prediction ($\hat{p}_i(y_i) = y_i$) would result in a score of 1 whereas the worst prediction would result in a score or 0. Proper losses provide a method of model comparison as, by definition, predictions closest to the true distribution will result in lower expected losses.

Another important property is *strict* properness. A loss is *strictly proper* if the loss is uniquely minimized by the 'correct' prediction. For example, the Brier score is minimized by only one value, which is the optimal prediction (Figure 9.1). Strictly proper losses are particular important for automated model optimization, as minimization of the loss will result in the 'optimum score estimator based on the scoring rule' (Gneiting and Raftery 2007).

Mathematically, a classification loss $L : \mathcal{P} \times \mathcal{Y} \to \bar{\mathbb{R}}$ is *proper* if for any distributions p_Y, p in $\mathcal{P}$ and for any random variables $Y \sim p_Y$, it holds that $\mathbb{E}[L(p_Y, Y)] \leq \mathbb{E}[L(p, Y)]$. The loss is *strictly proper* if, in addition, $p = p_Y$ uniquely minimizes the loss.

As well as the Brier score, which was defined above, another widely used loss is the log loss (Good 1952), defined by

$$L_{logloss}(\hat{p}_i, y_i) = -\log \hat{p}_i(y_i) \tag{9.1}$$

These losses are visualised in Figure 9.1, which highlights that both losses are strictly proper (A. Philip Dawid and Musio 2014) as they are minimized when the true prediction is made, and converge to the minimum as predictions are increasingly improved. It can also be seen from the scale of the plots that the log-loss penalizes wrong predictions stronger than the Brier score, which may be beneficial or not depending on the given use-case.

9.2 Survival Losses

Analogously to classification losses, a survival loss $L : \mathcal{P} \times \mathbb{R}_{>0} \times \{0, 1\} \to \bar{\mathbb{R}}$ is *proper* if for any distributions p_Y, p in $\mathcal{P}$, and for any random variables $Y \sim p_Y$, and C t.v.i. $\mathbb{R}_{>0}$; with $T := \min(Y, C)$ and $\Delta := \mathbb{I}(T = Y)$; it holds that, $\mathbb{E}[L(p_Y, T, \Delta)] \leq \mathbb{E}[L(p, T, \Delta)]$. The loss is *strictly proper* if, in addition, $p = p_Y$ uniquely minimizes the loss. A survival loss is referred to as outcome-independent (strictly) proper if it is only (strictly) proper when C and Y are independent.

With these definitions, the rest of this chapter lists common scoring rules in survival analysis and discusses some of their properties. As with other chapters, this list is likely not exhaustive

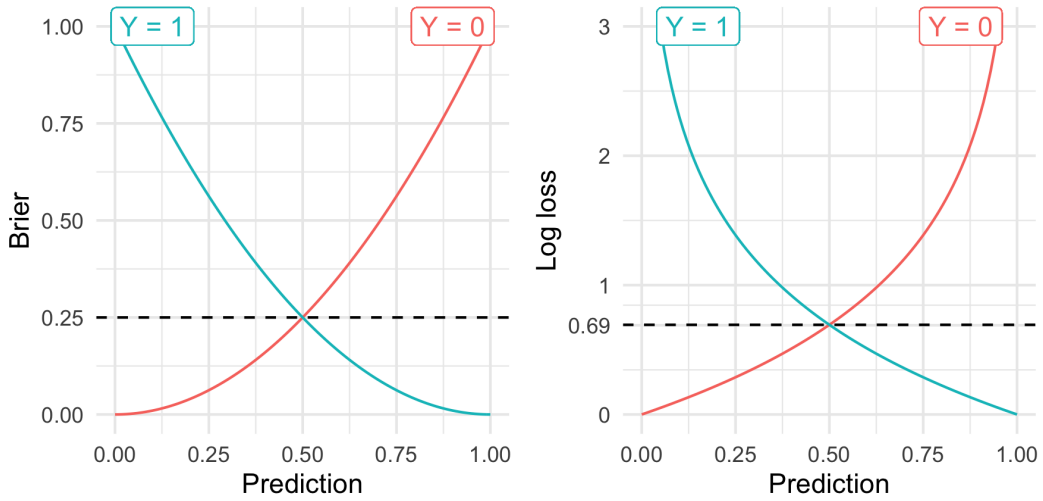


Figure 9.1: Brier and log loss scoring rules for a binary outcome and varying probabilistic predictions. x-axis is a probabilistic prediction in $[0, 1]$, y-axis is Brier score (left) and log loss (right). Blue lines are varying Brier score/log loss over different predicted probabilities when the true outcome is 1. Red lines are varying Brier score/log loss over different predicted probabilities when the true outcome is 0. Both losses are minimized when $\hat{p}_i(y_i) = y_i$.

but will cover commonly used losses. The losses are grouped into squared losses, absolute losses, and logarithmic losses, which respectively estimate the mean squared error, mean absolute error, and logloss in uncensored settings.

9.2.1 Squared Losses

The Integrated Survival Brier Score (ISBS) was introduced by Graf (Graf and Schumacher 1995; Graf et al. 1999) as an analogue to the integrated brier score in regression. It is likely the most commonly used scoring rule in survival analysis, possibly due to its intuitive interpretation.

The loss is defined by

$$L_{ISBS}(\tau^*, \hat{S}_i, t_i, \delta_i | \hat{G}_{KM}) = \int_0^{\tau^*} \frac{\hat{S}_i^2(\tau)\mathbb{I}(t_i \leq \tau, \delta_i = 1)}{\hat{G}_{KM}(t_i)} + \frac{\hat{F}_i^2(\tau)\mathbb{I}(t_i > \tau)}{\hat{G}_{KM}(\tau)} \, d\tau \tag{9.2}$$

where $\hat{S}_i^2(\tau) = (\hat{S}_i(\tau))^2$ and $\hat{F}_i^2(\tau) = (1 - \hat{S}_i(\tau))^2$, and $\tau^* \in \mathbb{R}_{\geq 0}$ is an upper threshold to compute the loss up to, and $\hat{G}_{KM}$ is the Kaplan-Meier trained on the censoring distribution for IPCW (Section 7.1).

At first glance this might seem intimidating but it is worth taking the time to understand the intuition behind the loss. Recall the classification Brier score, $L(\hat{p}_i, y_i) = (y_i - \hat{p}_i(y))^2$, this provides a method to evaluate a probability mass function at one point. In a regression setting, the *integrated Brier score*, also known as the continuous ranked probability score, is the integral of the Brier score for all real-valued thresholds (Gneiting and Raftery 2007) and hence allows predictions to be evaluated over multiple points as

$$L(\hat{F}_i, y_i) = \int (\mathbb{I}(y_i \leq \tau) - \hat{F}_i(\tau))^2 \, d\tau \tag{9.3}$$

where $\hat{F}_i$ is the predicted cumulative distribution function and τ is some meaningful threshold. As the left-hand indicator can only take one of two values 9.3 can be represented as two distinct cases, now using t instead of y to represent time:

$$L(\hat{F}_i, t_i) = \begin{cases} (1 - \hat{F}_i(\tau))^2 = \hat{S}_i^2(\tau), & \text{if } t_i \leq \tau \\ (0 - \hat{F}_i(\tau))^2 = \hat{F}_i^2(\tau), & \text{if } t_i > \tau \end{cases} \tag{9.4}$$

In the first case, the observation experienced the event before τ, hence the optimal prediction for $\hat{F}(\tau)$ (the probability of experiencing the event before τ) is 1, and therefore the optimal $\hat{S}(\tau)$ is 0. Conversely, in the second case has not experienced the event yet, the optimal $\hat{F}(\tau)$ is 0. The loss therefore meaningfully represents the ideal predictions in the two possible real-world scenarios.

The final component of the loss is accommodating for censoring. At τ an observation will either have:

1. Not experienced any outcome: $t_i > \tau$;
2. Experienced the event: $t_i \leq \tau \wedge \delta_i = 1$; or
3. Been censored: $t_i \leq \tau \wedge \delta_i = 0$

Censored observations are discarded after the censoring time as evaluating predictions after this time is impossible as the ground truth is unknown. To compensate for removing observations, IPCW (Section 7.1.1) is again used to upweight predictions as τ increases. IPC weights, W_i are defined such that observations are either weighted by $\hat{G}_{KM}(\tau)$ when they have not yet experienced the event or by their final observed time, $\hat{G}_{KM}(t_i)$, otherwise (Table 9.1).

Table 9.1: IPC weighting scheme for the ISBS.

$W_i := W(t_i, \delta_i)$	$t_i > \tau$	$t_i \leq \tau$
$\delta_i = 1$	$\hat{G}_{KM}^{-1}(\tau)$	$\hat{G}_{KM}^{-1}(t_i)$
$\delta_i = 0$	$\hat{G}_{KM}^{-1}(\tau)$	0

When censoring is uninformative, the Graf score consistently estimates the mean square error (Gerds and Schumacher 2006). Despite this, the score is not strictly proper and even its properness is in doubt (in doubt not proven due to their being open debate in the literature about how to define properness in a survival context) (Rindt et al. 2022). Fortunately, as the score is deeply embedded in the literature, experiments have demonstrated that scores generated from using the ISBS only differ very slightly to a strictly proper alternative (R. Sonabend et al. 2025).

9.2.2 Logarithmic losses

The development of logarithmic losses follows from adapting the negative likelihood for censored datasets. Consider the usual negative likelihood in a regression setting, which is a standard measure for evaluating a model's performance:

$$L_{NLL}(\hat{f}_i, y_i) = -\log[\hat{f}(y_i)]$$

for a predicted density function $\hat{f}_i$ and true outcome y_i. Note this is analagous to the classification log loss in (9.1) with the probably mass function replaced with the density function.

Now recall (4.13) from Section 4.6.1, which gives the contribution from a single observation as

$$\mathcal{L}(t_i) \propto \begin{cases} f(t_i), & \text{if } i \text{ is uncensored} \\ S(t_i), & \text{if } i \text{ is right-censored} \\ F(t_i), & \text{if } i \text{ is left-censored} \\ S(l_i) - S(r_i), & \text{if } i \text{ is interval-censored} \end{cases}$$

where r_i, l_i are the boundaries of the censoring interval (adaptations in the presence of left-truncation as described in Section 4.6.1 may also be applied).

The log-loss can then be constructed depending on what type of censoring or truncation is present in the data. For example, if only right-censoring is present then the right-censored logloss (RCLL) is defined as:

$$L_{RCLL}(\hat{S}_i, t_i, \delta_i) = -\log\left(\delta_i \hat{f}_i(t_i) + (1 - \delta_i)\hat{S}_i(t_i)\right) \tag{9.5}$$

If censoring is independent of the event (Section 4.4) then this scoring rule is strictly proper (Avati et al. 2020). The loss is also highly interpretable as a measure of predictive performance when broken down into its two halves:

1. An observation censored at t_i has not experienced the event and hence the ideal prediction would be close to $S(t_i) = 1$; correspondingly (9.5) becomes $-\log(1) = 0$.
2. If an observation experiences the event at t_i, then the ideal prediction would be close to $f(t_i) = \infty$ or $p(t_i) = 1$ in the discrete case; therefore (9.5) equals $-\infty$ or 0 for continuous and discrete time respectively.

Analagous losses follow when left- and/or interval-censoring is present by using the objective functions in Section 4.6.1.

Other logarithmic losses have also been proposed, such as the integrated survival log loss (ISLL) in Graf et al. (1999). The ISLL is similar to the ISBS except $\hat{S}_i^2$ and $\hat{F}_i^2$ are replaced with $\log(\hat{F}_i)$ and $\log(\hat{S}_i)$ respectively. To our knowledge, the ISLL does not appear used in practice and nor is there a practical benefit over other losses – though we note the work of Alberge et al. (2025) discussed in Section 9.5.

9.2.3 Absolute Losses

The final class of losses considered here can be viewed as analogs of the mean absolute error in an uncensored setting. The absolute survival loss, developed over time by Schemper and Henderson (2000) and Schmid et al. (2011) is similar to the ISBS but removes the squared term:

$$L_{ASL}(\hat{S}_i, t_i, \delta_i | \hat{G}_{KM}) = \int_0^{\tau^*} \frac{\hat{S}_i(\tau)\mathbb{I}(t_i \leq \tau, \delta_i = 1)}{\hat{G}_{KM}(t_i)} + \frac{\hat{F}_i(\tau)\mathbb{I}(t_i > \tau)}{\hat{G}_{KM}(\tau)} \, d\tau$$

where $\hat{G}_{KM}$ and τ^* are as defined above. Analogously to the ISBS, the absolute survival loss consistently estimates the mean absolute error when censoring is uninformative (Schmid et al. 2011) but there are also no proofs or claims of properness. The absolute survival loss and ISBS tend to yield similar results (Schmid et al. 2011) but in practice the former does not appear to be widely used.

9.3 Prediction Error Curves

As well as evaluating probabilistic outcomes with integrated scoring rules, non-integrated scoring rules can be utilized for evaluating distributions at a single point. For example, instead of evaluating a probabilistic prediction with the ISBS over $\mathbb{R}_{\geq 0}$, one could compute the Brier score at a single time-point, $\tau \in \mathbb{R}_{\geq 0}$, only. Plotting these for varying values of τ results in *prediction error curves*, which provide a simple visualization for how predictions vary over time. Prediction error curves are mostly used as a graphical guide when comparing few models, rather than as a formal tool for model comparison. Example prediction error curves are provided in Figure 9.2 for the ISBS where the the Cox PH consistently outperforms the SVM.

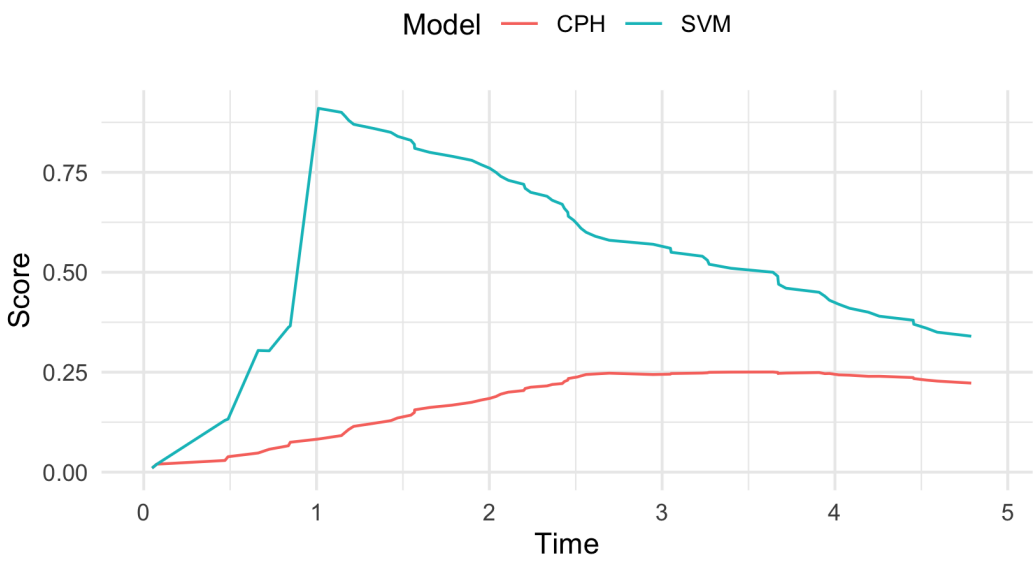


Figure 9.2: Prediction error curves for the CPH and SVM models from Chapter 8. x-axis is time and y-axis is the ISBS computed at different time-points. The CPH (red) performs better than the SVM (blue) as it scores consistently lower. Trained and tested on randomly simulated data from **mlr3proba**.

9.4 Baselines and ERV

A common criticism of scoring rules is a lack of interpretability, for example, an ISBS of 0.5 or 0.0005 has no meaning by itself, so below we present two methods to help overcome this problem.

The first method is to make use of baselines for model comparison, which are models or values that can be utilized to provide a reference for a loss and provide a universal method to judge all models of the same class (Gressmann et al. 2018). In classification, it is possible to derive analytical baseline values, for example a Brier score is considered 'bad' if it is above 0.25 or a log loss if it is above 0.693 (Figure 9.1), this is because these are the values obtained if you always predicted probabilities as 0.5, which is the best un-informed (i.e., data independent) baseline in a binary classification problem. In survival analysis, simple analytical expressions are not possible as losses are dependent on the unknown distributions of both the survival and censoring time. For this reason it is advisable to include baselines models for model comparison. Common baselines include the Kaplan-Meier estimator (Section 4.3) and Cox PH (**?@sec-surv-models-crank**). As a rule of thumb, if a model performs worse than the Kaplan-Meier than it's considered 'bad', whereas if it outperforms the Cox PH then it is considered 'good'.

As well as directly comparing losses from a 'sophisticated' model to a baseline, one can also compute the percentage increase in performance between the sophisticated and baseline models, which produces a measure of explained residual variation (ERV) (Edward L. Korn and Simon 1990; Edward L. Korn and Simon 1991). For any survival loss L, the ERV is,

$$R_L(S, B) = 1 - \frac{L_{|S}}{L_{|B}}$$

where $L_{|S}$ and $L_{|B}$ is the loss computed with respect to predictions from the sophisticated and baseline models respectively.

The ERV interpretation makes reporting of scoring rules easier within and between experiments. For example, say in experiment A we have $L_{|S} = 0.004$ and $L_{|B} = 0.006$, and in experiment B we have $L_{|S} = 4$ and $L_{|B} = 6$. The sophisticated model may appear worse at first glance in experiment A (as the losses are very close) but when considering the ERV we see that the performance increase is identical (both $R_L = 33\%$), thus providing a clearer way to compare models.

9.5 Extensions

9.5.1 Competing risks

Similarly to discrimination measures, scoring rules are primarily used with competing risks by evaluating cause-specific probabilities individually (Geloven et al. 2022; C. Lee et al. 2018; Bender et al. 2021).

For example, given the cause-specific survival, S_e, density, f_e, and cumulative distribution function, F_e, the right-censored log-loss for event e is defined as

$$L^{e}_{RCLL;i}(\hat{S}_{i;e}, t_i, \delta_i) = -\log[\delta_i \hat{f}_{i;e}(t_i) + (1-\delta_i)\hat{S}_{i;e}(t_i)]$$

Similar logic can be applied to the ISBS and other scoring rules.

Recently, an all-cause logarithmic scoring rule has been proposed which makes use of the IPC weighting in the ISBS (Alberge et al. 2024, 2025):

$$L_{AC;i}(\hat{S}_i, t_i, \delta_i) = \sum_{e=1}^{k} \frac{\mathbb{I}(t_i \leq \tau, \delta_i = e)\log(\hat{F}_{i;e})}{\hat{G}(t_i)} + \frac{\mathbb{I}(t_i > \tau)\log(\hat{S}_i(\tau))}{\hat{G}(\tau)}$$

This 'all-cause' loss is an adaptation of the ISLL (Section 9.2.2) with an adaptation to the weights to handle competing risks. Comparing this loss to the decomposition in Section 9.2.1, we can see observations either: experience the event of interest, in which case their cause-specific CIF is evaluated; do not experience any event and so the all-cause survival is evaluated; or experience a different event and contribute nothing to the loss.

This 'all-cause' loss could be minimized in an automated procedure and/or used for model comparison more easily than cause-specific losses. However, doing so may hide cause-specific patterns, for example a model might have better performance for some causes than others. If performance in individual causes is important, then cause-specific losses may be preferred, optionally with multi-objective optimization methods (Morales-Hernández, Van Nieuwenhuyse, and Rojas Gonzalez 2023).

9.5.2 Other censoring and truncation types

We have already seen in Section 9.2.2 how logarithmic losses can be extended to handle more diverse censoring and truncation types by updating the likelihood function as necessary. For squared losses there has been substantially less development in this area, a notable extension is an adaptation to the Brier score for administrative censoring (Kvamme and Borgan 2023). There is potential to extend the ISBS to handle interval censoring by estimating the probability of survival within the interval (Tsouprou 2015), however research is sparse and there is no evidence of use in practice.

9.6 Conclusion

Key takeaways

- Scoring rules are a useful tool for measuring a model's overall predictive ability, taking into account calibration and discrimination.
- Strictly proper scoring rules allow models to be compared to one another, which is important when choosing models in a benchmark experiment.
- Many scoring rules for censored data are *not* strictly proper, however experiments suggest that improper rules still provide useful and trustworthy results (R. Sonabend et al. 2025)

Limitations

- Scoring rules can be difficult to interpret but ERV representations can be a helpful way to overcome this.
- There is no consensus about which scoring rule to use and when so in practice multiple scoring rules may have to be reported in experiments to ensure transparency and fairness of results.
- For non- and semi-parametric survival models that return distribution predictions, estimates of $f(t)$ are not readily available and require approximations (Rindt et al. 2022), hence logarithmic losses such as RCLL can often not be directly used in practice.

Further reading

- A. Philip Dawid and Musio (2014) and Gneiting and Raftery (2007) provide a comprehensive summary of scoring rules in regression and classification settings.
- Rindt et al. (2022), R. Sonabend et al. (2025) and Yanagisawa (2023) review survival scoring rules, including loss forms not discussed in this chapter such as pinball losses.
- Choodari-Oskooei, Royston, and Parmar (2012a), Choodari-Oskooei, Royston, and Parmar (2012b), and Rahman et al. (2017) compare measures for external validation including some scoring rules.

10

Survival Time Measures

TODO (150-200 WORDS)

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

When it comes to evaluating survival time predictions, there are few measures available at our disposal. As a result of survival time predictions being uncommon compared to other prediction types (Chapter 6), there are limited survival time evaluation measures in the literature. The most common approach (P. Wang, Li, and Reddy 2019) is to use regression measures by ignoring censored observations and potentially adding some IPC weighting.

10.1 Uncensored distance measures

Survival time measures are often referred to as 'distance' measures as they measure the distance between the true, $(t, \delta = 1)$, and predicted, $\hat{t}$, values. These are presented in turn with brief descriptions of their interpretation. The measures below can be inflated using an IPC weighting by dividing by $\hat{G}_{KM}(t_i)$. However, evidence suggests that adding this weighting does not improve the measure with respect to ranking if one model is better than another (S.-A. Qi et al. 2023).

For all measures define $d := \sum_i \delta_i$, which is the number of uncensored observations in the dataset.

Censoring-ignored mean absolute error, MAE_C

In regression, the mean absolute error (MAE) is a popular measure because it is intuitive to understand as it measures the absolute difference between true and predicted outcomes; hence intuitively one can understand that a model predicting a height of 175cm is clearly better than one predicting a height of 180cm, for a person with true height of 174cm.

$$MAE_C(\hat{\mathbf{t}}, \mathbf{t}, \boldsymbol{\delta}) = \frac{1}{d} \sum_{i=1}^{d} \delta_i |t_i - \hat{t}_i|$$

Censoring-ignored mean squared error

In comparison to MAE, the mean squared error (MSE), computes the squared differences between true and predicted values. While the MAE provides a smooth, linear, 'penalty' for increasingly poor predictions (i.e., the difference between an error of predicting 2 vs. 5 is still 3), but the square in the MSE means that larger errors are quickly magnified (so the difference in the above example is 9). By taking the mean over all predictions, the effect of this inflation is to increase the MSE value as larger mistakes are made.

$$MSE_C(\hat{\mathbf{t}}, \mathbf{t}, \boldsymbol{\delta}) = \frac{1}{d}\sum_{i=1}^{d} \delta_i (t_i - \hat{t}_i)^2$$

Censoring-ignored root mean squared error

Finally, the root mean squared error (RMSE), is simply the square root of the MSE. This allows interpretation on the original scale (as opposed to the squared scale produced by the MSE). Given the inflation effect for the MSE, the RMSE will be larger than the MAE as increasingly poor predictions are made; it is common practice for the MAE and RMSE to be reported together.

$$RMSE_C(\hat{\mathbf{t}}, \mathbf{t}, \boldsymbol{\delta}) = \sqrt{MSE_C(\hat{\mathbf{t}}, \mathbf{t}, \boldsymbol{\delta})}$$

Note that these equations *completely* remove censored observations from the dataset under evaluation. This is in contrast to how IPC is used in the C-index (Section 7.1.1) and various scoring rules (Chapter 9), where censored observations are at least partially included in the calculation. By failing to include censored observations at all, these measures are only evaluating survival models on a (potentially biased) sample of the overall test data, and hence can never be fairly used to estimate the model's performance on new data.

10.2 Over- and under-predictions

The distance measures just discussed assume that the error for an over-prediction ($\hat{t} > t$) should be equal to an under-prediction ($\hat{t} < t$). That is, they assume it is 'as bad' if a model predicts an outcome time being 10 years longer than the truth compared to being 10 years shorter. In the survival setting, this assumption is often invalid as it is generally preferred for models to be overly cautious, hence to predict negative events to happen sooner (e.g., predict a life-support machine fails after three years not five if the truth is actually four) and to predict positive events to happen later (e.g., predict a patient recovers after four years not two if the truth is actually three). A simple method to incorporate this imbalance between over- and under-predictions is to add a weighting factor to any of the above measures, for example the MAE_C might become

$$MAE_C(\hat{\mathbf{t}}, \mathbf{t}, \boldsymbol{\delta}, \lambda, \mu, \phi) = \frac{1}{d}\sum_{i=1}^{m} \delta_i |(t_i - \hat{t}_i)[\lambda \mathbb{I}(t_i > \hat{t}_i) + \mu \mathbb{I}(t_i < \hat{t}_i) + \phi \mathbb{I}(t_i = \hat{t}_i)]|$$

where λ, μ, ϕ are any Real number to be used to weight over-, under-, and exact-predictions, and d is as above. The choice of these are highly context dependent and could even be tuned (Section 3.5).

10.3 Extensions

10.3.1 Competing risks

In a competing risks setting, there is no obvious metric to evaluate survival time predictions, primarily because there is no meaningful interpretation for an 'all-cause survival time' or a 'cause-specific survival time'. If an observation could realistically experience one of multiple, mutually-exclusive events, then predicting the time to one particular event has no inherent meaning without first attaching a probability of the event taking place (i.e., the CIF), hence evaluating this probability is a more sensible approach for evaluating a competing risks model's predictive performance.

10.3.2 Other censoring and truncation types

Given the above measures simply remove censored observations, the same measures can be easily applied to datasets with left-censoring and interval-censoring – with the same caveats applied. To handle truncation, the formulae above can be extended to remove truncated events, which will introduce the same biases as removing censored events.

10.4 Conclusion

Key takeaways

- There are few measures for evaluating survival time predictions, likely due this being a less popular survival task;
- Simple analogues to regression measures can be created by removing censored observations and optionally adding an IPC weighting;
- Weighting measures to account for over- and under-predictions may be useful in real-world settings.

Further reading

- S.-A. Qi et al. (2023) survey MAE-based survival losses, including a hinge loss introduced by Chapfuwa et al. (2018), as well as losses based on surrogate values including mean residual lifetime (Haider et al. 2020) and pseudo-observations. This is effectively the same as using the MAE to evaluate imputed censored predictions (Chapter 20).

Part III

Models

11

Classical Statistical Models

TODO (150-200 WORDS)

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

Defining if a model is 'machine learning' is surprisingly difficult, with terms such as 'iterative' and 'loss optimization' being frequently used in often clunky definitions. This book defines machine learning models as those that:

1. Require intensive model fitting procedures such as recursion or iteration;
2. Are flexible with hyper-parameters that can be tuned;
3. Trained using some form of loss optimization;
4. Are 'black boxes' without simple closed-form expressions that can be easily interpreted;
5. Aim to minimize the difference between predictions and truth whilst placing minimal assumptions on the underlying data form.

In contrast, 'classical statistical' (or 'classical') models, refer here to low-complexity models that are either non-parametric or have a simple closed-form expression with parameters that can be fit with maximum likelihood estimation (or an analogous method). Further, classical models are assumed to be fast to fit and highly interpretable, though can be inflexible and may make unreasonable assumptions as they assume underlying probability distributions to model data.

Whilst this book is focused on machine learning, there are a few reasons to include a chapter on classical models.

Firstly, the boundary between machine learning and classical models is fuzzy as the latter can be augmented with modern machine learning methods to create a model that could be termed as 'machine learning' (Section 11.6). In fact, several papers have demonstrated that augmenting classical models in this way yields models that outperform machine learning alternatives even on high-dimensional data (when the number of variables far exceeds the number of rows) (Yunwei Zhang et al. 2021; Spooner et al. 2020). Secondly, the majority of machine learning survival algorithms make use of classical models, often using the Kaplan-Meier estimator (Section 11.1) and/or assuming a proportional hazards form (Section 11.2), as a central component to construct an algorithm around. Finally, on lower dimensional data, classical algorithms often outperform machine learning models even without further adjustment (Beaulac et al. 2020; Burk et al. 2024).

Therefore, a robust understanding of classical models is imperative to fairly construct and evaluate machine learning survival models. This chapter begins with a recap of estimators seen in the first part of this book and demonstrating how they can be used as predictive models, with further estimators then introduced. Semi- and fully-parametric models are then introduced, most notably the Cox proportional hazards model and the accelerated failure time models. Finally, methods to improve classical models through machine learning methodology is presented.

11.1 Non-Parametric Estimators

The Kaplan-Meier estimator (Kaplan and Meier 1958) was introduced in Section 4.3 as the step-function

$$\hat{S}_{KM}(\tau) = \prod_{k:t_{(k)} \leq \tau} \left(1 - \frac{d_{t_{(k)}}}{n_{t_{(k)}}}\right) \tag{11.1}$$

where $t(k)$ are ordered event times, $d_{t_{(k)}}$ and $n_{t_{(k)}}$ are the number of events and observations at risk at $t_{(k)}$ respectively. In Section 4.3, this was discussed as an estimator for visualizing survival distributions and for descriptive analysis. However, now consider how this estimator could be used as a simple predictive method. As an unconditional non-parametric method, the estimator assumes no distribution for survival times and ignores any covariate data. Say we are given the following dataset of survival outcomes:

Table 11.1: Sample of the `rats` dataset from R package **survival**.

litter	rx	sex	time	status
6	0	m	62	0
57	0	f	55	1
60	1	m	104	0
85	1	f	92	1
97	0	f	91	0

Table 11.2 below demonstrates construction of the Kaplan-Meier estimator at each possible time-point, where the second column is the contribution to the estimator at the given time:

Table 11.2: Kaplan-Meier contributions (second column) and estimation (third column) at each time τ based on the data from Table 11.1.

τ	$1 - d/n$	$\hat{S}_{KM}(\tau)$
55	0.8	0.8
62	1	0.8
91	1	0.8
92	0.5	0.4
104	1	0.4

Now if Table 11.1 is taken to be training data, then the values in Table 11.2 can be used to predict the probability of survival at a given time. For example, without needing to know any other information about a new rat that enters the data (for these estimators ignore covariates), there is an 80% chance of survival until $\tau = 60$, which drops to 40% after 92 days.

Estimating the Kaplan-Meier on training data and using those values for predictions of new data, yields a 'baseline' model that can be compared to more sophisticated alternatives to improve interpretability in model evaluation (Section 9.4). In this book, $\hat{S}_{KM}$ specifically refers to the Kaplan-Meier estimator fit on some training data. Similarly, $\hat{G}_{KM}$ specifically refers to the Kaplan-Meier estimator fit on the observed censoring times $(t_i, 1 - \delta_i)$ from some training data.

The Nelson-Aalen estimator (O. Aalen 1978; Nelson 1972), introduced in Section 4.6.2, can also be used as a baseline predictive model.

$$\hat{H}_{NA}(\tau) = \sum_{t_{(k)} \leq \tau} \frac{d_{t_{(k)}}}{n_{t_{(k)}}}.$$

The Kaplan-Meier and Nelson-Aalen estimators are consistent estimators for the survival and cumulative hazard functions respectively; the former is more widely utilized as a baseline estimator (Herrmann et al. 2021; Huang et al. 2020a; P. Wang, Li, and Reddy 2019). As well as supporting in interpretability, both methods can be used for graphical calibration of models (Section 8.2.1), components of complex models (Chapter 16), and other diagnostic graphical tools (Habibi et al. 2018; Jager et al. 2008; Moghimi-dehkordi et al. 2008).

Extensions to other censoring and truncation types as well as event history analysis more generally follows from using the estimators defined in Section 4.6.2, Section 5.2.2, and Section 5.3.4.

By ignoring covariates, neither estimator is expected to perform well in practice as predictive models. Moreover, the assumption of uninformative censoring rarely holds true. One alternative is to consider a conditional non-parametric estimator that can take into account covariates. The Akritas estimator (Michael G. Akritas 1994) is usually defined by (Blanche, Dartigues, and Jacqmin-Gadda 2013):

$$\hat{S}(\tau|\mathbf{x}^*, \lambda) = \prod_{k:t(k) \leq \tau} \left(1 - \frac{\sum_{i=1}^{n} K(\mathbf{x}^*, \mathbf{x}_i|\lambda)\mathbb{I}(t_i = t(k), \delta_i = 1)}{\sum_{i=1}^{n} K(\mathbf{x}^*, \mathbf{x}_i|\lambda)\mathbb{I}(t_i \geq t(k))}\right) \tag{11.2}$$

where K is a kernel function, usually $K(x, y|\lambda) = \mathbb{I}(|\hat{F}_X(x) - \hat{F}_X(y)| < \lambda), \lambda \in (0, 1]$, $\hat{F}_X$ is the empirical distribution function of the data, and λ is a hyper-parameter. The estimator can be interpreted as a conditional Kaplan-Meier estimator which is computed on a neighbourhood of subjects closest to $\mathbf{x}^*$. In fact, if $\lambda = 1$ then $K(\cdot|\lambda) = 1$ and 11.2 is identical to 11.1.

The formulation in 11.2 includes fitting and predicting in one step as the usual application of the model is as a non-parametric estimator. By first estimating $\hat{F}_X$ on separate training data, the estimator can be used as a baseline predictive model.

11.2 Proportional Hazards

This section begins with an introduction to the proportional hazards concept using the Cox PH model and then moves to fully parametric proportional hazards, with WeibullPH as a motivating example.

11.2.1 Semi-Parametric PH

The Cox Proportional Hazards (Cox PH) (Cox 1972), or Cox model, is likely the most widely known semi-parametric model and the most studied survival model (Reid 1994; P. Wang, Li, and Reddy 2019). Let $\eta_i = \mathbf{x}_i^\top \boldsymbol{\beta}$ be the linear predictor for some observation i with covariates $\mathbf{x}_i$ and model coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$, then the Cox model assumes the hazard function for i follows the form:

$$h_{Cox}(\tau|\mathbf{x}_i) = h_0(\tau)\exp(\eta_i) \tag{11.3}$$

or in equivalent forms:

$$S_{Cox}(\tau|\mathbf{x}_i) = S_0(\tau)^{\exp(\eta_i)} \tag{11.4}$$

$$H_{Cox}(\tau|\mathbf{x}_i) = H_0(\tau)\exp(\eta_i) \tag{11.5}$$

Ignoring the h_0 term for now, this form encodes the eponymous 'proportional hazards' assumption as the individual's hazard at time τ is directly proportional to their own covariates: $h(\tau|\mathbf{x}_i) \propto \exp(\eta_i)$. In other words, a unit change in a covariate acts multiplicatively on the estimated hazard. Further, the hazard ratio, which is a measure of the difference in risk, between two different subjects, depends solely on the value of their linear predictors and not on time. Take the special case of a model with one covariate, x, and coefficient, β, see what happens when x is increased by one unit:

$$\frac{h_{Cox}(\tau|x+1)}{h_{Cox}(\tau|x)} = \frac{h_0(\tau)\exp((x+1)\beta)}{h_0(\tau)\exp(x\beta)} = \exp(\beta) \tag{11.6}$$

The difference in risk depends only on β and is independent of time, specifically

$$h_{Cox}(\tau|x+1) = \exp(\beta)h_{Cox}(\tau|x) \tag{11.7}$$

Estimation of the β terms is possible by ignoring h_0 as a 'nuissance parameter' and using the 'partial likelihood' function (Cox 1975):

$$\mathcal{L}(\beta) = \prod_{k:t_{(k)}} \left(\frac{\exp(\eta_k)}{\sum_{j \in \mathcal{R}_{t_{(k)}}} \exp(\eta_j)} \right) \tag{11.8}$$

with log-likelihood

$$\updownarrow(\beta) = \sum_{k:t_{(k)}} \left(\eta_k - \log\Big(\sum_{j \in \mathcal{R}_{t_{(k)}}} \exp(\eta_j) \Big) \right)$$

Given these formulas, one can make relative risk predictions (Chapter 6) as well as analysing patterns in the data by computing hazard ratios, which provide a method for identifying the covariates that have a significant effect on an individual's hazard. However, to make a full survival distribution prediction, the baseline hazard must be considered.

The baseline hazard, h_0, captures the average (baseline) risk for all observations at a given time-point. In complex machine learning algorithms, the Kaplan-Meier estimator is often used to estimate the baseline survival, S_0, the hazard then follows from $h_0(\tau) = -S'(\tau)/S(\tau)$. However, in the case of the Cox model, the conditional non-parametric Breslow estimator (Cox 1975) is more suitable as it accounts for covariates and is directly related to the partial likelihood function (11.8):

$$\hat{H}_{Bres}(\tau) = \sum_{t_{(k)} \leq \tau} \frac{d_{t_{(k)}}}{\sum_{j \in \mathcal{R}_{t_{(k)}}} \exp(\hat{\eta}_j)} \tag{11.9}$$

where $\hat{\eta}_j$ is the estimated linear predictor for observation j. Note that if the value for all covariates was zero, which one could think of equivalent as there being no covariates, then the Breslow estimator is identical to the Nelson-Aalen estimator:

$$\hat{H}_{Bres}(\tau) = \sum_{t_{(k)} \leq \tau} \frac{d_{t_{(k)}}}{\sum_{j \in \mathcal{R}_{t_{(k)}}} 1} = \sum_{t_{(k)} \leq \tau} \frac{d_{t_{(k)}}}{n_{t_{(k)}}} = \hat{H}_{NA}(\tau)$$

With these formulae, the Cox PH can be used as a predictive model by using training data to estimate $\hat{\boldsymbol{\beta}}$ with (11.8). These fitted coefficients are then used to predict $\hat{\boldsymbol{\eta}}$ for new observations and finally the cumulative baseline hazard is computed with (11.9) to return a predicted distribution.

The Cox model is a powerful tool that is constantly being found to outperform machine learning models (Michael F. Gensheimer and Narasimhan 2018; Luxhoj and Shyur 1997; Van Belle et al. 2011). Moreover, it is highly interpretable and as such has a long history of usage in clinical prediction modelling and analysis. However, the proportional hazards assumption is often violated in real life, leaving the model to be a questionable choice when used for data analysis. It is possible to extend the model to handle time-varying coefficients and model stratification (Cox 1972), however it is surprisingly difficult to make meaningful and interpretable predictions from a time-varying or stratified model (Reid 1994).

In a predictive setting, it also may not matter if the PH assumption is violated, as long as the model is clearly demonstrated to outperform others (with less stringent assumptions) in an empirical benchmark experiment.

11.2.2 Parametric PH

The baseline hazard is advantageous as it removes any burden to assume a particular underlying probability distribution. However, there are some cases where modelling a particular distribution may make sense. On these occasions, the baseline hazard can be replaced by the hazard function corresponding to a particular probability distribution, with

three common choices (John D. Kalbfleisch and Prentice 2011; P. Wang, Li, and Reddy 2019) being the Exponential, Gompertz, and Weibull distributions. The Weibull distribution is particularly important as it reduces to the Exponential distribution when the shape parameter equals 1 and moreover it is the only distribution for which both the proportional hazards and accelerated failure time assumptions can hold true.

Assuming a PH model one can plug in the hazard and survival functions from the Weibull distribution into 11.3 and 11.4 respectively. First recall for a Weibull(γ, λ) distribution with shape parameter γ and scale parameter λ, the relevant functions can be given by (J. D. Kalbfleisch and Prentice 1973):

$$h(x) = \lambda\gamma x^{\gamma-1}$$

and

$$S(x) = \exp(-\lambda x^{\gamma})$$

Taking these to be the baseline hazard and survival functions respectively, they can be substituted into the Cox model as follows:

$$h_{WeibullPH}(\tau|\mathbf{x}_i) = (\lambda\gamma\tau^{\gamma-1})\exp(\eta_i) \tag{11.10}$$

or equivalently

$$S_{WeibullPH}(\tau|\mathbf{x}_i) = (\exp(-\lambda\tau^{\gamma}))^{\exp(\eta_i)}$$

Finally, these formulae can now be used to define the full likelihood (Section 4.6.1) for the WeibullPH model:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \prod_{i=1}^{n} h_Y(t_i|\mathbf{x}_i, \boldsymbol{\theta})^{\delta_i} S_Y(t_i|\mathbf{x}_i, \boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} \Big((\lambda\gamma t_i^{\gamma-1}\exp(\eta_i))^{\delta_i}\Big)\Big(\exp(-\lambda t_i^{\gamma}\exp(\eta_i))\Big)\end{aligned}$$

with log-likelihood

$$\begin{aligned}\updownarrow(\boldsymbol{\theta}) &= \sum_{i=1}^{n} \delta_i[\log(\lambda\gamma) + (\gamma-1)\log(t_i) + \eta_i] - \lambda t_i^{\gamma}\exp(\eta_i) \\ &\propto \sum_{i=1}^{n} \delta_i[\log(\lambda\gamma) + \gamma\log(t_i) + \eta_i] - \lambda t_i^{\gamma}\exp(\eta_i)\end{aligned}$$

These likelihoods can be passed into machine learning models that are constructed around closed-form likelihood expressions to be optimized, for example in boosting (Chapter 14) and neural networks (Chapter 15). Parameters can then be fit using maximum likelihood estimation (MLE) with respect to all unknown parameters $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \gamma, \lambda\}$. Expansion to other censoring types and truncation follows by using other likelihood forms presented in Section 4.6.

When considering which probability distributions to model in predictive experiments, Weibull is a common starting choice (Hielscher et al. 2010; R. and J. 1968; Rahman et al. 2017), its two parameters make it a flexible fit to data but on the other hand it can be easily reduced to Exponential when $\gamma = 1$. Gompertz (Gompertz 1825) is commonly used in medical domains, especially when describing adult lifespans. In a machine learning context, one can select the optimal distribution for future predictive performance by running a benchmark experiment. Moreover, it transpires that model inference and predictions are largely insensitive to the choice of distribution (Collett 2014; Reid 1994). In contrast to the semi-parametric Cox model, fully parametric PH models can predict absolutely continuous survival distributions, they do not treat the baseline hazard as a nuisance, and in general will result in more precise and interpretable predictions if the distribution is correctly specified (Reid 1994; Patrick Royston and Parmar 2002).

11.2.3 Competing risks

There are two common methods to extend the Cox model to the competing risks setting. The first makes use of the cause-specific hazard to fit a cause-specific Cox model, the second fits a 'subdistribution' hazard.

Cause-specific PH models

In the first case, the baseline hazard function and the coefficients are replaced with cause-specific hazard and coefficients:

$$h_{Cox;e}(\tau|\mathbf{x}_i) = h_{e;0}(\tau)\exp(\eta_{e;i}) \tag{11.11}$$

Where $h_{e;0}$ is a cause-specific baseline hazard and $\eta_{e;i}$ is the cause-specific coefficient for observation i.

The coefficients are fit by estimating $\hat{\eta}_{e;i}$ with the cause-specific partial likelihood:

$$\mathcal{L}(\boldsymbol{\beta}_e) = \prod_{k:t_{e;(k)}} \left(\frac{\exp(\eta_{e;k})}{\sum_{j \in \mathcal{R}_{t_{e;(k)}}} \exp(\eta_{e;j})} \right) \tag{11.12}$$

This is identical to the single-event partial likelihood in (11.8), with the only difference being that the product and sum are over the ordered event times for cause e. The risk-set definition is unaltered such that $\mathcal{R}_{t_{e;(k)}}$ is the set of observations that have not experienced *any* cause by $t_{e;(k)}$, which is the kth observed event time for cause e. Using the same logic, the Breslow estimator follows from (11.9) by again taking the sums over $t_{e;(k)}$ instead of $t_{(k)}$:

$$\hat{H}_{Bres;e}(\tau) = \sum_{t_{e;(k)} \leq \tau} \frac{d_{t_{e;(k)}}}{\sum_{j \in \mathcal{R}_{t_{e;(k)}}} \exp(\hat{\eta}_{e;j})} \tag{11.13}$$

Here the baseline hazard is estimated using the Breslow estimator for the Cox semi-parametric cause-specific model. One could instead assume a parametric form for the baseline hazard and create a fully-parametric cause-specific model, where again the likelihood is calculated for each independent cause separately:

$$\mathcal{L}(\boldsymbol{\theta}_e) = \prod_{i=1}^{n} h_e(t_i|\mathbf{x}_i, \boldsymbol{\theta}_e)^{\delta_{i;e}} S_e(t_i|\mathbf{x}_i, \boldsymbol{\theta}_e)$$

where $\delta_{i;e}$ is the event indicator for cause e. Note that here $S_e(t_i|\mathbf{x}_i, \boldsymbol{\theta}_e) = \exp(H(t_i|\mathbf{x}_i, \boldsymbol{\theta}_e))$ is the survival for event e, not the all-cause survival, and thus assumes that other causes are treated as censored.

Any PH distribution (Weibull etc.) can then be substituted into h_e and S_e in the same way as in the single-event case, with distribution parameters being estimated separately for each cause e.

Subdistribution PH models

The methods discussed thus far estimate cause-specific hazards (5.3), which represent the instantaneous risk of an individual experiencing the cause of interest, given that they have not yet experienced *any* event. An alternative approach is to model subdistribution hazards, which model the risk of an individual experiencing the cause of interest, given they have not yet experienced the event of interest, but may have experienced a competing event. As will be shown below, the benefit of the subdistribution model is the ability to directly predict the cumulative incidence function (CIF) under a PH model. In this framework, the subdistribution hazard provides a relationship between covariates and the CIF for an event of interest (Austin, Lee, and Fine 2016). Mathematically, the difference between cause-specific and subdistribution hazards comes from the definition of the risk set. As stated above, there is no cause-specific risk set definition. However, the subdistribution risk set is defined as:

$$\mathcal{R}_{e;\tau} := \{i : t_i \geq \tau \vee [e_i \neq e \wedge \delta_i = 1]\} \tag{11.14}$$

Observe that in this definition the left-hand side is the same as the standard risk set definition (4.9) and the right-hand side additionally includes those that have experienced a different event already. Anyone that has been censored ($e = e_0$) before τ is removed from the risk set.

Using this latter definition of the risk set gives rise to the subdistribution hazard function for cause e:

$$h_e^{SD}(\tau) = \lim_{\mathrm{d}\tau \to 0} \frac{P(\tau \leq Y \leq \tau + \mathrm{d}\tau, E = e \mid Y \geq \tau \vee (E \neq e \wedge \Delta = 1))}{\mathrm{d}\tau} \tag{11.15}$$

Subsituting (11.15) into the baseline hazard of the Cox model (11.3) yields the Fine-Gray subhazard model for the eth cause (Fine and Gray 1999):

$$h_{FG;e}(\tau|\mathbf{x}_i) = h_{e;0}^{SD}(\tau) \exp(\eta_i) \tag{11.16}$$

This model form assumes that the subdistribution hazards are all proportional. Estimating the β coefficients follows from a weighted, subdistribution adaptation of the partial likelihood (11.8) which depends on the subdistribution risk set definition given in (11.14).

$$\mathcal{L}_e(\beta) = \prod_{k:t_{e;(k)}} \left(\frac{\exp(\eta_k)}{\sum_{j \in \mathcal{R}_{e;t_{e;(k)}}} w_{kj} \exp(\eta_j)} \right) \tag{11.17}$$

Where $t_{e;(k)}$ is the ordered event times for observations that experience event e. To account for the updated risk set definition in (11.14), Fine and Gray defined the weights

$$w_{kj} := \frac{\hat{G}_{KM}(t_{e;(k)})}{\hat{G}_{KM}(\min\{t_{e;(k)}, t_j\})}$$

Where $\hat{G}_{KM}$ is the Kaplan-Meier estimator fit on the censoring distribution (Section 4.3). The Fine-Gray model therefore uses inverse probability of censoring weighting (as in many measures Section 7.1.1) in order to compensate for the effect of competing events. Because of the way the subdistribution risk set is defined in (11.14), the denominator of (11.17) is a sum over individuals, $j \in \mathcal{R}_{e;t_{e;(k)}}$, at time $t_{e;(k)}$ who have either yet to experience an event ($t_j \geq t_{e;(k)}$) or experienced a different event ($t_j < t_{e;(k)} \wedge e_i \neq e \wedge \delta_i = 1$). The weighting function handles these cases as follows:

1. If $t_j \geq t_{e;(k)}$ then $w_{kj} = 1$ and thus observations who have not experienced any event contribute the most to the likelihood.
2. If $t_j < t_{e;(k)}$ then $w_{kj} < 1$ as $\hat{G}_{KM}$ is a monotonically decreasing function with w_{kj} continuing to decrease as the distance between t_j and $t_{e;(k)}$ increases. Thus the contribution from observations that have experienced competing events reduces over time.

Whilst modelling the subdistribution can seem unintuitive note that if there is only one event of interest then $w_{kj} = 1$ for all k and j and further $e_i \neq e$ must always be false, meaning (11.14) reduces to the standard risk set definition (4.9) as only the left condition can ever be true and by the same logic the subdistribution hazard reduces to the usual hazard definition. Therefore the standard Cox PH for single events is perfectly recovered.

Instead of interpreting the subdistribution hazards directly, Austin and Fine (2017) recommend interpreting the fitted coefficients via the cause-specific CIF and cumulative hazard forms of (11.16), which can be obtained in the 'usual' way by first integrating to obtain the cumulative hazard form:

$$H_{FG;e}(\tau|\mathbf{x}_i) = H^{SD}_{e;0}(\tau)\exp(\eta_i) \tag{11.18}$$

where $H^{SD}_{e;0}$ is the cause-specific baseline cumulative hazard for cause e. Then using (4.4) to relate the survival and hazard functions and representing this in terms of the CIF:

$$F_{FG;e}(\tau|\mathbf{x}_i) = 1 - \exp(-H^{SD}_{e;0}(\tau))^{\exp(\eta_i)} \tag{11.19}$$

Or more simply:

$$F_{FG;e}(\tau|\mathbf{x}_i) = 1 - (1 - F^{SD}_{e;0}(\tau))^{\exp(\eta_i)} \tag{11.20}$$

where $F^{SD}_{e;0}$ is the cause-specific baseline cumulative incidence function for cause e. The model in (11.20) is fit by estimating the baseline cumulative hazard function and substituting into (11.19). Similarly to how the subdistribution hazard was created, estimation of $\hat{H}^{SD}$ follows by updating (11.9) to use the subdistribution risk set definition and applying the same weighting to compensate for multiple events:

$$\hat{H}^{SD}_{Bres;e}(\tau) = \sum_{t_{e;(k)} \leq \tau} \frac{d_{t_{e;(k)}}}{\sum_{j \in \mathcal{R}_{e;t_{e;(k)}}} w_{kj}\exp(\hat{\eta}_j)} \tag{11.21}$$

Use of the Fine-Gray model has to be carefully considered before model fitting. The subdistribution risk set definition, which includes competing events, treats all causes as non-terminal (even if realistically impossible), which means an observation could be considered at risk of the event of interest even after experiencing a competing event. In practice this is often not the case, especially in medical models where death is often a competing risk. Moreover, combining subdistribution CIF estimates across causes can result in probabilities that exceed 1, which should be impossible as events are mutually exclusive and exhaustive (Austin et al. 2022). In these above cases, estimating and summing cause-specific hazard functions is preferred (Austin et al. 2022).

The Fine-Gray model is most appropriate when only concerned with a single event of interest and all-cause estimates are not of interest. Interpreting coefficient changes via (11.20) is slightly more intuitive as making inferences about the magnitude of effects of covariates on the incidence is more standard in clinical settings (Austin and Fine 2017).

11.3 Accelerated Failure Time

Whilst the proportional hazards models is a powerful model, it often does not represent real-world phenomena well. The accelerated failure time (AFT) model is a popular alternative which models the effect of covariates as 'acceleration factors' that act multiplicatively on time. In contrast to the PH model, AFT models are all fully-parametric. A semi-parametric model has been suggested (Buckley and James 1979) however this is not used in practice as it lacks 'theoretical justification' and is 'not reliable' (Wei 1992). Similarly, whilst its theoretically possible to fit cause-specific AFT models for the competing risks setting, this does not appear common in practice.

11.3.1 Understanding acceleration

Moving from a PH to AFT framework can be confusing, so to elucidate this further, take the following example adapted from Kleinbaum and Klein (1996). Consider a model that compares the lifespans of humans and small dogs, which on average are said to age five times faster than humans. If we take this heuristic to be true, then one would expect the probability of a dog surviving to age τ to be approximately the same probability of a human surviving to age 5τ:

$$S_{dog}(\tau) = S_{human}(5\tau) \tag{11.22}$$

This is visualised in Figure 11.1, alongside the AFT curve (red) which 'correctly' models the relationship between humans and dogs, the analogous PH curve (blue) is also plotted. The PH curve assumes the difference in risk between humans and dogs remains the same over time, specifically that at a given time, $S_{dog}(\tau) = S_{human}(\tau)^5$ (from (11.4)).

More generally, the accelerated failure time model estimates survival functions as

$$S_{AFT}(\tau|\mathbf{x}_i) = S_0(\tau e^{-\eta_i}) \tag{11.23}$$

with respective hazard function

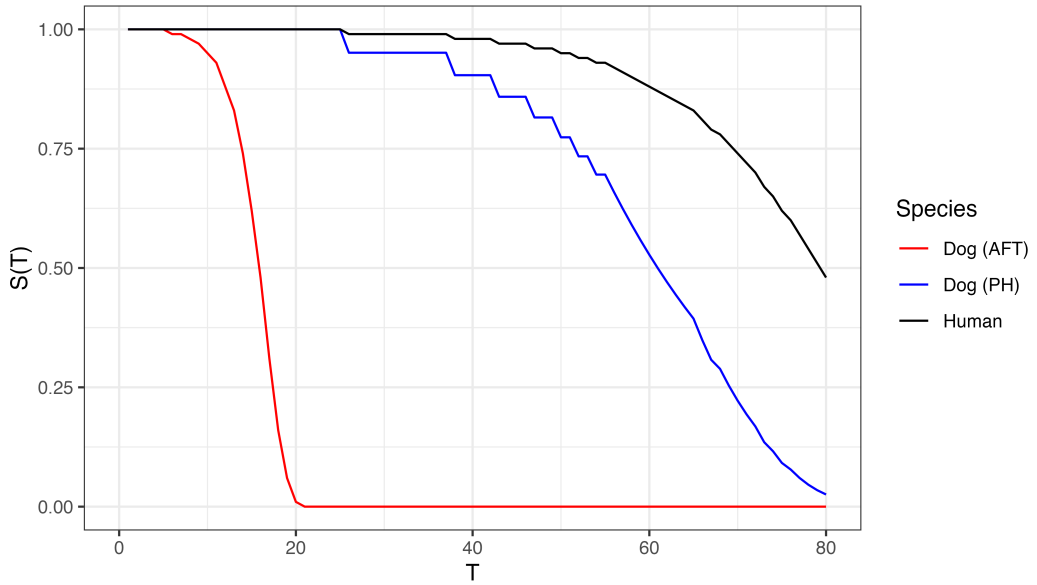


Figure 11.1: Comparing human (black) and dog lifespans where the latter is modelled using an AFT model (red) versus a PH model (blue). Clearly the AFT model is (sadly) a better reflection of reality. Human lifespan modelled with Gompertz(0.09, 0.00005).

$$h_{AFT}(\tau|\mathbf{x}_i) = e^{-\eta_i} h_0(\tau e^{-\eta_i}) \tag{11.24}$$

Note three key differences compared to the PH model. Firstly, $\exp(-\eta_i)$ is modelled instead of $\exp(\eta_i)$, hence in a PH model $h_{PH}(\eta_i+1) > h_{PH}(\eta_i)$ whereas in an AFT model $h_{AFT}(\eta_i + 1) < h_{AFT}(\eta_i)$. Secondly, the baseline risk now clearly depends on both time and the linear predictor. Thirdly, an increase in a covariate results in a multiplicative increase over *time* compared to the PH model in which the hazard is increased – this is often seen as more intuitive to understand than hazard ratio, especially to clinicians who may be more interested in survival times and not abstract relative risks.

This third point is visualised in Figure 11.2 in which a covariate is increase by log(2). The left panel shows that the estimated hazard function from a PH model is double the baseline at all time points – the multiplicative effect is seen on the y-axis (risk). In contrast, the right panel shows how the survival function from an AFT model decreases at double the speed to the baseline – the multiplicative effects is now on the x-axis (time). Another way to demonstrate this effect is through the log-linear form of the accelerated failure time model:

$$\log(t_i) = \mu + \eta_i + \sigma\epsilon_i \tag{11.25}$$

where σ is a scale parameter, ϵ_i is an error term, and μ is an intercept. Now consider the difference in t_i when η_i is increased by one (assuming just one covariate):

$$\log(t_i|x_i + 1) - \log(t_i|x_i) = (\mu + \beta(x_i + 1) + \sigma\epsilon_i) - (\mu + \beta x_i + \sigma\epsilon_i) = \beta$$

Taking exponentials

$$\frac{t_i|x_i + 1}{t_i|x_i} = \exp(\beta)$$

Hence increasing a covariate effectively multiplies the survival time by $\exp(\beta)$:

$$t_i|x_i + 1 = e^{\beta} t_i|x_i$$

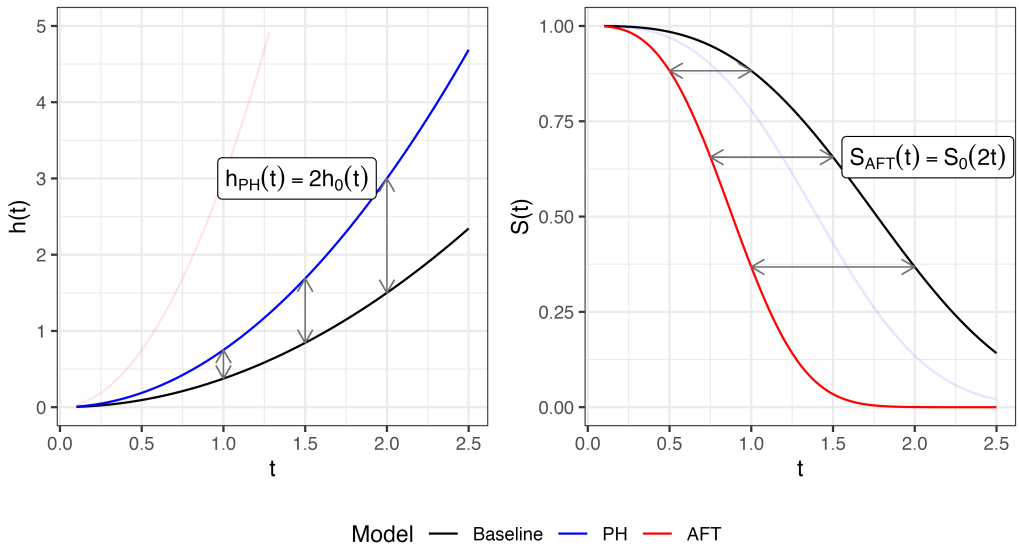


Figure 11.2: Comparing increasing a covariate x_i between PH (left) and AFT (right) models. An increase of $x_i + \log(2)$ multiplies $h(t)$ by $\exp(\log(2)) = 2$ in a PH model. Whereas the result in the AFT is to multiply time t by 2, hence for any t, the AFT model reaches $S(t)$ in half the time as the baseline.

11.3.2 Parametric AFTs

As stated, in practice AFTs are always fully parametric, which means S_0 and h_0 are chosen according to some specific distribution. Common distribution choices include Weibull, Exponential, Log-logistic, and Log-Normal (John D. Kalbfleisch and Prentice 2011; P. Wang, Li, and Reddy 2019). The Weibull distribution (and Exponential as a special case) is unique in that it has both the PH and AFT property (technically a less known representation of Gompertz also has this property). Therefore, selecting an alterative distribution, for example log-logistic or log-normal, can be particular useful if the PH assumption does not hold up. The hazard function of the log-logistic distribution is plotted in Figure 11.3, note the hazard is non-monotonic, allowing non-PH representations to be modelled where the risk of an event may increase before decreasing, or vice versa. When distributions are well-specified and the PH assumption is violated, AFTs can outperform PH alternatives (Patel, Kay, and Rowell 2006; J. Qi 2009; Zare et al. 2015).

As with the PH model, AFT models can be fit using maximum likelihood estimation of the full-likelihood by plugging in distribution defining functions into (11.23) and (11.23) and

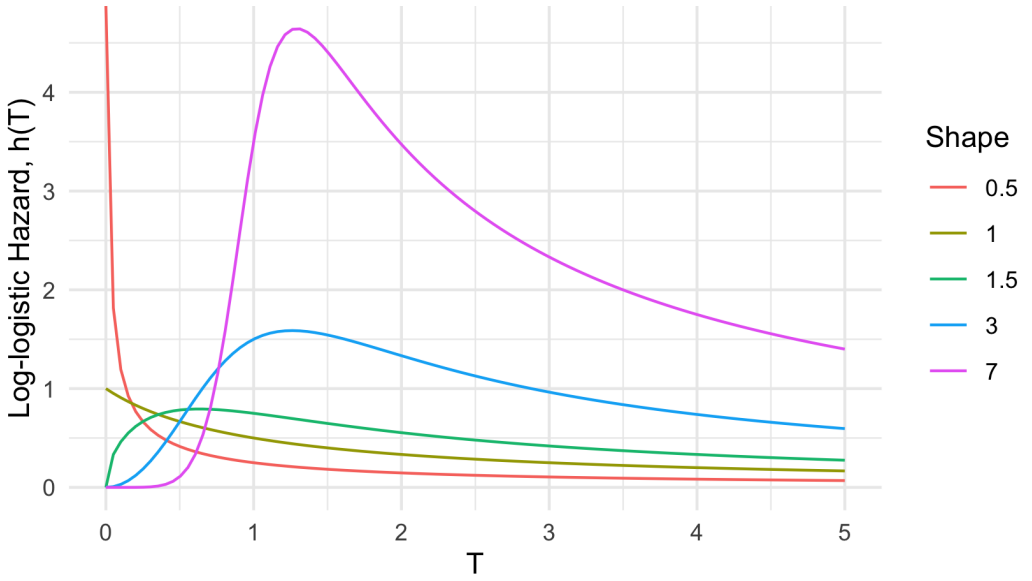


Figure 11.3: Log-logistic hazard curves with a fixed scale parameter of 1 and a changing shape parameter. x-axis is time and y-axis is the log-logistic hazard as a function of time.

likelihoods defined in (Section 4.6.1). Using Exponential this time as an example (the maths is a bit more friendly), first recall that if $X \sim \text{Exp}(\lambda)$ then $h_X(\tau) = \lambda$ and $S_X(\tau) = \exp(-\lambda\tau)$. Then:

$$h_{ExpAFT}(\tau|\mathbf{x}_i) = \lambda e^{-\eta_i}$$

and

$$S_{ExpAFT}(\tau|\mathbf{x}_i) = \exp(-\lambda\tau e^{-\eta_i})$$

Giving the ExpAFT likelihood (Section 4.6.1):

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= \prod_{i=1}^{n} h_Y(t_i|\boldsymbol{\theta})^{\delta_i} S_Y(t_i|\boldsymbol{\theta}) \\ &= \prod_{i=1}^{n} \left(\lambda e^{-\eta_i}\right)^{\delta_i} \left(\exp(-\lambda t_i e^{-\eta_i})\right) \\ &= \prod_{i=1}^{n} \lambda^{\delta_i} \exp(-\lambda t_i e^{-\eta_i} - \delta_i \eta_i)\end{aligned}$$

with log-likelihood

$$\begin{aligned}\updownarrow(\boldsymbol{\theta}) &= \sum_{i=1}^{n} \log(\lambda^{\delta_i} \exp(-\lambda t_i e^{-\eta_i} - \delta_i \eta_i)) \\ &= \sum_{i=1}^{n} \delta_i \log(\lambda) - \lambda t_i e^{-\eta_i} - \delta_i \eta_i\end{aligned}$$

Likelihoods can also be derived using the log-linear form in (11.25) however these are beyond the scope of this book. As before, extensions to other censoring types and truncation follows by specifying the correct likelihood form from Section 4.6.1.

11.4 Proportional Odds

Proportional odds models (Bennett 1983) are the final of the three major linear model classes in survival analysis. In contrast to the PH and AFT models just discussed, proportional models are rarely, if ever, used on their own to make inferences about underlying data or as predictive models. Instead they are more commonly found as components within neural networks (Chapter 15) or in flexible parametric models (discussed next). Therefore this section very briefly describes the motivation for the model and its key properties.

As the name may suggest, the proportional odds model is analogous to the proportional hazards model except with the goal of modelling odds instead of hazards. For a given time τ, the odds of an event happening at **at** τ are

$$O(\tau) = \frac{p(\tau)}{1 - p(\tau)}$$

Where $p(\tau)$ is the probability of the event happening at τ. Of course as has been seen throughout this book the general interest is centered around the survival probability and therefore a more relevant quantity is the odds of the event not happening **before** τ:

$$O(\tau) = \frac{S(\tau)}{1 - S(\tau)} = \frac{S(\tau)}{F(\tau)}$$

By considering the same functional form as in the proportional hazards model, the proportional odds model follows analogously, subsituting odds in place of the hazards:

$$O_{PO}(\tau|\mathbf{x}_i) = O_0(\tau) \exp(\eta_i)$$

where O_0 is the baseline odds.

Clearly by the same logic as in the proportional hazads model, this model assumes $O(\tau|\mathbf{x}_i) \propto \eta_i$ and that a unit increase in a covariate is a multiplicative increase in the odds of survival after a given time. Perhaps the most useful feature of the model is convergence of hazard functions, which states $h_i(\tau)/h_0(\tau) \to 1$ as $\tau \to \infty$ (Kirmani and Gupta 2001). To see this note that the PO model can be represented in terms of the hazard function via (Collett 2014)

$$h_{PO}(\tau|\mathbf{x}_i) = h_0(\tau)\left(1 - \frac{S_0(\tau)}{(\exp(\eta_i) - 1)^{-1} + S_0(\tau)}\right)$$

Dividing by the baseline hazard yields

$$\frac{h_{PO}(\tau|\mathbf{x}_i)}{h_0(\tau)} = \left(1 - \frac{S_0(\tau)}{(\exp(\eta_i) - 1)^{-1} + S_0(\tau)}\right) = \frac{(\exp(\eta_i) - 1)^{-1}}{(\exp(\eta_i) - 1)^{-1} + S_0(\tau)}$$

Simplifying gives $(1 + S_0(\tau)(\exp(\eta_i) - 1))^{-1}$. As $S_0(\tau) \to 0$ as $\tau \to \infty$ it follows that the hazard ratio tends to 1 as τ increases. In practice, this means that for any individual, their risk converges to the average risk over a long period of time. Put differently, any two groups compared over the time will have the same risk of event after enough time has passed. This is clearly an accurate assumption in several contexts, most commonly any medical context in which death is the event of interest.

There is no simple closed form expression for the partial likelihood of a semi-parametric proportional odds model and hence in practice a Log-logistic distribution is usually assumed for the baseline odds and the model is fit by maximum likelihood estimation on the full likelihood (Bennett 1983), discussed further in the next section.

11.5 Flexible Parametric Models

Royston-Parmar flexible parametric models (Patrick Royston and Parmar 2002) extend PH and PO models by estimating the baseline hazard with natural cubic splines. The model was designed to keep the form of the PH or PO methods but without being forced to estimate a misleading baseline hazard (for semi-parametric models) or missspecifying the survival distribution (for fully-parametric models). This is achieved by fitting natural cubic splines in place of the baseline hazard.

The crux of the method is to use splines to model time on a log-scale and to either estimate the log cumulative Hazard for PH models,or the log Odds for PO models. The flexible PH model assumes a Weibull For the flexible PH model, a Weibull distribution is the basis for the baseline distribution and a Log-logistic distribution for the baseline odds in the flexible PO model. The exact derivation of the model requires a lot of mathematical exposition which is not included here, a very good summary of the model is given in Collett (2014). Instead, below is the model in its full form with an explanation of the variables and figures that demonstrate cubic splines.

The flexible parametric Royston-Parmar (RP) proportional hazards and proportional odds model are respectively defined by

$$\log H_{RP}(\tau|\mathbf{x}_i) = s(\tau|\boldsymbol{\gamma}, \boldsymbol{k}) + \eta_i \tag{11.26}$$

$$\log O_{RP}(\tau|\mathbf{x}_i) = s(\tau|\boldsymbol{\gamma}, \boldsymbol{k}) + \eta_i \tag{11.27}$$

where $\boldsymbol{\gamma}$ are spline coefficients to be estimated by maximum likelihood estimation, $\boldsymbol{k}$ are the positions of K knots, and s is the restricted cubic spline function in log time defined by

$$s(\tau|\boldsymbol{\gamma}, \boldsymbol{k}) = \gamma_0 + \gamma_1 \log(\tau) + \gamma_2 \nu_1(\log(\tau)) + ... + \gamma_K \nu_K(\log(\tau))$$

ν_j is the basis function at knot k_j defined by

$$\nu_j(x) = (x - k_j)_+^3 - \lambda_j (x - k_{min})_+^3 - (1 - \lambda_j)(x - k_{max})_+^3$$

where

$$\lambda_j = \frac{k_{max} - k_j}{k_{max} - k_{min}}$$

and $(x - y)_+ = \max\{0, (x - y)\}$ for any x, y, and where k_{min} and k_{max} are the boundaries of the cubic spline, meaning the curve is linear when $\log(t) < k_{min}$ or $\log(t) > k_{max}$.

To see how the proportional hazards RP model relates to the Weibull distribution, first we integrate and take logs of the Weibull-PH hazard function from equation (11.10):

$$\log(H_{WeibullPH}(\tau|\mathbf{x}_i)) = \log\left((\lambda\tau^\gamma)\exp(\eta_i)\right) = \log(\lambda) + \gamma\log(\tau) + \eta_i$$

Setting $\gamma_0 = \log(\lambda)$ and $\gamma_1 = \gamma$ yields (11.26) when there are no knots. Analogous results can be shown between (11.27) and the Log-logistic distribution.

To fit the model, the number and position of knots are theoretically tunable, although Royston and Parmar advised against tuning and suggest often only one internal knot is required and the placement of the knot does not make a significant difference to performance (Patrick Royston and Parmar 2002). Increasing knots can increase model overfitting however Bower et al. (2019) showed up to seven knots does not significantly increase model bias. The model's primary advantage is it's flexibility to model non-linear effects and can also be extended to time-dependent covariates. Moreover, the model can be fit via maximum likelihood estimation and thus many standard off-shelf routines for estimating a smooth survival time function. Despite advantages, the model appears to be limited in common use which makes it difficult to verify the model's utility across different contexts (Ng et al. 2018).

In the same manner as other proportional hazards models, the Royston-Parmar model can be extended to competing risks by modelling cause-specific hazards, considering only one event at a time and censoring competing events (Hinchliffe and Lambert 2013).

11.6 Improving classical models

A number of model-agnostic algorithms have been created to improve a model's predictive ability. When applied to classical algorithms, these methods can be used to create powerful models that outperform other machine learning. As each could be the subject of a whole book, this section remains brief and just covers the general overview. These are split into methods for:

1. modelling non-linear data effects;

2. reducing the number of variables in a dataset; and
3. combining predictions from multiple models.

11.6.1 Non-linear effects

One of the major limitations of traditional statistical models is the assumption of a rigid, linear relationship between covariates and outcomes. At first one might view the Cox model as non-linear due to the presence of the exponential function. However, the linearity becomes clear when the model is equivalently expressed as:

$$\log\left(\frac{h_i(\tau|\mathbf{x}_i)}{h_0(\tau)}\right) = x_1\beta_1 + ...x_p\beta_p$$

Consider modelling the time until disease progression with covariates age (continuous) and treatment (binary):

$$\log\left(\frac{h_i(\tau|\mathbf{x}_i)}{h_0(\tau)}\right) = x_{age}\beta_{age}x_{trt}\beta_{trt}$$

In this form, increasing age from 1 to 21 or 81 to 101 has the same effect on the log hazard ratio; this is clearly not realistic. There are many approaches to relaxing linearity; these are discussed extensively elsewhere and not repeated here, we recommend (James et al. 2013) which covers non-linear modelling in detail.

In brief, PH and AFT can be extended to *generalised additive models* (GAM) in the 'usual' way. For example for the Cox model,

$$\log\left(\frac{h_i(\tau|\mathbf{x}_i)}{h_0(\tau)}\right) = f_1(x_1) + ... + f_p(x_p)$$

where each f_j is a smooth, possibly non-linear function of its covariate. If all f_j are the identity function then this reduces to the standard Cox model. The functions f_j are often chosen to be natural splines, but step functions, polynomial bases, or any other transformation can also be used. The Royston-Parmar model (Section 11.5) is a special case of a GAM where splines are used to model the baseline hazard.

11.6.2 Dimension reduction and feature selection

In a predictive modelling problem, only a small subset of variables in datasets tend to be relevant for correctly predicting the outcome. Other variables are either redundant – they provide no more information than their counterparts – or irrelevant – they do not influence the outcome. In these cases, using all variables for modelling results in worse interpretability, increased computational complexity, and often inferior model performance as model's tend to overfit the training data and generalise poorly to new data.

To improve model performance, one can therefore 'help' models by applying feature selection methods to reduce the number of variables in the dataset. Feature selection is often grouped into three categories: 1) filters; 2) wrappers; and 3) embedded methods. Wrappers fit multiple models on subsets of variables and select the best performing subsets, this is often computationally infeasible in the context of very large datasets, and as such have less relevance in survival analysis which often tackles very high-dimensional datasets such as

genomic datasets and detailed time-series economic data. This section only looks at methods specific to survival analysis and does not consider general algorithms such as PCA, see further reading below for suggested additional reading that covers these areas.

Embedded methods

Embedded methods refers to those that are incorporated during model fitting. The vast majority of machine learning models incorporate embedded methods and thus reduce a dataset's size as part of the training process. In contrast, classical models do not apply any form of feature selection and therefore can perform poorly when there is a large number of variables in a dataset. One model that straddles the line of 'classic' and 'machine learning' methodology is the elastic Cox model, which incorporates the 'lasso' and 'ridge' regularization methods. Given a generic learning algorithm, lasso and ridge regularization respectively constrain the model coefficients subject to the $\updownarrow^{\infty}$-norm and $\updownarrow^{\in}$-norm respectively. In survival analysis, the Lasso-Cox (Tibshirani 1997) model fits the usual Cox model in (11.3) by estimating $\boldsymbol{\beta}$ as

$$\hat{\boldsymbol{\beta}} = \arg\max \mathcal{L}(\boldsymbol{\beta}); \text{ subject to } \sum |\beta_j| \leq \gamma$$

where $\mathcal{L}$ is the likelihood defined in (11.8) and $\gamma > 0$ is a hyper-parameter.

In contrast, the Ridge-Cox model estimates $\boldsymbol{\beta}$ as

$$\hat{\boldsymbol{\beta}} = \arg\max \mathcal{L}(\boldsymbol{\beta}); \text{ subject to } \sum \beta_j^2 \leq \gamma$$

Ridge and lasso are both shrinkage methods, which are used to reduce model variance and overfitting, especially in the context of multi-collinearity. However, the $\updownarrow^1$ norm in Lasso regression can also shrink coefficients to zero and thus performs variable selection as well. It is therefore possible to incorporate a feature selection method first and then pass the results to a Ridge-Cox model – though experiments have shown Ridge-Cox on its own is already a powerful tool (Spooner et al. 2020). Alternatively, as with many machine learning algorithms, deciding between lasso and ridge can be performed in empirical benchmark experiments by using elastic net (N. Simon et al. 2011; Zou and Hastie 2005), which is a convex combination of $\updownarrow^1$ and $\updownarrow^2$ penalties such that $\boldsymbol{\beta}$ is estimated as

$$\hat{\boldsymbol{\beta}} = \arg\max \mathcal{L}(\boldsymbol{\beta}); \text{ subject to } \alpha \sum |\beta_j| + (1-\alpha) \sum \beta_j^2 \leq \gamma$$

where α is a hyper-parameter to be tuned.

Filter methods

Filter methods are a two-step process that score features according to some metric and then select either a given number of top-performing features (i.e., have the best score) or those where the score exceeds some threshold. Once again determining the number of features to select, or the threshold to exceed, can be performed via hyperparameter optimisation. Bommert et al. (2021) compared 14 filter methods for high-dimensional survival data by extending existing methods, making use of tools seen throughout this book including non-parametric estimators, martingale residuals, and inverse probability of censoring weighting. However, the authors found that the method that outperformed all others was a simple variance filter:

$$S(\mathbf{x}_{;j}) = \text{Var}(\mathbf{x}_{;j})$$

where $\mathbf{x}_{;j}$ is the jth variable in the data and S is the resulting score. The filter measures the amount of variance in the feature and removes features that have little variation, as these do not significantly impact the outcome.

Another common filter method is to train another model and make use of its embedded feature selection and pass these results to a simpler classical model. This allows a classical model to be trained on relevant features without loss to interpretability or performance. Common choices for models to use in the first step of th pipeline include random forests (Chapter 12) and gradient boosting machines (Chapter 14). One could also use a Cox model by fitting the model then significance testing the features and removing those that are not significant. However this does not appear to be an improvement over using an embedded regularization method (Spooner et al. 2020).

11.6.3 Ensemble methods

Ensemble methods fit multiple models and combine the result into a single prediction. Common ensemble methods include boosting, bagging, and stacking. These are briefly explained below and can be applied to any model, whether machine learning or a simple linear model. Note that nested cross-validation should be used when fitting any of the models below in order to ensure no data is 'leaked' between training and predictions (Section 3.5).

Bagging and averaging

The simplest ensemble method is to fit multiple models on the same data and make predictions by taking an average over the individual model predictions. The average over predictions could be uniform, weighted with weights selected through expert knowledge, or weighted with weights optimised as hyper-parameters. Ensemble methods perform best when there is high variance between each model as each then captures unique information about the underlying data. Therefore ensemble averaging is more common after first subsetting the training data and training each model on a different subset.

Whilst increasing variance is beneficial, too much variance may result in worse predictions. Hence, bagging (Bootstrap AGGregatING) is a common approach to increase variance without losing predictive accuracy. Bootstrapping is the process of sampling data with replacement, meaning the rows may be duplicated in each subset – this is discussed further in Chapter 12. After sampling the process is the same with predictions made and averaged.

Model-based boosting

Model-based boosting iteratively fits models such that the first model is trained and tested as usual then subsequent models are fit on the same training data but with the purpose of predicting the residuals (i.e., losses) from the previous model. The result is a gradual and iterative increase in improvement as each model captures patterns missed previously. In a survival analysis context, suitable residuals can include many of the losses discussed in Part II, with the choice largely dependent on the task of interest. In contrast to the boosting methods discussed in Chapter 14, model-based boosting is a generic process for fitting iterative models, however it is unlikely a generic boosting pipeline will outperform the specific algorithms introduced in Chapter 14.

Stacking

When stacking is used, multiple models are fit on the same training data, and their predictions are used as training data to a meta-model. The meta-model uses the model predictions to predict the target from the original training data. Fitting a meta-model, often a simple linear model, results in fitted coefficients that weight the input models according to how close or far they were from the truth. Effectively this results in a weighted average of predictions when a linear meta-model is used, where the weights are determined by model accuracy.

11.7 Conclusion

Key takeaways

- Classical statistical models broadly fall into three categories: non-parametric estimators, semi-parametric models that include a baseline estimate, and fully-parametric estimates. Each has its own advantages and disadvantages.
- Non-parametric estimators are used throughout survival analysis with the Kaplan-Meier estimator being a very common component in many machine learning models.
- Proportional hazards and accelerated failure time models encode different assumptions but both are powerful tools for learning patterns from data and even in prediction problems.
- The boundary between machine learning and classical statistical models is fuzzy, and classical survival models can outperform machine learning alternatives.

Limitations

- Several classical models can be extended to time-dependent covariates however these are not well-developed for predictive problems.
- Classical models encode assumptions that are rarely met in practice, for example the proportional hazards assumption. However, even if assumptions are violated, these could still generalise well to new data and are therefore worth including in benchmark experiments.
- Classical models perform badly on high-dimensional data without some form of pre-processing, however this is relatively simple with modern off-shelf software such as **sklearn** and **mlr3**.

Further reading

- To learn more about hazard ratios from Cox models and complexities in interpretation, we recommend Sashegyi and Ferry (2017) and Spruance et al. (2004).
- Collett (2014) and J. D. Kalbfleisch and Prentice (1973) both provide comprehensive reading for classical statistical models. The former is slightly less technical and covers extensions to multiple settings.
- For more abstract feature selection algorithms that can be applied to any data (survival or otherwise), see Chandrashekar and Sahin (2014) and Guyon and Elisseeff (2003).

12

Random Forests

TODO (150-200 WORDS)

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

Random forests are a composite (or ensemble) algorithm built by fitting many simpler component models, *decision trees*, and then averaging the results of predictions from these trees. Due to in-built variable importance properties, random forests are commonly used in high-dimensional settings when the number of variables in a dataset far exceeds the number of rows. High-dimensional datasets are very common in survival analysis, especially when considering omics, genetic and financial data. It is therefore no surprise that *random survival forests*, remain a popular and well-performing model in the survival setting.

12.1 Random Forests for Regression

Training of decision trees can include a large number of hyper-parameters and different training steps including 'growing' and subsequently 'pruning'. However, when utilised in random forests, many of these parameters and steps can be safely ignored, hence this section only focuses on the components that primarily influence the resulting random forest. This section will start by discussing decision trees and will then introduce the *bagging* algorithm used to create random forests.

12.1.1 Decision Trees

Decision trees are a (relatively) simple machine learning model that are comparatively easy to implement in software and are highly interpretable. The decision tree algorithm takes an input, a dataset, selects a variable that is used to partition the data according to some *splitting rule* into distinct non-overlapping datasets or *nodes* or *leaves*, and repeats this step for the resulting partitions, or *branches*, until some criterion has been reached. The final nodes are referred to as *terminal nodes*.

By example, (Figure 12.1) demonstrates a decision tree predicting the price that a car sells for in India (price in thousands of dollars). The dataset includes as variables the registration year, kilometers driven, fuel type (petrol or automatic), seller type (individual or dealer), transmission type (manual or automatic), and number of owners. The decision tree was

trained with a maximum depth of 2 (the number of rows excluding the top), and it can be seen that with this restriction only the transmission type, registration year, and fuel type were selected variables. During training, the algorithm identified that the first optimal variable to split the data was transmission type, partitioning the data into manual and automatic cars. Manual cars are further subset by registration year whereas automatic cars are split by fuel type. It can also be seen how the average sale price (top value in each leaf) diverges between leaves as the tree splits – the average sale prices in the final leaves are the terminal node predictions.

The graphic highlights several core features of decision trees:

1. They can model non-linear and interaction effects: The hierarchical structure allows for complex interactions between variables with some variables being used to separate all observations (transmission) and others only applied to subsets (year and fuel);
2. They are highly interpretable: it is easy to visualise the tree and see how predictions are made;
3. They perform variable selection: not all variables were used to train the model.

To understand how random forests work, it is worth looking a bit more into the most important components of decision trees: splitting rules, stopping rules, and terminal node predictions.

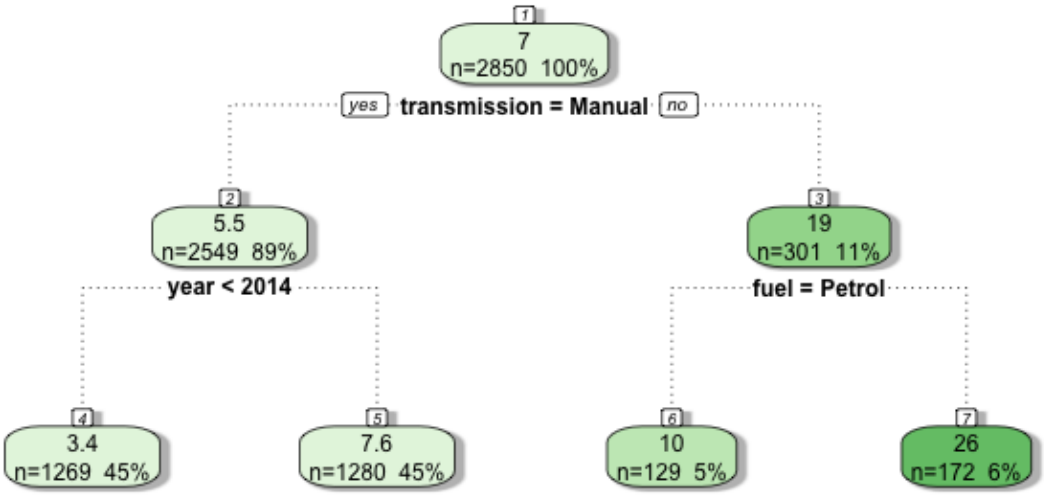


Figure 12.1: Predicting the price a vehicle is sold for in India using a regression tree, dataset from kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho. Rounded rectangles are leaves, which indicate the variable that is being split. Edges are branches, which indicate the cut-off at which the variable is split. Variables are car transmission type (manual or automatic), fuel type (petrol or diesel) and registration year. The number at the top of each leaf is the average selling price in thousands of dollars for all observations in that leaf. The numbers at the bottom of each leaf are the number of observations in the leaf, and the proportion of data contained in the leaf.

Splitting and Stopping Rules

Precisely how the data partitions/splits are derived and which variables are utilised is determined by the *splitting rule*. The goal in each partition is to find two resulting leaves/nodes that have the greatest difference between them and thus the maximal homogeneity within each leaf, hence with each split, the data in each node should become increasingly similar. The splitting rule provides a way to measure the homogeneity within the resulting nodes. In regression, the most common splitting rule is to select a variable and cut-off (a threshold on the variable at which to separate observations) that minimises the mean squared error in the two potential resulting leaves.

For all decision tree and random forest algorithms going forward, let L denote some leaf, then let L_{xy}, L_x, L_y respectively be the set of observations, features, and outcomes in leaf L. Let $L_{y;i}$ be the ith outcome in L_y and finally let $\bar{y}_L = \frac{1}{|L_y|}\sum_{i=1}^{|L_y|} L_{y;i}$ be the mean outcome in leaf L.

Let $j = 1, \ldots, p$ be the index of features and let c_j be a possible cutoff value for feature $\mathbf{x}_{;j}$. Define

$$L^a_{xy}(j, c_j) := \{(\mathbf{x}_i, y_i) | x_{i;j} < c_j, i = 1, ..., n\}$$
$$L^b_{xy}(j, c_j) := \{(\mathbf{x}_i, y_i) | x_{i;j} \geq c_j, i = 1, ..., n\}$$

as the two leaves containing the set of observations resulting from partitioning variable j at cutoff c_j. To simplify equations let L^a, L^b be shorthands for $L^a(j, c_j)$ and $L^b(j, c_j)$. Then a split is determined by finding the arguments, $(j^*, c^*_{j^*})$ that minimise the residual sum of squares across both leaves (James et al. 2013),

$$(j^*, c^*_{j^*}) = \arg\min_{j, c_j} \sum_{y \in L^a_y} (y - \bar{y}_{L^a})^2 + \sum_{y \in L^b_y} (y - \bar{y}_{L^b})^2 \tag{12.1}$$

This method is repeated from the first leaf to the last such that observations are included in a given leaf L if they satisfy all conditions from all previous branches (splits); features may be considered multiple times in the growing process allowing complex interaction effects to be captured.

Leaves are repeatedly split until a *stopping rule* has been triggered – a criterion that tells the algorithm to stop partitioning data. The stopping rule is usually a condition on the number of observations in each leaf such that leaves will continue to be split until some minimum number of observations has been reached in a leaf. Other conditions may be on the depth of the tree (as in Figure 12.1 which is restricted to a maximum depth of 2), which corresponds to the number of levels of splitting. Stopping rules are often used together, for example by setting a maximum tree depth *and* determining a minimum number of observations per leaf. Deciding the number of minimum observations and/or the maximum depth can be performed with automated hyper-parameter optimisation.

Terminal Node Predictions

The final major component of decision trees are *terminal node predictions*. As the name suggests, this is the part of the algorithm that determines how to actually make a prediction for a new observation. A prediction is made by 'dropping' the new data 'down' the tree according to the optimal splits that were found during training. The resulting prediction is then a simple baseline statistic computed from the training data that fell into the corresponding node. In regression, this is commonly the sample mean of the training outcome data.

Returning to Figure 12.1, say a new data point is {transmission = Manual, fuel = Diesel, year = 2015}, then in the first split the left branch is taken as 'transmission = Manual', in the second split the right branch is taken as 'year' = 2015 $\geq$ 2014, hence the new data point lands in the second terminal leaf and is predicted to sell for $7,600. The 'fuel' variable is ignored as it is only considered for automatic vehicles. As the final predictions are simple statistics based on training data, all potential predictions can be saved in the original trained model and no complex computations are required in the prediction algorithm.

12.1.2 Random Forests

Decision trees often overfit the training data, hence they have high variance, perform poorly on new data, and are not robust to even small changes in the original training data. Moreover, important variables can end up being ignored as only subsets of dominant variables are selected for splitting.

To counter these problems, *random forests* are designed to improve prediction accuracy and decrease variance. Random forests utilise bootstrap aggregation, or *bagging* (Leo Breiman 1996), to aggregate many decision trees. Bagging is a relatively simple algorithm, as follows:

1. **For** $b = 1, ..., B$:
2. $D_b \leftarrow$ Randomly sample with replacement $\mathcal{D}_{train}$
3. $\hat{g}_b \leftarrow$ Train a decision tree on D_b
4. **end For**
5. **return** $\{\hat{g}_b\}_{b=1}^{B}$

Step 2 is known as *bootstrapping*, which is the process of sampling a dataset *with* replacement – which is in contrast to more standard subsampling where data is sampled *without* replacement. Commonly, the bootstrapped sample size is the same as the original. However, as the same value may be sampled multiple times, on average the resulting data only contains around 63.2% unique observations (Efron and Tibshirani 1997). Randomness is further injected to decorrelate the trees by randomly subsetting the candidates of features to consider at each split of a tree. Therefore, every split of every tree may consider a different subset of variables. This process is repeated for B trees, with the final output being a collection of trained decision trees.

Prediction from a random forest for new data $\mathbf{x}^*$ follows by making predictions from the individual trees and aggregating the results by some function σ, which is usually the sample mean for regression:

$$\hat{g}(\mathbf{x}^*) = \sigma(\hat{g}_1(\mathbf{x}^*), ..., \hat{g}_B(\mathbf{x}^*)) = \frac{1}{B}\sum_{b=1}^{B} \hat{g}_b(\mathbf{x}^*)$$

where $\hat{g}_b(\mathbf{x}^*)$ is the prediction from the bth tree for some new data $\mathbf{x}^*$ and B are the total number of grown trees.

As discussed above, individual decision trees result in predictions with high variance that are not robust to small changes in the underlying data. Random forests decrease this variance by aggregating predictions over a large sample of decorrelated trees, where a high degree of difference between trees is promoted through the use of bootstrapped samples and random candidate feature selection at each split.

Usually many (hundreds or thousands) trees are grown, which makes random forests robust to changes in data and 'confident' about individual predictions. Other advantages include having tunable and meaningful hyper-parameters, including: the number of variables to consider for a single tree, the splitting rule, and the stopping rule. Random forests treat trees as *weak learners*, which are not intended to be individually optimized. Instead, each tree captures a small amount of information about the data, which are combined to form a powerful representation of the dataset.

Whilst random forests are considered a 'black-box', in that one cannot be reasonably expected to inspect thousands of individual trees, variable importance can still be aggregated across trees, for example by counting the frequency a variable was selected across trees, calculating the minimal depth at which a variable was used for splitting, or via permutation based feature importance. Hence the model remains more interpretable than many alternative methods. Finally, random forests are less prone to overfitting and this can be relatively easily controlled by using *early-stopping* methods, for example by continually growing trees until the performance of the model stops improving.

12.2 Random Survival Forests

Unlike other machine learning methods that may require complex changes to underlying algorithms, random forests can be relatively simply adapted to *random survival forests* by updating the splitting rules and terminal node predictions to those that can handle censoring and can make survival predictions. This chapter is therefore focused on outlining different choices of splitting rules and terminal node predictions, which can then be flexibly combined into different models.

12.2.1 Splitting Rules

Survival trees and RSFs have been studied for the past four decades and whilst there are many possible splitting rules (Bou-Hamad, Larocque, and Ben-Ameur 2011), only two broad classes are commonly utilised (as judged by number of available implementations, e.g., Pölsterl (2020); Wright and Ziegler (2017); H. Ishwaran et al. (2011)). The first class rely on hypothesis tests, and primarily the log-rank test, to maximise dissimilarity between splits, the second class utilises likelihood-based measures. The first is discussed in more detail as this is common in practice and is relatively straightforward to implement and understand, moreover it has been demonstrated to outperform other splitting rules (Bou-Hamad, Larocque, and Ben-Ameur 2011). Likelihood rules are more complex and require assumptions that may not be realistic, these are discussed briefly.

Hypothesis Tests

The log-rank test statistic has been widely utilized as a splitting-rule for survival analysis (Ciampi et al. 1986; B. H. Ishwaran et al. 2008; LeBlanc and Crowley 1993; Segal 1988). The log-rank test compares the survival distributions of two groups under the null-hypothesis that both groups have the same underlying risk of (immediate) events, with the hazard function used to compare underlying risk.

Let L^a and L^b be two leaves and let h^a, h^b be the (theoretical, true) hazard functions in the two leaves respectively and let $i \in L$ be a shorthand for the indices of the observations in

leaf L so that $i = 1, \ldots, |L|$. Define:

- $\mathcal{U}_D$, the set of unique event times across the data (in both leaves)
- n^a_τ, the number of observations at risk at τ in leaf a

$$n^a_\tau = \sum_{i \in L^a} \mathbb{I}(t_i \geq \tau)$$

- o^a_τ, the observed number of events in leaf a at τ

$$o^a_\tau = \sum_{i \in L^a} \mathbb{I}(t_i = \tau, \delta_i = 1)$$

- $n_\tau = n^a_\tau + n^b_\tau$, the number of observations at risk at τ in both leaves
- $o_\tau = o^a_\tau + o^b_\tau$, the observed number of events at τ in both leaves

Then, the log-rank hypothesis test is given by $H_0 : h^a = h^b$ with test statistic (Segal 1988),

$$LR(L^a) = \frac{\sum_{\tau \in \mathcal{U}_D} (o^a_\tau - e^a_\tau)}{\sqrt{\sum_{\tau \in \mathcal{U}_D} v^a_\tau}}$$

where:

- e^a_τ is the expected number of events in leaf a at τ

$$e^a_\tau := \frac{n^a_\tau o_\tau}{n_\tau}$$

- v^a_τ is the variance of the number of events in leaf a at τ

$$v^a_\tau := e^a_\tau \Big(\frac{n_\tau - o_\tau}{n_\tau} \Big) \Big(\frac{n_\tau - n^a_\tau}{n_\tau - 1} \Big)$$

These results follow as under the assumption of equal hazards in both leafs, the number of events at each $\tau \in \mathcal{U}_D$ is distributed according to a Hypergeometric distribution. The same statistic results if L^b is instead considered.

The higher the log-rank statistic, the greater the dissimilarity between the two groups (Figure 12.2), thereby making it a sensible splitting rule for survival, moreover it has been shown that it works well for splitting censored data (LeBlanc and Crowley 1993). Additionally, the log-rank test requires no knowledge about the shape of the survival curves or distribution of the outcomes in either group (Bland and Altman 2004), making it ideal for an automated process that requires no user intervention.

The log-rank *score* rule (Hothorn and Lausen 2003) is a standardized version of the log-rank rule that could be considered as a splitting rule, though simulation studies have demonstrated non-significant improvements in predictive performance when comparing the two (B. H. Ishwaran et al. 2008). Alternative dissimiliarity measures and tests have also been suggested as splitting rules, including modified Kolmogorov-Smirnov test and Gehan-Wilcoxon tests (Ciampi et al. 1988). Simulation studies have demonstrated that both of these may have higher power and produce 'better' results than the log-rank statistic (Fleming et al. 1980), however neither appears to be commonly used.

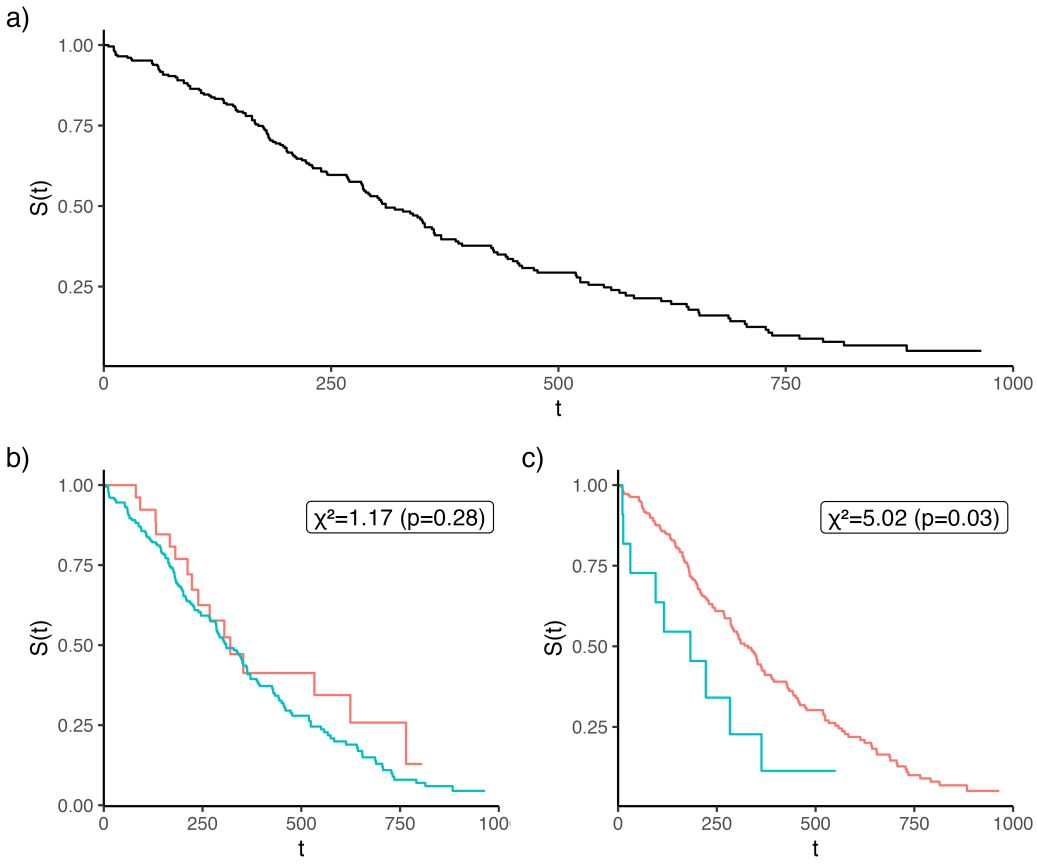


Figure 12.2: Panel (a) is the Kaplan-Meier estimator fit on the complete `lung` dataset from the R package **survival**. (b-c) is the same data stratified according to whether 'age' is greater or less than 50 (panel b) or 75 (panel c). The higher χ^2 statistic (panel c) results in a lower p-value and a greater difference between the stratified Kaplan-Meier curves. Hence splitting age at 75 results in a greater dissimilarity between the resulting branches and thus makes a better choice for splitting the variable.

In a competing risk setting, Gray's test (Gray 1988) can be used instead of the log-rank test, as it compares cumulative incidence functions rather than all-cause hazards. Similarly to the log-rank test, Gray's test also compares survival distributions using hypothesis tests to determine if there are significant differences between the groups, thus making it a suitable option to build competing risk RSFs.

Alternative Splitting Rules

A common alternative to the log-rank test is to instead use *likelihood ratio*, or *deviance*, statistics. When building RSFs, the likelihood-ratio statistic can be used to test if the model fit is improved or worsened with each split, thus providing a way to partition data. However, as discussed in Section 4.6.1, there are many different likelihoods that can be assumed for survival data, and there is no obvious way to determine if one is more sensible than another. Furthermore the choice of likelihood must fit the underlying model assumptions. For example, one could assume the data fits the proportional hazards assumption and in each split one could calculate the likelihood-ratio using the Cox PH partial likelihood. Alternatively, a parametric form could be assumed and a likelihood proposed by LeBlanc and Crowley (1992) may be calculated to test model fit. While potentially useful, these methods are implemented in very few off-shelf software packages, thus empirical comparisons to other splitting rules are lacking.

Other rules have also been studied including comparison of residuals (Therneau, Grambsch, and Fleming 1990), scoring rules (H. Ishwaran and Kogalur 2018), distance metrics (Gordon and Olshen 1985), and concordance metrics (Schmid, Wright, and Ziegler 2016). Experiments have shown different splitting rules may perform better or worse depending on the underlying data (Schmid, Wright, and Ziegler 2016), hence one could even consider treating the splitting rule as a hyper-parameter for tuning. However, if there is a clear goal in prediction, then the choice of splitting rule can be informed by the prediction type. For example, if the goal is to maximise separation, then a log-rank splitting rule to maximise homogeneity in terminal nodes is a natural starting point. Whereas if the goal is to accurately rank observations, then a concordance splitting rule may be optimal.

12.2.2 Terminal Node Prediction

As in the regression setting, the usual strategy for predictions is to create a simple estimate based on the training data that lands in the terminal nodes. However, as seen throughout this book, the choice of estimator in the survival setting depends on the prediction task of interest, which are now considered in turn. First, note that all terminal node predictions can only yield useful results if there are a sufficient number of uncensored observations in each terminal node. Hence, a common RSF stopping rule is the minimum number of *uncensored* observations per leaf, meaning a leaf is not split if that would result in too few uncensored observations in the resulting leaves.

Probabilistic Predictions

Starting with the most common survival prediction type, the algorithm requires a simple estimate for the underlying survival distribution in each of the terminal nodes, which can be estimated using the Kaplan-Meier or Nelson-Aalen methods (Hothorn et al. 2004; B. H. Ishwaran et al. 2008; LeBlanc and Crowley 1993; Segal 1988).

Denote b as a decision tree and $L^{b(h)}$ as the terminal node h in tree b. Then the predicted survival function and cumulative hazard for a new observation $\mathbf{x}^*$ is,

$$\hat{S}_{b(h)}(\tau|\mathbf{x}^*) = \prod_{k:t_{(k)}\leq\tau} 1 - \frac{d_{t_{(k)}}}{n_{t_{(k)}}}, \quad \{k \in L^{b(h)} : \mathbf{x}^* \in L^{b(h)}\} \tag{12.2}$$

$$\hat{H}_{b(h)}(\tau|\mathbf{x}^*) = \sum_{k:t_{(k)}\leq\tau} \frac{d_{t_{(k)}}}{n_{t_{(k)}}}, \quad \{k \in L^{b(h)} : \mathbf{x}^* \in L^{b(h)}\} \tag{12.3}$$

where $t_{(k)}$ is the ordered event times and $d_{t_{(k)}}$ and $n_{t_{(k)}}$ are the observed number of events, and the number of observations at risk, respectively at $t_{(k)}$. See Figure 12.3 for an example using the `lung` dataset (Therneau 2015).

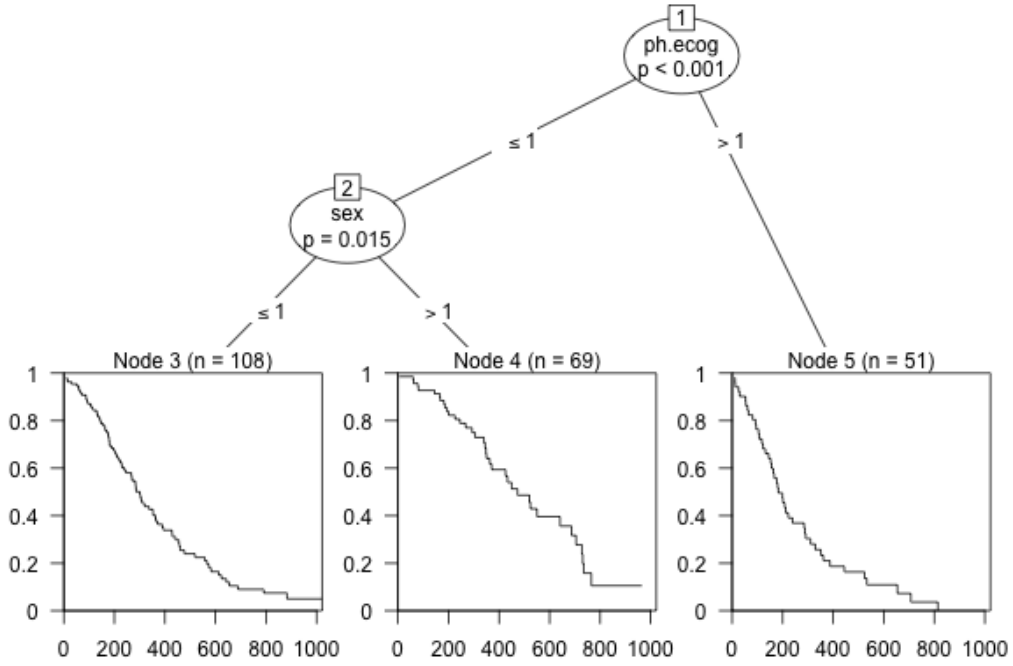


Figure 12.3: Survival tree trained on the `lung` dataset from the R package **survival**. The terminal node predictions are survival curves.

The bootstrapped prediction is the cumulative hazard function or survival function averaged over individual trees. Note that understanding what these bootstrapped functions represents depends on how they are calculated. By definition, a mixture of n distributions with cumulative distribution functions $F_i, i = 1, ..., n$ is given by

$$F(x) = \sum_{i=1}^{n} w_i F_i(x)$$

Subsituting $F = 1 - S$ and noting $\sum w_i = 1$ gives the computation $S(x) = \sum_{i=1}^{n} w_i S_i(x)$, allowing the bootstrapped survival function to exactly represent the mixture distribution averaged over all trees:

$$\hat{S}_{Boot}(\tau|\mathbf{x}^*) = \frac{1}{B}\sum_{b=1}^{B} w_i \hat{S}_b(\tau|\mathbf{x}^*) \tag{12.4}$$

usually with $w_i = 1/B$ where B is the number of trees.

In contrast, if one were to instead substitute $F = 1 - \exp(-H)$, then the mixture distribution depends on a logarithmic function that can only be approximately computed if predicted survival probabilities are close to one, which is an assumption that deteriorates over time. Therefore, to ensure the bootstrapped prediction accurately represents the underlying mixed probability distribution, the bootstrapped cumulative hazard function should be computed as:

$$\hat{H}_{Boot}(\tau|\mathbf{x}^*) = -\log(\hat{S}_{Boot}(\tau|\mathbf{x}^*)) \tag{12.5}$$

Another practical consideration to take into account is how to average the survival probabilities over the decision trees as each individual Kaplan-Meier estimate may have been trained on different time points. This is overcome by recognising that the Kaplan-Meier estimation results in a piece-wise function that can be linearly interpolated between training data. Figure 12.4 demonstrates this process for three decision trees (panel a), where the survival probability is calculated at all possible time points (panels b-c), and the average is computed with linear interpolation added between time-points (panel d).

Extensions to competing risks follow naturally using bootstrapped cause-specific cumulative incidence functions.

Deterministic Predictions

As discussed in Chapter 10, predicting and evaluating survival times is a complex and fairly under-researched area. For RSFs, there is an inclination to estimate survival times based on the mean or median survival times of observations in terminal nodes, however this would lead to biased estimations. Therefore, research has tended to focus on relative risk predictions.

As discussed, relative risks are arbitrary values where only the resulting rank matters when comparing observations. In RSFs, each terminal node should be as homogeneous as possible, hence within a terminal node, the risk between observations should be the same. The most common method to estimate average risk appears to be a transformation from the Nelson-Aalen method (B. H. Ishwaran et al. 2008), which exploits results from counting process to provide a measure of expected mortality (Hosmer Jr, Lemeshow, and May 2011) – the same result is used in the van Houwelingen calibration measure discussed in Section 8.1.2. Given new data, $\mathbf{x}^*$, falling into terminal node $b(h)$, the relative risk prediction is the sum of the predicted cumulative hazard, $\hat{H}_{b(h)}$, computed at each observation's observed outcome time:

$$\phi_{b(h)}(\mathbf{x}^*) = \sum_{i \in \mathcal{U}_O} \hat{H}_{b(h)}(t_i|\mathbf{x}^*)$$

where $\hat{H}_{b(h)}$ is the terminal node prediction as in 12.3. This is interpreted as the number of expected events in $b(h)$ and the assumption is that a terminal node with more expected events is a higher risk group than a node with less expected events. The bootstrapped risk prediction is the sample mean over all trees:

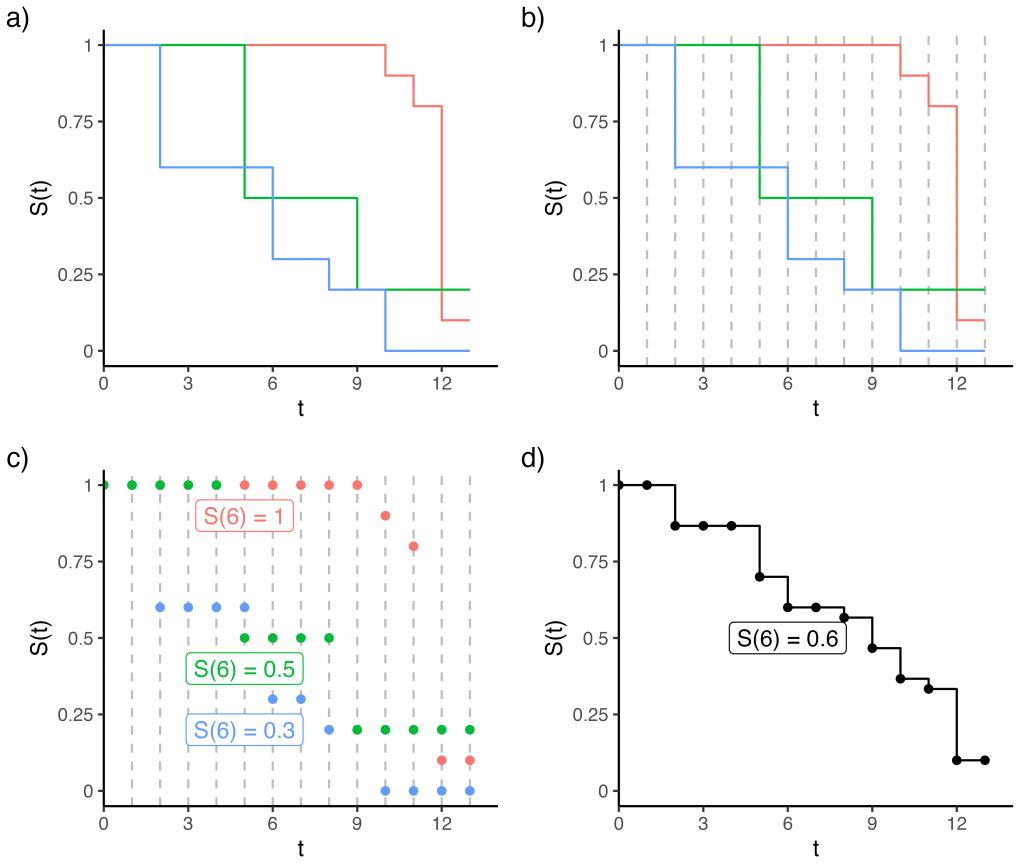


Figure 12.4: Bootstrapping Kaplan-Meier estimators across three decision trees (red, blue, green). Panel a) shows the individual estimates, b) shows the time points to aggregate the trees over, c) is the predicted survival probability from each tree at the desired time points, and d) is the average survival probabilities connected by a step function.

$$\phi_{Boot}(\mathbf{x}^*) = \frac{1}{B}\sum_{b=1}^{B}\phi_{b(h)}(\mathbf{x}^*)$$

More complex methods have also been proposed that are based on the likelihood-based splitting rule and assume a PH model form (H. Ishwaran et al. 2004; LeBlanc and Crowley 1992). However, these do not appear to be in wide-spread usage.

12.3 Conclusion

Key takeaways

- Random forests are a highly flexible algorithm that allow the various components to be adapted and altered without major changes to the underlying algorithm. This allows random survival forests (RSFs) to be readily available 'off-shelf' in many open-source packages;
- RSFs have in-built variable selection methods that mean they tend to perform well on high-dimensional data, routinely outperforming other models Burk et al. (2024);
- Despite having many potential hyper-parameters to tune, all are intuitive and many can even be ignored as sensible defaults exist in most off-shelf software implementations.

Limitations

- Due to the number of trees and the constant bootstrapping procedures, RSFs can be more computationally intensive than other models, though still much less intensive than neural networks and other deep learning methods.
- Despite having some in-built methods for model interpretation, RSFs are still black-boxes that can be difficult to fully interpret.
- With too few trees random forests can have similar limitations to decision trees and with too many random forests can overfit the data. Though most software has sensible defaults to prevent either scenario.

Further reading

- A comprehensive review of random survival forests (RSFs) is provided in Bou-Hamad (2011) (Bou-Hamad, Larocque, and Ben-Ameur 2011), which includes extensions to time-varying covariates and different censoring types.
- The discussion of decision trees omitted many methods for growing and pruning trees, if you are interest in those technical details see L. Breiman et al. (1984).
- RSFs have been shown to perform well in benchmark experiments on high-dimensional data, see Herrmann et al. (2021) and Spooner et al. (2020) for examples.
- This chapter considered the most 'traditional' forms of RSFs. Conditional inference forests are popular in the regression setting and whilst they are under-researched in survival, see Hothorn et al. (2005) for literature on the topic. A more recent method that seems to perform well is the (accelerated) oblique random survival

forest discussed in (**Jaeger2024?**).

13

Support Vector Machines

This chapter introduces support vector machines (SVMs) for regression and then describes the extensions to survival analysis. Regression SVMs extend simple linear methods by estimating flexible, non-linear hyperplanes that minimise the difference between predictions and the truth for individual observations. In survival analysis, SVMs may make survival time or ranking predictions, however there is no current formulation for survival distribution predictions. The chapter begins by discussing survival time SVMs and then ranking models before concluding with a hybrid formulation that combines both model forms. This primarily covers the work of Shivaswamy and Van Belle. SVMs are a powerful method for estimating non-linear relationships in data and have proven to be well-performing models in regression and classification. However, SVMs are less developed in survival analysis and have been shown to perform worse than other models in experiments.

> ⚠ Minor changes expected!
>
> **This page is a work in progress and minor changes will be made over time.**

Support vector machines are a popular class of models in regression and classification settings due to their ability to make accurate predictions for complex high-dimensional, non-linear data. Survival support vector machines (SSVMs) predict continuous responses that can be used as ranking predictions with some formulations that provide survival time interpretations. This chapter starts with SVMs in the regression setting before moving to adaptions for survival analysis.

13.1 SVMs for Regression

In simple linear regression, the aim is to estimate the line $y = \alpha + x\beta_1$ by estimating the α, β_1 coefficients. As the number of coefficients increases, the goal is to instead estimate the *hyperplane*, which divides the higher-dimensional space into two separate parts. To visualize a hyperplane, imagine looking at a room from a birds eye view that has a dividing wall cutting the room into two halves (Figure 13.1). In this view, the room appears to have two dimensions (x=left-right, y=top-bottom) and the divider is a simple line of the form $y = \alpha + x\beta_1$. In reality, this room is actually three dimensional and has a third dimension (z=up-down) and the divider is therefore a hyperplane of the form $y = \alpha + x\beta_1 + z\beta_2$.

Continuing the linear regression example, consider a simple model where the objective is to find the $\boldsymbol{\beta} = (\beta_1 \ \beta_2 \cdots \beta_p)^\top$ coefficients that minimize $\sum_{i=1}^{n}(g(\mathbf{x}_i) - y_i)^2$ where

Figure 13.1: Visualising a hyperplane by viewing a 3D room in two-dimensions with a wall that is now seen as a simple line. When standing in this room, the wall will clearly exist in three dimensional space.

$g(\mathbf{x}_i) = \alpha + \mathbf{x}_i^\top \boldsymbol{\beta}$ and $(\mathbf{X}, \mathbf{y})$ is training data such that $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{y} \in \mathbb{R}^n$. In a higher-dimensional space, a penalty term can be added for variable selection to reduce model complexity, commonly of the form

$$\frac{1}{2}\sum_{i=1}^{n}(g(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2}\|\boldsymbol{\beta}\|^2$$

for some penalty term $\lambda \in \mathbb{R}$. Minimizing this error function effectively minimizes the *average* difference between all predictions and true outcomes, resulting in a hyperplane that represents the best *linear* relationship between coefficients and outcomes.

Similarly to linear regression, support vector machines (SVMs) (Cortes and Vapnik 1995) also fit a hyperplane, g, on given training data, $\mathbf{X}$. However, in SVMs, the goal is to fit a *flexible* (non-linear) hyperplane that minimizes the difference between predictions and the truth for *individual* observations. A core feature of SVMs is that one does not try to fit a hyperplane that makes perfect predictions as this would overfit the training data and is unlikely to perform well on unseen data. Instead, SVMs use a regularized error function, which allows incorrect predictions (errors) for some observations, with the magnitude of error controlled by an $\epsilon > 0$ parameter as well as slack parameters, $\boldsymbol{\xi'} = (\xi'_1\ \xi'_2 \cdots \xi'_n)^\top$ and $\boldsymbol{\xi^*} = (\xi^*_1\ \xi^*_2 \cdots \xi^*_n)^\top$:

$$\begin{aligned} &\min_{\boldsymbol{\beta},\alpha,\boldsymbol{\xi'},\boldsymbol{\xi^*}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}(\xi'_i + \xi^*_i) \\ &\text{subject to} \begin{cases} g(\mathbf{x}_i) & \geq y_i - \epsilon - \xi'_i \\ g(\mathbf{x}_i) & \leq y_i + \epsilon + \xi^*_i \\ \xi'_i, \xi^*_i & \geq 0 \end{cases} \end{aligned} \tag{13.1}$$

$\forall i \in 1, ..., n$ where $g(\mathbf{x}_i) = \alpha + \mathbf{x}_i^\top \boldsymbol{\beta}$ for model weights $\boldsymbol{\beta} \in \mathbb{R}^p$ and $\alpha \in \mathbb{R}$ and the same training data $(\mathbf{X}, \mathbf{y})$ as above.

Figure 13.2 visualizes a support vector regression model in two dimensions. The red circles are values within the ϵ-tube and are thus considered to have a negligible error. In fact, the red circles do not affect the fitting of the optimal line g and even if they moved around, as long as they remain within the tube, the shape of g would not change. In contrast the blue diamonds have an unacceptable margin of error – as an example the top blue diamond will have $\xi'_i = 0$ but $\xi^*_i > 0$, thus influencing the estimation of g. Points on or outside the epsilon tube are referred to as *support vectors* as they affect the construction of the hyperplane. The $C \in \mathbb{R}_{>0}$ hyperparameter controls the slack parameters and thus as C increases, the number of errors (and subsequently support vectors) is allowed to increase resulting in low variance but higher bias, in contrast a lower C is more likely to introduce overfitting with low bias but high variance (Hastie, Tibshirani, and Friedman 2001). C should be tuned to control this trade-off.

The other core feature of SVMs is exploiting the *kernel trick*, which uses functions known as *kernels* to allow the model to learn a non-linear hyperplane whilst keeping the computations limited to lower-dimensional settings. Once the model coefficients have been estimated using the optimization above, predictions for a new observation $\mathbf{x}^*$ can be made using a function of the form

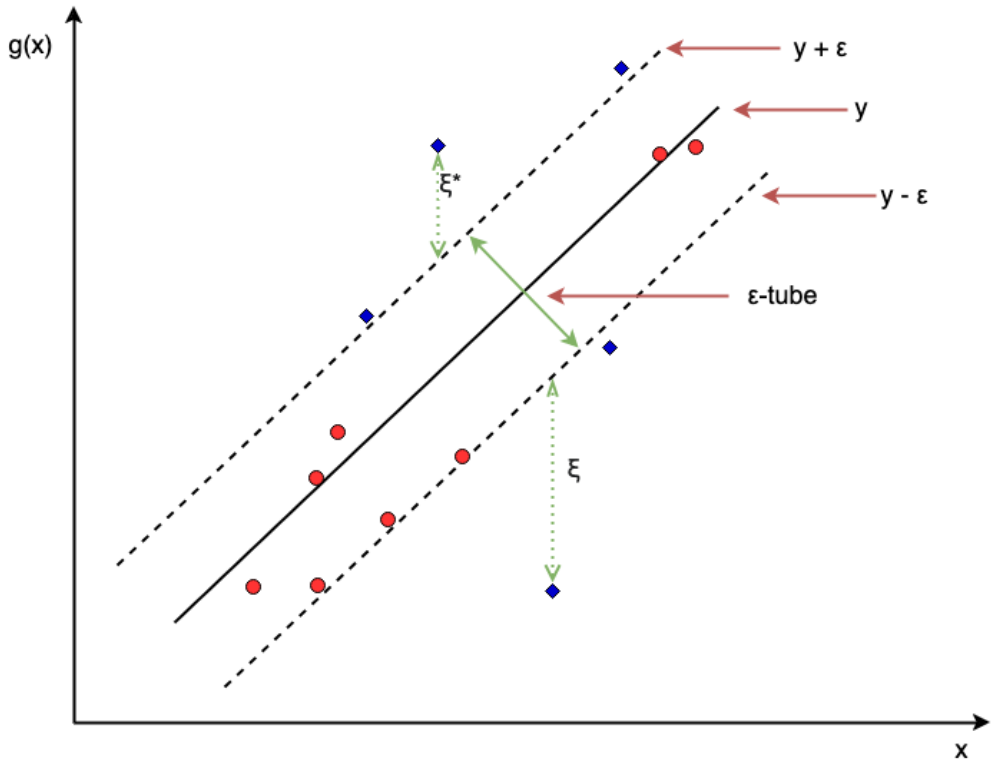


Figure 13.2: Visualising a support vector machine with an ϵ-tube and slack parameters $\boldsymbol{\xi}'$ and $\boldsymbol{\xi}^*$. Red circles are values within the ϵ-tube and blue diamonds are support vectors on and outside the tube. x-axis is single covariate, x, and y-axis is $g(x) = x\beta_1 + \alpha$.

$$\hat{g}(\mathbf{x}^*) = \sum_{i=1}^{n} \mu_i K(\mathbf{x}^*, \mathbf{x}_i) + \alpha \tag{13.2}$$

Details (including estimation) of the μ_i Lagrange multipliers are beyond the scope of this book, references are given at the end of this chapter for the interested reader. K is a kernel function, with common functions including the linear kernel, $K(x^*, x_i) = \sum_{j=1}^{p} x_{ij} x_j^*$, radial kernel, $K(x^*, x_i) = \exp(-\omega \sum_{j=1}^{p} (x_{ij} - x_j^*)^2)$ for some $\omega \in \mathbb{R}_{>0}$, and polynomial kernel, $K(x^*, x_i) = (1 + \sum_{j=1}^{p} x_{ij} x_j^*)^d$ for some $d \in \mathbb{N}_{>0}$.

The choice of kernel and its parameters, the regularization parameter C, and the acceptable error ϵ, are all tunable hyper-parameters, which makes the support vector machine a highly adaptable and often well-performing machine learning method. The parameters C and ϵ often have no clear apriori meaning (especially true in the survival setting predicting abstract rankings) and thus require tuning over a great range of values; no tuning usually results in a poor model fit (Probst, Boulesteix, and Bischl 2019).

13.2 SVMs for Survival Analysis

Extending SVMs to the survival domain (SSVMs) is a case of: i) identifying the quantity to predict; and ii) updating the optimization problem (13.1) and prediction function (13.2) to accommodate for censoring. In the first case, SSVMs can be used to either make survival time or ranking predictions, which are discussed in turn. The notation above is reused below for SSVMs, with additional notation introduced when required and now using the survival training data $(\mathbf{X}, \mathbf{t}, \boldsymbol{\delta})$.

13.2.1 Survival time SSVMs

To begin, consider the objective for support vector regression with the y variable replaced with the usual survival time outcome t, for all $i \in 1, ..., n$:

$$\begin{aligned} &\min_{\boldsymbol{\beta}, \alpha, \boldsymbol{\xi}', \boldsymbol{\xi}^*} \frac{1}{2} \|\boldsymbol{\beta}\|^2 + C \sum_{i=1}^{n} (\xi_i' + \xi_i^*) \\ &\text{subject to} \begin{cases} g(\mathbf{x}_i) & \geq t_i - \epsilon - \xi_i' \\ g(\mathbf{x}_i) & \leq t_i + \epsilon + \xi_i^* \\ \xi_i', \xi_i^* & \geq 0 \end{cases} \end{aligned} \tag{13.3}$$

In survival analysis, this translates to fitting a hyperplane in order to predict the true survival time. However, as with all adaptations from regression to survival analysis, there needs to be a method for incorporating censoring.

Recall the (t_l, t_u) notation to describe censoring as introduced in Chapter 4 such that the outcome occurs within the range (t_l, t_u). Let $\tau \in \mathbb{R}_{>0}$ be some known time-point, then an observation is:

- left-censored if the survival time is less than τ: $(t_l, t_u) = (-\infty, \tau)$;
- right-censored if the true survival time is greater than τ: $(t_l, t_u) = (\tau, \infty)$; or

- uncensored if the true survival time is known to be τ: $(t_l, t_u) = (\tau, \tau)$.

Define $\mathcal{L} = \{i : t_i > -\infty\}$ as the set of observations with a finite lower-bounded time, which can be seen above to be those that are right-censored or uncensored. Define $\mathcal{U} = \{i : t_i < \infty\}$ as the analogous set of observations with a finite upper-bounded time, which are those that are left-censored or uncensored.

Consider these definitions in the context of the constraints in 13.3. The first constraint ensures the hyperplane is greater than some lower-bound created by subtracting the slack parameter from the true outcome – given the set definitions above this constraint only has meaning for observations with a finite lower-bound, $i \in \mathcal{L}$, otherwise the constraint would include $g(\mathbf{x}_i) \geq -\infty$, which is clearly not useful. Similarly the second constraint ensures the hyperplane is less than some upper-bound, which again can only be meaningful for observations $i \in \mathcal{U}$. Restricting the constraints in this way leads to the optimization problem (Shivaswamy, Chu, and Jansche 2007) below and visualised in Figure 13.3:

$$\min_{\boldsymbol{\beta},\alpha,\boldsymbol{\xi}',\boldsymbol{\xi}^*} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\Big(\sum_{i\in\mathcal{U}} \xi_i + \sum_{i\in\mathcal{L}} \xi_i^*\Big)$$
$$\text{subject to} \begin{cases} g(\mathbf{x}_i) & \geq t_i - \xi_i', i \in \mathcal{L} \\ g(\mathbf{x}_i) & \leq t_i + \xi_i^*, i \in \mathcal{U} \\ \xi_i' \geq 0, \forall i \in \mathcal{L}; \xi_i^* \geq 0, \forall i \in \mathcal{U} \end{cases}$$

If no observations are censored then the optimization becomes the regression optimization in (13.1). Note that in SSVMs, the ϵ parameters are typically removed to better accommodate censoring and to help prevent the same penalization of over- and under-predictions. In contrast to this formulation, one *could* introduce more ϵ and C parameters to separate between under- and over-predictions and to separate right- and left-censoring, however this leads to eight tunable hyperparameters, which is inefficient and may increase overfitting (Fouodo et al. 2018; Land et al. 2011). The algorithm can be simplified to right-censoring only by removing the second constraint completely for anyone censored:

$$\min_{\boldsymbol{\beta},\alpha,\boldsymbol{\xi}',\boldsymbol{\xi}^*} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{n}(\xi_i' + \xi_i^*)$$
$$\text{subject to} \begin{cases} g(\mathbf{x}_i) & \geq t_i - \xi_i^* \\ g(\mathbf{x}_i) & \leq t_i + \xi_i', i : \delta_i = 1 \\ \xi_i', \xi_i^* & \geq 0 \end{cases}$$

$\forall i \in 1, ..., n$. With the prediction for a new observation $\mathbf{x}^*$ calculated as,

$$\hat{g}(\mathbf{x}^*) = \sum_{i=1}^{n} \mu_i^* K(\mathbf{x}_i, \mathbf{x}^*) - \delta_i \mu_i' K(\mathbf{x}_i, \mathbf{x}^*) + \alpha$$

Where again K is a kernel function and the calculation of the Lagrange multipliers is beyond the scope of this book.

13.2.2 Ranking SSVMs

Support vector machines can be used to estimate rankings by penalizing predictions that result in disconcordant predictions. Recall the definition of concordance from Chapter 7:

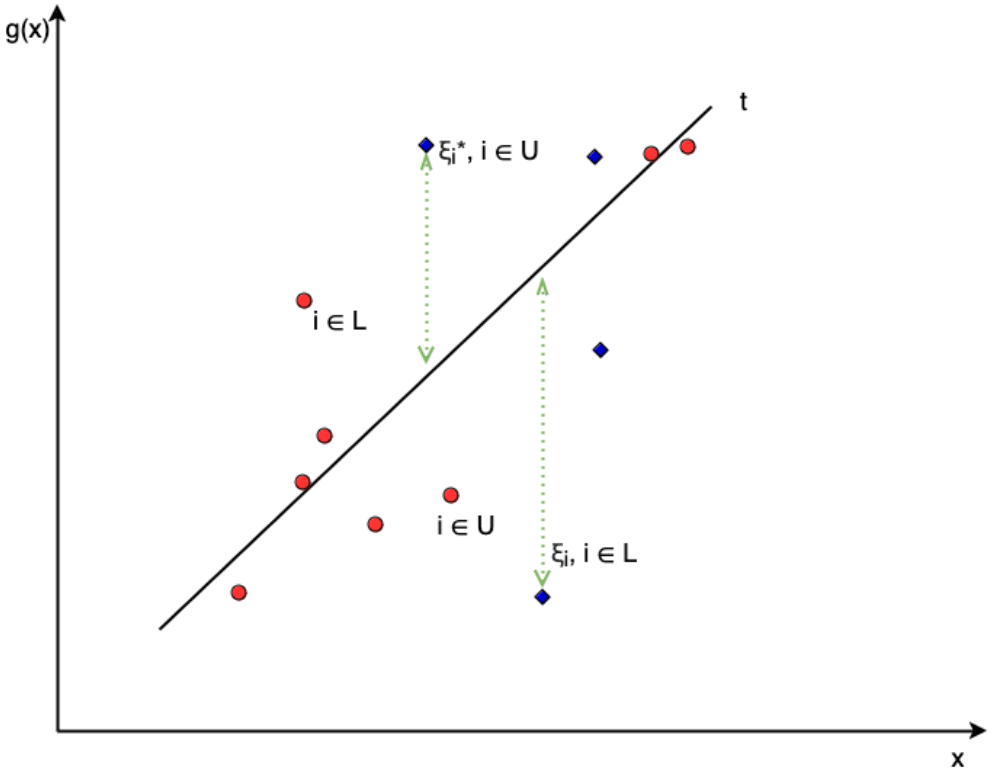


Figure 13.3: Visualising a survival time SVM. Blue diamonds are influential support vectors, which are uncensored or left-censored when $g(\mathbf{x}) < t$ or uncensored or right-censored when $g(\mathbf{x}) > t$. Red circles are non-influential observations.

ranking predictions for a pair of comparable observations (i, j) where $t_i < t_j \cap \delta_i = 1$, are called concordant if $r_i > r_j$ where r_i, r_j are the predicted ranks for observations i and j respectively and a higher value implies greater risk. Using the prognostic index as a ranking prediction (Section 6.3), a pair of observations is concordant if $g(\mathbf{x}_i) > g(\mathbf{x}_j)$ when $t_i < t_j$, leading to:

$$\min_{\boldsymbol{\beta},\alpha,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum_{i=1}^{n} \xi_i$$
$$\text{subject to} \begin{cases} g(\mathbf{x}_i) - g(\mathbf{x}_j) & \geq \xi_i, \forall i, j \in CP \\ \xi_i & \geq 0, i = 1, ..., n \end{cases}$$

where CP is the set of comparable pairs defined by $CP = \{(i, j) : t_i < t_j \wedge \delta_i = 1\}$. Given the number of pairs, the optimization problem quickly becomes difficult to solve with a very long runtime. To solve this problem Van Belle et al. (2011) found an efficient reduction that sorts observations in order of outcome time and then compares each data point i with the observation that has the next smallest *survival* time, skipping over censored observations, in maths: $j(i) := \arg\max_{j \in 1,...,n}\{t_j : t_j < t_i\}$. This is visualized in Figure 13.4 where six observations are sorted by outcome time from smallest (left) to largest (right). Starting from right to left, the first pair is made by matching the observation to the first uncensored outcome to the left, this continues to the end. In order for all observations to be used in the optimisation, the algorithm sets the first outcome to be uncensored hence observation 2 being compared to observation 1.

Using this reduction, the algorithm becomes

$$\min_{\boldsymbol{\beta},\alpha,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum_{i=1}^{n} \xi_i$$
$$\text{subject to} \begin{cases} g(\mathbf{x}_i) - g(\mathbf{x}_{j(i)}) & \geq t_i - t_{j(i)} - \xi_i \\ \xi_i & \geq 0 \end{cases}$$

$\forall i = 1, ..., n$. Note the updated right hand side of the constraint, which plays a similar role to the ϵ parameter by allowing 'mistakes' in predictions without penalty.

Predictions for a new observation $\mathbf{x}^*$ are calculated as,

$$\hat{g}(\mathbf{x}^*) = \sum_{i=1}^{n} \mu_i (K(\mathbf{x}_i, \mathbf{x}^*) - K(\mathbf{x}_{j(i)}, \mathbf{x}^*)) + \alpha$$

Where μ_i are again Lagrange multipliers.

13.2.3 Hybrid SSVMs

Finally, Van Belle et al. (2011) noted that the ranking algorithm could be updated to add the constraints of the regression model, thus providing a model that simultaneously optimizes for ranking whilst providing continuous values that can be interpreted as survival time predictions. This results in the hybrid SSVM with constraints $\forall i = 1, ..., n$:

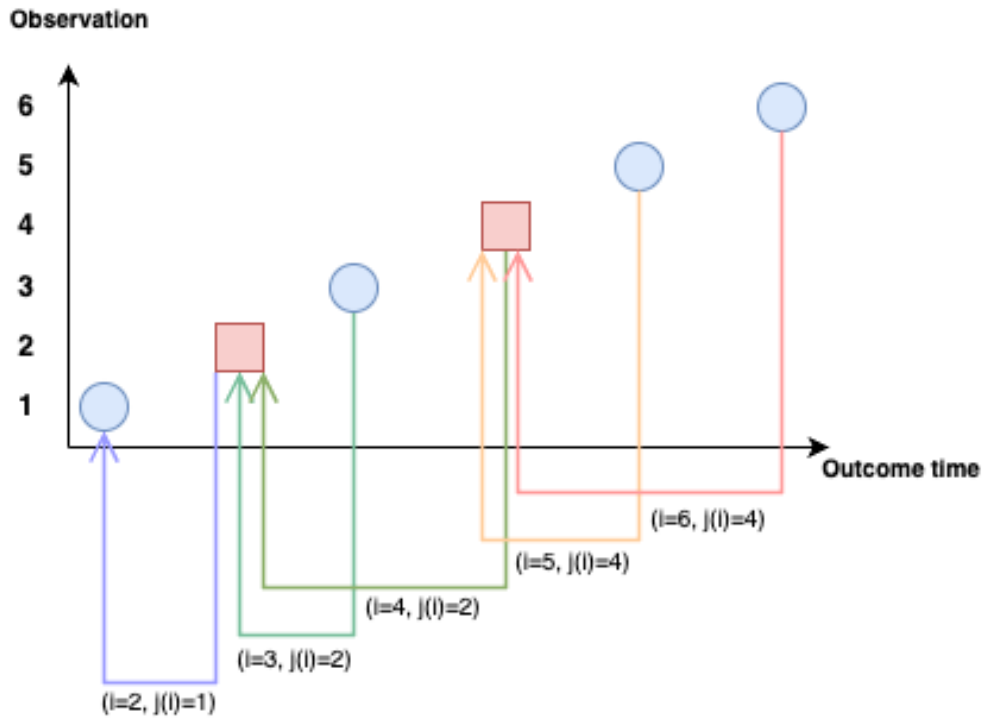


Figure 13.4: Van Belle SVM nearest neighbors reduction. Sorted observations are paired with the nearest uncensored outcome 'to the left'. Red squares are uncensored observations and blue circles are censored. The observation with the smallest outcome time is always treated as uncensored.

$$\min_{\boldsymbol{\beta},\alpha,\boldsymbol{\xi},\boldsymbol{\xi}',\boldsymbol{\xi}^*} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \gamma \sum_{i=1}^{n} \xi_i + C \sum_{i=1}^{n} (\xi_i' + \xi_i^*)$$

$$\text{subject to} \begin{cases} g(\mathbf{x}_i) - g(\mathbf{x}_{j(i)}) & \geq t_i - t_{j(i)} - \xi_i \\ g(\mathbf{x}_i) & \leq t_i + \xi_i^*, i : \delta_i = 1 \\ g(\mathbf{x}_i) & \geq t_i - \xi_i' \\ \xi_i, \xi_i', \xi_i^* & \geq 0 \end{cases}$$

The blue parts of the equation make up the ranking model and the red parts are the regression model. γ is the penalty associated with the regression method and C is the penalty associated with the ranking method. Setting $\gamma = 0$ results in the regression SVM and $C = 0$ results in the ranking SSVM. Hence, fitting the hybrid model and tuning these parameters is an efficient way to automatically detect which SSVM is best suited to a given task.

Once the model is fit, a prediction from given features $\mathbf{x}^* \in \mathbb{R}^p$, can be made using the equation below, again with the ranking and regression contributions highlighted in blue and red respectively.

$$\hat{g}(\mathbf{x}^*) = \sum_{i=1}^{n} \mu_i (K(\mathbf{x}_i, \mathbf{x}^*) - K(\mathbf{x}_{j(i)}, \mathbf{x}^*)) + \mu_i^* K(\mathbf{x}_i, \mathbf{x}^*) - \delta_i \mu_i' K(\mathbf{x}_i, \mathbf{x}^*) + \alpha$$

where μ_i, μ_i^*, μ_i' are Lagrange multipliers and K is a chosen kernel function, which may have further hyper-parameters to select or tune.

13.3 Conclusion

Key takeaways

- Support vector machines (SVMs) are a highly flexible machine learning method that can use the 'kernel trick' to represent infinite dimensional spaces in finite domains;
- Survival SVMs (SSVMs) extend regression SVMs by either making survival time predictions, ranking predictions, or a combination of the two;
- The hybrid SSVM provides an efficient method that encapsulates all the elements of regression and ranking SSVMs and is therefore a good model to include in benchmark experiments to test the potential of SSVMs.

Limitations

- SSVMs can only perform well with extensive tuning of hyper-parameters over a wide parameter space. To-date, no papers have experimented with the tuning range for the γ and C parameters, we note (Fouodo et al. 2018) tune over $(2^{-5}, 2^5)$.
- Even using the regression or hybrid model, the authors' experiments with the SSVM have consistently shown 'survival time' estimates tend to be unrealistically large.
- Due to the above limitation, regression estimates cannot be meaningful interpreted and as a consequence there is no sensible composition to create a distribution

prediction from an SSVM. Hence, we are hesitant to suggest usage of SSVMs outside of ranking-based problems.

Further reading

- Shivaswamy, Chu, and Jansche (2007), Khan and Bayer Zubek (2008), Land et al. (2011), and Van Belle et al. (2011) to learn more about regression SSVMs.
- Evers and Messow (2008), Van Belle et al. (2007), Van Belle et al. (2008), and Van Belle et al. (2011) for more information about ranking SSVMs.
- Goli, Mahjub, Faradmal, and Soltanian (2016) and Goli, Mahjub, Faradmal, Mashayekhi, et al. (2016) introduce mean residual lifetime optimization SSVMs.
- Fouodo et al. (2018) surveys and benchmarks SSVMs.

14

Boosting Methods

This chapter introduces gradient boosting machines (GBMs) for regression and then describes the extensions to survival analysis. GBMs are a highly modular and flexible algorithm that can be applied to any machine learning model though are generally used with 'weak' models such as decision trees. Boosting fits a weak learner to data and then trains models iteratively on the residuals from the previous iteration. GBMs extend naturally to survival analysis by providing loss functions suitable for the survival setting, such as likelihoods associated with PH and AFT models, as well as differentiable adaptations to the C-index. Survival GBMs generally make ranking predictions but in some cases can be composed to survival distribution predictions, for example when using boosting to estimate proportional hazards and accelerated failure time models. This chapter primarily covers the work of Binder, Mayr, and Schmid. GBMs have been demonstrated to perform well on both low- and high-dimensional data in survival analysis and are therefore a popular class of machine learning model within this domain. However, GBMs can be computationally intensive and may require specialist software such as XGBoost to efficiently optimise performance.

> ! Major changes expected!
>
> **This page is a work in progress and major changes will be made over time.**

Boosting is a machine learning strategy that can be applied to any predictive model. Similarly to random forests, boosting is an ensemble method that creates a model from a 'committee' of learners. The committee is formed of *weak* learners that make poor predictions individually, which creates a *slow learning* approach that requires many iterations for a model to be a good fit to the data. Boosting models are similar to random forests in that both make predictions from a large committee of learners. However the two differ in how the members of the committee are correlated and in how they are combined to make a prediction. In random forests, each decision tree is grown independently and their predictions are combined by a simple mean calculation. In contrast, weak learners in a boosting model are fit sequentially with errors from one learner used to train the next, predictions are then made by a weighted linear combination of predictions from each learner (Figure 14.1).

14.1 GBMs for Regression

One of the earliest boosting algorithms is AdaBoost (Freund and Schapire 1996), which is more generally a Forward Stagewise Additive Model (FSAM) with an exponential loss

(Hastie, Tibshirani, and Friedman 2001). Today, the most widely used boosting model is the Gradient Boosting Machine (GBM) (J. H. Friedman 2001) or extensions thereof.

Figure 14.1 illustrates the process of training a GBM in a least-squares regression setting:

1. A weak learner, f_1, often a decision tree of shallow depth is fit on the training data $(\mathbf{X}, \mathbf{y})$.
2. Predictions from the learner, $f_1(\mathbf{X})$, are compared to the ground truth, $\mathbf{y}$, and the residuals are calculated as $\mathbf{r}_1 = f_1(\mathbf{X}) - \mathbf{y}$.
3. The next weak learner, f_2, uses the previous residuals for the target prediction, $(\mathbf{X}, \mathbf{r}_1)$
4. This is repeated to train M learners, $f_1, ..., f_M$

Predictions are then made as $\hat{\mathbf{y}} = f_1(\mathbf{X}) + f_2(\mathbf{X}) + ... + f_M(\mathbf{X})$ (for now ignoring any individual learner weights).

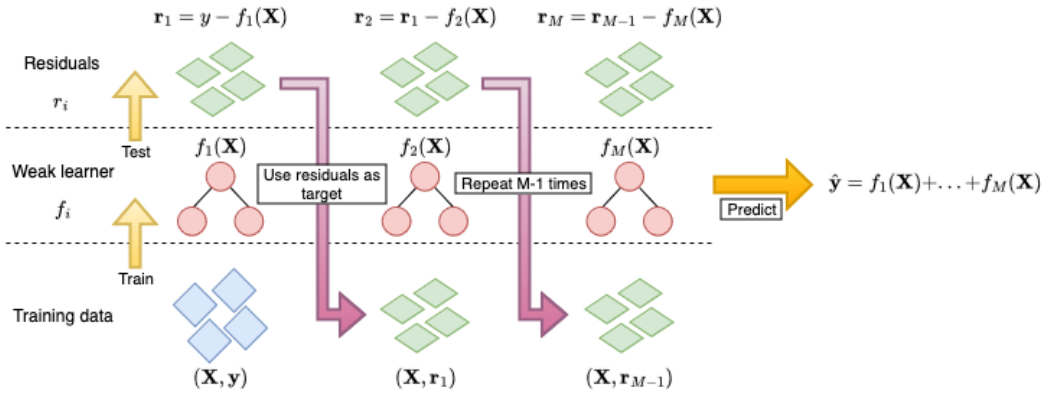


Figure 14.1: Least squares regression Boosting algorithm where the gradient is calculated as the difference between ground truth and predictions.

This is a simplification of the general gradient boosting algorithm, where the residuals are used to train the next model. More generally, the algorithm involves choosing a suitable, differentiable loss function related to the problem of interest. In each iteration, the negative gradient of this loss function is computed with respect to the current model's predictions. This negative gradient acts as a pseudo-residual, guiding the training of the next model. Residuals can be used in the regression case as these are proportional to the negative gradient of the mean squared error.

The algorithm above is also a simplification as no hyper-parameters other than M were included for controlling the algorithm. In order to reduce overfitting, three common hyper-parameters are utilised:

Number of iterations, M: The number of iterations is often claimed to be the most important hyper-parameter in GBMs and it has been demonstrated that as the number of iterations increases, so too does the model performance (with respect to a given loss on test data) up to a certain point of overfitting (Buhlmann 2006; Hastie, Tibshirani, and Friedman 2001; Schmid and Hothorn 2008a). This makes sense as the foundation of boosting rests on the idea that weak learners can slowly be combined to form a single powerful model. Finding the optimal value of M is critical as a value too small will result in poor predictions, whilst a value too large will result in biased predictions.

Step-size, ν: The step-size parameter is a shrinkage parameter that controls the contribution of each weak learner at each iteration. Several studies have demonstrated that GBMs perform better when shrinkage is applied and a value of $\nu \leq 0.1$ is often suggested (Buhlmann and Hothorn 2007; Hastie, Tibshirani, and Friedman 2001; J. H. Friedman 2001; D. K. K. Lee, Chen, and Ishwaran 2019; Schmid and Hothorn 2008a). The optimal values of ν and M depend on each other, such that smaller values of ν require larger values of M, and vice versa. This is intuitive as smaller ν results in a slower learning algorithm and therefore more iterations are required to fit the model. Accurately selecting the M parameter is generally considered to be of more importance, and therefore a value of ν is often chosen heuristically (e.g. the common value of 0.1) and then M is tuned by cross-validation and/or early-stopping, which is the process of monitoring the model's training performance and stopping when a set performance is reached or when performance stagnates (i.e., no improvement over a set number of rounds).

Subsampling proportion, ϕ: Sampling a fraction, ϕ, of the training data at each iteration can improve performance and reduce runtime (Hastie, Tibshirani, and Friedman 2001), with $\phi = 0.5$ often used. Motivated by the success of bagging in random forests, stochastic gradient boosting (J. Friedman 1999) randomly samples the data in each iteration. It appears that subsampling performs best when also combined with shrinkage (Hastie, Tibshirani, and Friedman 2001) and as with the other hyper-parameters, selection of ϕ is usually performed by nested cross-validation.

As well as these parameters, the underlying weak learner hyper-parameters are also commonly tuned. If using a decision tree, then it is usual to restrict the number of terminal nodes in the tree to be between 4 and 8, which corresponds to two or three splits in the tree. Including these hyper-parameters, and some differentiable loss, L, the general gradient boosting machine algorithm is as follows:

1. $g_0 \leftarrow$ Initial guess
2. **For** $m = 1, ..., M$:
3. $\quad \mathcal{D}^*_{train} \leftarrow$ Randomly sample $\mathcal{D}_{train}$ with probability ϕ
4. $\quad r_{im} \leftarrow -[\frac{\partial L(y_i, g_{m-1}(X_i))}{\partial g_{m-1}(X_i)}], \forall i \in \{i : X_i \in \mathcal{D}^*_{train}\}$
5. $\quad$ Fit a weak learner, h_m, to $(\mathbf{X}, \mathbf{r}_m)$
6. $\quad g_m \leftarrow g_{m-1} + \nu h_m$
7. **end For**
8. **return** $\hat{g} = g_M$

Note:

1. The initial guess (or *offset*), g_0, is often the mean of y for regression problems but can also simply be 0.
2. Line 4 is the calculation of the negative gradient, which is equivalent to calculating the residuals in a regression problem with the mean squared error loss.
3. Lines 5-6 differ between implementations, with some fitting multiple weak learners and selecting the one that minimizes a simple optimization problem. The version above is simplest to implement and quickest to run, whilst still providing good model performance.

Once the model is trained, predictions are made for new data, $\mathbf{X}_{test}$ with

$$\hat{Y} = \hat{g}(\mathbf{X}_{test}) = g_0(\mathbf{X}_{test}) + \nu \sum_{i=1}^{M} g_i(\mathbf{X}_{test})$$

GBMs provide a flexible, modular algorithm, primarily comprised of a differentiable loss to minimize, L, and the selection of weak learners. Perhaps the most common weak learners are decision trees, linear least squares (J. H. Friedman 2001) and smoothing splines (Bühlmann and Yu 2003). The use of trees means that the structure of the resulting prediction function can vary substantially, for example using stumps (trees with only one split) results in an additive model with main effects only, whereas decision trees with an extra layer yield two-way interactions.

This chapter focuses on tree-based learners, which are primarily used for survival analysis, due to the flexibility demonstrated in Chapter 12. See references at the end of the chapter for other weak learners. Extension to survival analysis therefore follows by considering alternative losses.

14.2 GBMs for Survival Analysis

Unlike other machine learning algorithms that historically ignored survival analysis, early GBM papers considered boosting in a survival context (Ridgeway 1999); though there appears to be a decade gap before further considerations were made in the survival setting. After that period, developments, discussed in this chapter, by Binder, Schmid, and Hothorn, adapted GBMs to a framework suitable for survival analysis.

All survival GBMs make ranking predictions and none are able to directly predict survival distributions. However, depending on the underlying model, the predictions may be indirectly composed into a survival distribution, for example algorithms that assume a proportional hazards (PH) or accelerated failure time (AFT) form. This section starts with those models with simpler underlying forms, then explores more complex alternatives.

14.2.1 PH and AFT GBMs

The negative log-likelihood of the semi-parametric PH and fully-parametric AFT models can be derived from the (partial) likelihoods presented in Section 4.6.1. Boosting algorithms use these losses to train the model coefficients, $\boldsymbol{\beta}$, hence at each iteration in the algorithm, $g_m(\mathbf{x}_i) = \mathbf{x}_i \boldsymbol{\beta}^{(m)}$, where $\boldsymbol{\beta}^{(m)}$ are the updated coefficients in iteration m.

The Cox partial likelihood (Cox 1972, 1975) is given by

$$L^{PH}(\boldsymbol{\beta}) = \prod_{i:\delta_i=1}^{n} \frac{\exp(\eta_i)}{\sum_{j \in \mathcal{R}_{t_i}}^{n} \exp(\eta_j)}$$

with corresponding negative log-likelihood

$$-l^{PH}(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \delta_i \Big[\eta_i \; - \; \log\Big(\sum_{j \in \mathcal{R}_{t_i}}^{n} \exp(\eta_i) \Big) \Big] \tag{14.1}$$

where $\mathcal{R}_{t_i}$ is the set of patients at risk at time t_i and $\eta_i = \mathbf{x}_i\boldsymbol{\beta}$.

The gradient of $-l^{PH}$ at iteration m is then

$$r_{im} := \delta_i - \sum_{j=1}^{n} \delta_j \frac{\mathbb{I}(t_i \geq t_j)\exp(g_{m-1}(\mathbf{x}_i))}{\sum_{k\in\mathcal{R}_{t_j}} \exp(g_{m-1}(\mathbf{x}_k))} \tag{14.2}$$

where $g_{m-1}(\mathbf{x}_i) = \mathbf{x}_i\boldsymbol{\beta}^{(m-1)}$.

For non-PH data, boosting an AFT model can outperform boosted PH models (Schmid and Hothorn 2008b). The AFT is defined by

$$\log \mathbf{y} = \boldsymbol{\eta} + \sigma W$$

where W is a random noise variable independent of X, and σ is a scale parameter controlling the amount of noise; again $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$. By assuming a distribution on W, a distribution is assumed for the full parametric distribution of the survival time $\mathbf{y}$. The model is boosted by simultaneously estimating σ and $\boldsymbol{\beta}$. Assuming a location-scale distribution with location $g(\mathbf{x}_i)$ and scale σ, the negative log-likelihood in the mth iteration is (Klein and Moeschberger 2003)

$$-l_m^{AFT}(\boldsymbol{\beta}) = -\frac{\partial}{\partial f}\sum_{i=1}^{n} \delta_i \Big[-\log \hat{\sigma_{m-1}} + \log f_W\Big(\frac{\log(t_i) - \hat{g}_{m-1}(\mathbf{x}_i)}{\hat{\sigma}_{m-1}}\Big)\Big] + (1-\delta_i)\Big[\log S_W\Big(\frac{\log(t_i) - \hat{g}_{m-1}(\mathbf{x}_i)}{\hat{\sigma}_{m-1}}\Big)\Big]$$

where $\hat{g}_{m-1}, \hat{\sigma}_{m-1}$ are the location-scale parameters estimated in the previous iteration. After updating $\hat{g}_m$, the scale parameter, $\hat{\sigma}_m$, is updated as

$$\hat{\sigma}_m := \arg\min_{\sigma} \; -l_m^{AFT}(\boldsymbol{\beta})$$

σ_0 is commonly initialized as 1 (Schmid and Hothorn 2008b).

As well as boosting fully-parametric AFTs, one could also consider boosting semi-parametric AFTs, for example using the Gehan loss (Johnson and Long 2011) or using Buckley-James imputation (Z. Wang and Wang 2010). However, known problems with semi-parametric AFT models and the Buckey-James procedure (Wei 1992), as well as a lack of off-shelf implementation, mean that these methods are rarely used in practice.

14.2.2 Discrimination Boosting

Instead of optimising models based on a given model form, one could instead estimate $\hat{\eta}$ by optimizing a concordance index, such as Uno's or Harrell's C (Y. Chen et al. 2013; Mayr and Schmid 2014). Consider Uno's C (Section 7.1):

$$C_U(\hat{g}, \mathcal{D}_{train}) = \frac{\sum_{i\neq j} \delta_i \{\hat{G}_{KM}(t_i)\}^{-2}\mathbb{I}(t_i < t_j)\mathbb{I}(\hat{g}(\mathbf{x}_i) > \hat{g}(\mathbf{x}_j))}{\sum_{i\neq j} \delta_i \{\hat{G}_{KM}(t_i)\}^{-2}\mathbb{I}(t_i < t_j)}$$

where $\hat{G}_{KM}$ is again the Kaplan-Meier estimate of the censoring distribution.

The GBM algorithm requires that the chosen loss, here C_U, be differentiable with respect to $\hat{g}(X)$, which is not the case here due to the indicator term, $\mathbb{I}(\hat{g}(X_i) > \hat{g}(X_j))$, however this term can be replaced with a sigmoid function to create a differentiable loss (Ma and Huang 2006)

$$K(u|\omega) = \frac{1}{1 + \exp(-u/\omega)}$$

where ω is a tunable hyper-parameter controlling the smoothness of the approximation. The measure to optimise is then,

$$C_{USmooth}(\boldsymbol{\beta}|\omega) = \sum_{i \neq j} \frac{k_{ij}}{1 + \exp\left[(\hat{g}(X_j) - \hat{g}(X_i))/\omega)\right]} \tag{14.3}$$

with

$$k_{ij} = \frac{\Delta_i (\hat{G}_{KM}(T_i))^{-2} \mathbb{I}(T_i < T_j)}{\sum_{i \neq j} \Delta_i (\hat{G}_{KM}(T_i))^{-2} \mathbb{I}(T_i < T_j)}$$

The negative gradient at iteration m for observation i is then calculated as,

$$r_{im} := -\sum_{j=1}^{n} k_{ij} \frac{-\exp(\frac{\hat{g}_{m-1}(\mathbf{x}_j) - \hat{g}_{m-1}(\mathbf{x}_i)}{\omega})}{\omega(1 + \exp(\frac{\hat{g}_{m-1}(\mathbf{x}_j) - \hat{g}_{m-1}(\mathbf{x}_i)}{\omega}))} \tag{14.4}$$

The GBM algorithm is then followed as normal with the above loss and gradient. This algorithm may be more insensitive to overfitting than others (Mayr, Hofner, and Schmid 2016), however stability selection (Meinshausen and Bühlmann 2010), which is implemented in off-shelf software packages (Hothorn et al. 2020), can be considered for variable selection.

14.2.3 CoxBoost

Finally, 'CoxBoost' is an alternative method to boost Cox models and has been demonstrated to perform well in experiments. This algorithm boosts the Cox PH by optimising the penalized partial-log likelihood; additionally the algorithm allows for mandatory (or 'forced') covariates (Binder and Schumacher 2008). In medical domains the inclusion of mandatory covariates may be essential, either for model interpretability, or due to prior expert knowledge. CoxBoost deviates from the algorithm presented above by instead using an offset-based approach for generalized linear models (Tutz and Binder 2007).

Let $\mathcal{I} = \{1, ..., p\}$ be the indices of the covariates, let $\mathcal{I}_{mand}$ be the indices of the mandatory covariates that must be included in all iterations, and let $\mathcal{I}_{opt} = \mathcal{I} \setminus \mathcal{I}_{mand}$ be the indices of the optional covariates that may be included in any iteration. In the mth iteration, the algorithm fits a weak learner on all mandatory covariates and *one* optional covariate:

$$\mathcal{I}_m = \mathcal{I}_{mand} \cup \{x | x \in \mathcal{I}_{opt}\}$$

In addition, a penalty matrix $\mathbf{P} \in \mathbb{R}^{p \times p}$ is considered such that $P_{ii} > 0$ implies that covariate i is penalized and $P_{ii} = 0$ means no penalization. In practice, this is usually a diagonal matrix (Binder and Schumacher 2008) and by setting $P_{ii} = 0, i \in I_{mand}$ and $P_{ii} > 0, i \notin I_{mand}$, only

optional (non-mandatory) covariates are penalized. The penalty matrix *can* vary with each iteration, which allows for a highly flexible approach. However, in implementation a simpler approach is to use the same penalty in each iteration, which could be a single number, so the same penalty is applied to all penalized covariates, or a single matrix, such that the penalty differs between covariates but stays the same in each iteration (Binder 2013).

At the mth iteration and the kth set of indices to consider ($k = 1, ..., p$), the loss to optimize is the penalized partial-log likelihood given by

$$l_{pen}(\gamma_{mk}) = \sum_{i=1}^{n} \delta_i \Big[\eta_{i,m-1} + \mathbf{x}_{i,\mathcal{I}_{mk}} \gamma_{mk}^{\top}\Big] - \\ \delta_i \log\Big(\sum_{j=1}^{n} \mathbb{I}(t_j \leq t_i) \exp(\eta_{i,m-1} + \mathbf{x}_{i,\mathcal{I}_{mk}} \gamma_{mk}^{\top}\Big) - \lambda \gamma_{mk} \mathbf{P}_{mk} \gamma_{mk}^{\top}$$

where $\eta_{i,m} = \mathbf{x}_i \beta_m$, γ_{mk} are the coefficients corresponding to the covariates in $\mathcal{I}_{mk}$ which is the possible set of candidates for a subset of total candidates $k = 1, ..., p$; $\mathbf{P}_{mk}$ is the penalty matrix; and λ is a penalty hyper-parameter to be tuned or selected.[1]

In each iteration, all potential candidate sets (the union of mandatory covariates and one other covariate) are updated by

$$\hat{\gamma}_{mk} = \mathbf{I}_{pen}^{-1}(\hat{\gamma}_{(m-1)k}) U(\hat{\gamma}_{(m-1)k})$$

where $U(\gamma) = \partial l / \partial \gamma(\gamma)$ and $\mathbf{I}_{pen}^{-1} = \partial^2 l / \partial \gamma \partial \gamma^T (\gamma + \lambda \mathbf{P}_{(m-1)k})$ are the first and second derivatives of the unpenalized partial-log-likelihood. This is the key difference between CoxBoost and standard gradient boosting as CoxBoost uses updates based on Fisher scoring iterations (including both the first and the second derivatives), whereas the latter only makes use of the gradient (first derivative).

The optimal set is then found as

$$k^* := \arg\max_k \; l_{pen}(\hat{\gamma}_{mk})$$

and the estimated coefficients are updated with

$$\hat{\beta}_m = \hat{\beta}_{m-1} + \hat{\gamma}_{mk^*}, \quad k^* \in \mathcal{I}_{mk}$$

This deviates from the standard GBM algorithm by directly optimizing l_{pen} and not its gradient, additionally model coefficients are iteratively updated instead of a more general model form.

CoxBoost is extended to the competing risks setting by modelling the subdistribution hazards under a Fine and Gray model (Binder et al. 2009).

14.3 Conclusion

[1] On notation, note that $\mathbf{P}_{ij}$ refers to the penalty matrix in the ith iteration for the jth set of indices, whereas P_{ij} is the (i, j)th element in the matrix $\mathbf{P}$.

Key takeaways

- GBMs are a highly flexible and powerful machine learning tool due to their modular nature, where any differentiable loss and any weak base-learner can be combined. This makes them particularly useful in the survival setting as they can be easily adapted to different survival settings.
- There is evidence that boosting models can outperform the Cox PH even in low-dimensional settings (Schmid and Hothorn 2008b), which is not not something all ML models can claim.
- GBMs can perform very well in high-dimensional settings due to variable selection that can take place both during covariate selection as weak learners are chosen to optimize performance in each step, and within the base learners (especially when using trees).

Limitations

- Boosting, especially with tree learners, is viewed as a black-box model that is increasingly difficult to interpret as the number of iterations increase. However, there are several methods for increasing interpretability, such as variable importance and SHAPs (Lundberg and Lee 2017).
- Boosting often relies on intensive computing power, however, dedicated packages such as **xgboost** (T. Chen et al. 2020), exist to push CPU/GPUs to their limits in order to optimise predictive performance.

Further reading

- Bühlmann and Yu (2003); Hothorn et al. (2020); Z. Wang and Wang (2010) for more general information and background on componentwise GBMs
- J. H. Friedman (2001); Z. Wang and Wang (2010) for linear least squares weak learners
- Bühlmann and Yu (2003); J. H. Friedman (2001) for decision tree weak learners
- Ridgeway (1999) for early research into GBMs for survival analysis
- Johnson and Long (2011) and Z. Wang and Wang (2010) for semi-parametric AFT boosting

15

Neural Networks

TODO (150-200 WORDS)

> ! Major changes expected!
>
> **This page is a work in progress and major changes will be made over time.**

Before starting the survey on neural networks, first a comment about their transparency and accessibility. Neural networks are infamously difficult to interpret and train, with some calling building and training neural networks an 'art' (Hastie, Tibshirani, and Friedman 2001). As discussed in the introduction of this book, whilst neural networks are not transparent with respect to their predictions, they are transparent with respect to implementation. In fact the simplest form of neural network, as seen below, is no more complex than a simple linear model. With regard to accessibility, whilst it is true that defining a custom neural network architecture is complex and highly subjective, established models are implemented with a default architecture and are therefore accessible 'off-shelf'.

15.0.1 Neural Networks for Regression

(Artificial) Neural networks (ANNs) are a class of model that fall within the greater paradigm of *deep learning*. The simplest form of ANN, a feed-forward single-hidden-layer network, is a relatively simple algorithm that relies on linear models, basic activation functions, and simple derivatives. A short introduction to feed-forward regression ANNs is provided to motivate the survival models. This focuses on single-hidden-layer models and increasing this to multiple hidden layers follows relatively simply.

The single hidden-layer network is defined through three equations

$$
\begin{aligned}
&Z_m = \sigma(\alpha_{0m} + \alpha_m^T X_i), \quad m = 1, ..., M \\
&T = \beta_{0k} + \beta_k^T Z, \quad k = 1, .., K \\
&g_k(X_i) = \phi_k(T)
\end{aligned}
$$

where $(X_1, ..., X_n) \overset{i.i.d.}{\sim} X$ are the usual training data, α_{0m}, β_0 are bias parameters, and $\theta = \{\alpha_m, \beta\}$ $(m = 1, .., , M)$ are model weights where M is the number of hidden units. K is the number of classes in the output, which for regression is usually $K = 1$. The function ϕ is a 'link' or 'activation function', which transforms the predictions in order to provide an outcome of the correct return type; usually in regression, $\phi(x) = x$. σ is the 'activation function', which transforms outputs from each layer. The α_m parameters are often referred

to as 'activations'. Different activation functions may be used in each layer or the same used throughout, the choice is down to expert knowledge. Common activation functions seen in this section include the sigmoid function,

$$\sigma(v) = (1 + \exp(-v))^{-1}$$

tanh function,

$$\sigma(v) = \frac{\exp(v) - \exp(-v)}{\exp(v) + \exp(-v)} \tag{15.1}$$

and ReLU (Nair and Hinton 2010)

$$\sigma(v) = \max(0, v) \tag{15.2}$$

A single-hidden-layer model can also be expressed in a single equation, which highlights the relative simplicity of what may appear a complex algorithm.

$$g_k(X_i) = \sigma_0(\beta_{k0} + \sum_{h=1}^{H}(\beta_{kh}\sigma_h(\beta_{h0} + \sum_{m=1}^{M}\beta_{hm}X_{i;m})) \tag{15.3}$$

where H are the number of hidden units, β are the model weights, σ_h is the activation function in unit h, also σ_0 is the output unit activation, and $X_{i;m}$ is the ith observation features in the mth hidden unit.

An example feed-forward single-hidden-layer regression ANN is displayed in (Figure 15.1). This model has 10 input units, 13 hidden units, and one output unit; two bias parameters are fit. The model is described as 'feed-forward' as there are no cycles in the node and information is passed forward from the input nodes (left) to the output node (right).

Back-Propagation

The model weights, θ, in this section are commonly fit by 'back-propagation' although this method is often considered inefficient compared to more recent advances. A brief pseudo-algorithm for the process is provided below.

Let L be a chosen loss function for model fitting, let $\theta = (\alpha, \beta)$ be model weights, and let $J \in \mathbb{N}_{>0}$ be the number of iterations to train the model over. Then the back-propagation method is given by,

- **For** $j = 1, ..., J$: *[]* Forward Pass [i.] Fix weights $\theta^{(j-1)}$. *[ii.] Compute predictions* $\hat{Y} := \hat{g}_k^{(j)}(X_i|\theta^{(j-1)})$ *with (15.3).* [] *Backward Pass [iii.] Calculate the gradients of the loss* $L(\hat{Y}|\mathcal{D}_{train})$. [] *Update* *[iv.] Update $\alpha^{(r)}, \beta^{(r)}$ with gradient descent.
- **End For**

In regression, a common choice for L is the squared loss,

$$L(\hat{g}, \theta|\mathcal{D}_{train}) = \sum_{i=1}^{n}(Y_i - \hat{g}(X_i|\theta))^2$$

which may help illustrate how the training outcome, $(Y_1, ..., Y_n) \overset{i.i.d.}{\sim} Y$, is utilised for model fitting.

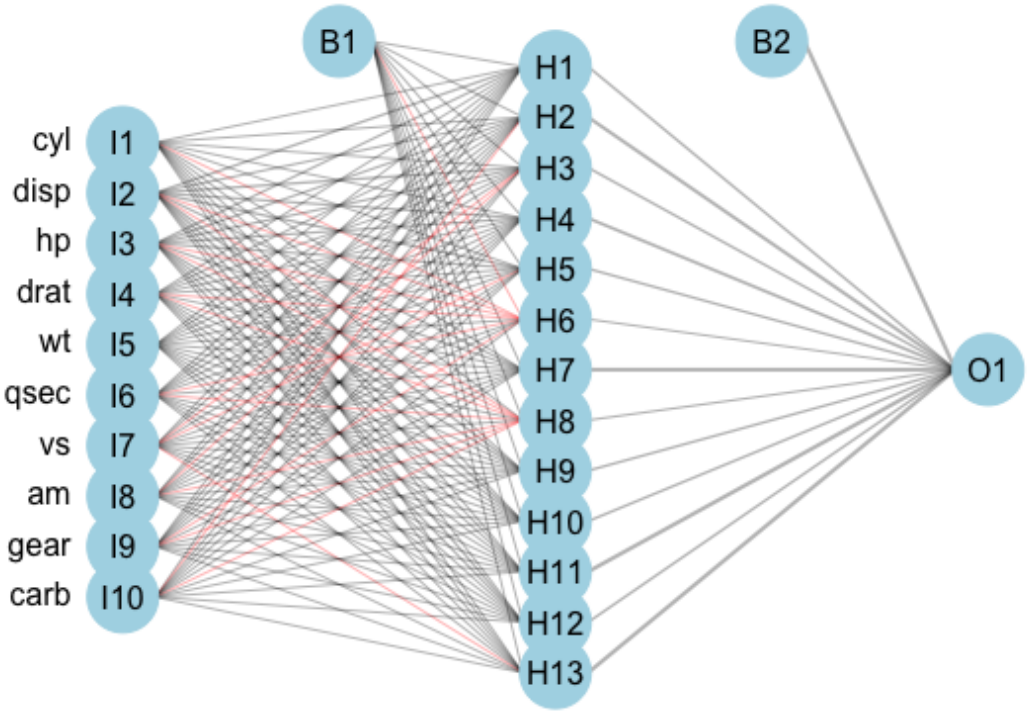


Figure 15.1: Single-hidden-layer artificial neural network with 13 hidden units fit on the `mtcars` (Henderson and Velleman 1981) dataset using the **nnet** (N. Venables and D. Ripley 2002) package, and **gamlss.add** (Stasinopoulos et al. 2020) for plotting. Left column are input variables, I1-I10, second column are 13 hidden units, H1-H13, right column is single output variable, O1. B1 and B2 are bias parameters.

Making Predictions

Once the model is fitted, predictions for new data follow by passing the testing data as inputs to the model with fitted weights,

$$g_k(X^*) = \sigma_0(\hat{\beta}_{k0} + \sum_{h=1}^{H}(\hat{\beta}_{kh}\sigma_h(\hat{\beta}_{h0} + \sum_{m=1}^{M}\hat{\beta}_{hm}X_m^*))$$

Hyper-Parameters

In practice, a regularization parameter, λ, is usually added to the loss function in order to help avoid overfitting. This parameter has the effect of shrinking model weights towards zero and hence in the context of ANNs regularization is usually referred to as 'weight decay'. The value of λ is one of three important hyper-parameters in all ANNs, the other two are: the range of values to simulate initial weights from, and the number of hidden units, M.

The range of values for initial weights is usually not tuned but instead a consistent range is specified and the neural network is trained multiple times to account for randomness in initialization.

The regularization parameter and number of hidden units, M, depend on each other and have a similar relationship to the learning rate and number of iterations in the GBMs (**?@sec-surv-ml-models-boost**). Like the GBMs, it is simplest to set a high number of hidden units and then tune the regularization parameter (Bishop 2006; Hastie, Tibshirani, and Friedman 2001). Determining how many hidden layers to include, and how to connect them, is informed by expert knowledge and well beyond the scope of this book; decades of research has been required to derive sensible new configurations.

Training Batches

ANNs can either be trained using complete data, in batches, or online. This decision is usually data-driven and will affect the maximum number of iterations used to train the algorithm; as such this will also often be chosen by expert-knowledge and not empirical methods such as cross-validation.

Neural Terminology

Neural network terminology often reflects the structures of the brain. Therefore ANN units are referred to as nodes or neurons and sometimes the connections between neurons are referred to as synapses. Neurons are said to be 'fired' if they are 'activated'. The simplest example of activating a neuron is with the Heaviside activation function with a threshold of 0: $\sigma(v) = \mathbb{I}(v \geq 0)$. Then a node is activated and passes its output to the next layer if its value is positive, otherwise it contributes no value to the next layer.

15.0.2 Neural Networks for Survival Analysis

Surveying neural networks is a non-trivial task as there has been a long history in machine learning of publishing very specific data-driven neural networks with limited applications; this is also true in survival analysis. This does mean however that where limited developments for survival were made in other machine learning classes, ANN survival adaptations have been around for several decades. A review in 2000 by Schwarzer *et al.* surveyed 43 ANNs for diagnosis and prognosis published in the first half of the 90s, however only up to ten

of these are specifically for survival data.[1] Of those, Schwarzer *et al.* deemed three to be 'na"ive applications to survival data', and recommended for future research models developed by Liestøl *et al.* (1994) (Liestol, Andersen, and Andersen 1994), Faraggi and Simon (1995) (Faraggi and Simon 1995), and Biganzoli *et al.* (1998) (E. Biganzoli et al. 1998).

This survey will not be as comprehensive as the 2000 survey, and nor has any survey since, although there have been several ANN reviews (B. D. Ripley and Ripley 2001; Huang et al. 2020b; Ohno-Machado 1996; Yang 2010; W. Zhu et al. 2020). ANNs are considered to be a black-box model, with interpretability decreasing steeply as the number of hidden layers and nodes increases. In terms of accessibility there have been relatively few open-source packages developed for survival ANNs; where these are available the focus has historically been in Python, with no R implementations. The new **survivalmodels** (R. Sonabend 2020) package,[2] implements these Python models via **reticulate** (Ushey, Allaire, and Tang 2020). No recurrent neural netwoks are included in this survey though the survival models SRN (Oh et al. 2018) and RNN-Surv (Giunchiglia, Nemchenko, and Schaar 2018) are acknowledged.

This survey is made slightly more difficult as neural networks are often proposed for many different tasks, which are not necessarily clearly advertised in a paper's title or abstract. For example, many papers claim to use neural networks for survival analysis and make comparisons to Cox models, whereas the task tends to be death at a particular (usually 5-year) time-point (classification) (I. Han et al. 2018; Lundin et al. 1999; B. D. Ripley and Ripley 2001; R. M. Ripley, Harris, and Tarassenko 1998; Huseyin Seker et al. 2002), which is often not made clear until mid-way through the paper. Reviews and surveys have also conflated these different tasks, for example a very recent review concluded superior performance of ANNs over Cox models, when in fact this is only in classification (Huang et al. 2020a) (RM2) {sec:car_reduxstrats_mistakes}. To clarify, this form of classification task does fall into the general *field* of survival analysis, but not the survival *task* ((**box-task-surv?**)). Therefore this is not a comment on the classification task but a reason for omitting these models from this survey.

Using ANNs for feature selection (often in gene expression data) and computer vision is also very common in survival analysis, and indeed it is in this area that most success has been seen (Bello et al. 2019; Y.-C. Chen, Ke, and Chiu 2014; Cui et al. 2020; Lao et al. 2017; McKinney et al. 2020; Rietschel, Yoon, and Schaar 2018; H. Seker et al. 2002; Yucheng Zhang et al. 2020; X. Zhu, Yao, and Huang 2016), but these are again beyond the scope of this survey.

The key difference between neural networks is in their output layer, required data transformations, the model prediction, and the loss function used to fit the model. Therefore the following are discussed for each of the surveyed models: the loss function for training, L, the model prediction type, $\hat{g}$, and any required data transformation. Notation is continued from the previous surveys with the addition of θ denoting model weights (which will be different for each model).

15.0.2.1 Probabilistic Survival Models

Unlike other classes of machine learning models, the focus in ANNs has been on probabilistic models. The vast majority make these predictions via reduction to binary classification **??**. Whilst almost all of these networks implicitly reduce the problem to classification, most are not transparent in exactly how they do so and none provide clear or detailed interface

[1] Schwarzer conflates the prognosis and survival task, therefore it is not clear if all 10 of these are for time-to-event data (at least five definitely are).

[2] Created in order to run the experiments in [@Sonabend2021b].

points in implementation allowing for control over this reduction. Most importantly, the majority of these models do not detail how valid survival predictions are derived from the binary setting,[3] which is not just a theoretical problem as some implementations, such as the Logistic-Hazard model in **pycox** (Kvamme 2018), have been observed to make survival predictions outside the range $[0, 1]$. This is not a statement about the performance of models in this section but a remark about the lack of transparency across all probabilistic ANNs.

Many of these algorithms use an approach that formulate the Cox PH as a non-linear model and minimise the partial likelihood. These are referred to as 'neural-Cox' models and the earliest appears to have been developed by Faraggi and Simon (Faraggi and Simon 1995). All these models are technically composites that first predict a ranking, however they assume a PH form and in implementation they all appear to return a probabilistic prediction.

ANN-COX {#mod-anncox}\ Faraggi and Simon (Faraggi and Simon 1995) proposed a non-linear PH model

$$h(\tau|X_i, \theta) = h_0(\tau)\exp(\phi(X_i\beta)) \tag{15.4}$$

where ϕ is the sigmoid function and $\theta = \{\beta\}$ are model weights. This model, 'ANN-COX', estimates the prediction functional, $\hat{g}(X^*) = \phi(X^*\hat{\beta})$. The model is trained with the partial-likelihood function

$$L(\hat{g}, \theta|\mathcal{D}_{train}) = \prod_{i=1}^{n} \frac{\exp(\sum_{m=1}^{M} \alpha_m \hat{g}_m(X^*))}{\sum_{j\in\mathcal{R}_{t_i}} \exp(\sum_{m=1}^{M} \alpha_m \hat{g}_m(X^*))}$$

where $\mathcal{R}_{t_i}$ is the risk group alive at t_i; M is the number of hidden units; $\hat{g}_m(X^*) = (1 + \exp(-X^*\hat{\beta}_m))^{-1}$; and $\theta = \{\beta, \alpha\}$ are model weights.

The authors proposed a single hidden layer network, trained using back-propagation and weight optimisation with Newton-Raphson. This architecture did not outerperform a Cox PH (Faraggi and Simon 1995). Further adjustments including (now standard) pre-processing and hyper-parameter tuning did not improve the model performance (Mariani et al. 1997). Further independent studies demonstrated worse performance than the Cox model (Faraggi and Simon 1995; Xiang et al. 2000).

COX-NNET {#mod-coxnnet}\ COX-NNET (Ching, Zhu, and Garmire 2018) updates the ANN-COX by instead maximising the regularized partial log-likelihood

$$L(\hat{g}, \theta|\mathcal{D}_{train}, \lambda) = \sum_{i=1}^{n} \Delta_i \Big[\hat{g}(X_i) \; - \; \log\Big(\sum_{j\in\mathcal{R}_{t_i}} \exp(\hat{g}(X_j))\Big)\Big] + \lambda(\|\beta\|_2 + \|w\|_2)$$

with weights $\theta = (\beta, w)$ and where $\hat{g}(X_i) = \sigma(wX_i + b)^T\beta$ for bias term b, and activation function σ; σ is chosen to be the tanh function ((15.1)). In addition to weight decay, dropout (Srivastava et al. 2014) is employed to prevent overfitting. Dropout can be thought of as a similar concept to the variable selection in random forests, as each node is randomly deactivated with probability p, where p is a hyper-parameter to be tuned.

Independent simulation studies suggest that COX-NNET does not outperform the Cox PH (Michael F. Gensheimer and Narasimhan 2019).

DeepSurv {#mod-deepsurv}\ DeepSurv (J. L. Katzman et al. 2018) extends these models to deep learning with multiple hidden layers. The chosen error function is the average

[3] One could assume they use procedures such as those described in Tutz and Schmid (2016) [@Tutz2016] but there is rarely transparent writing to confirm this.

negative log-partial-likelihood with weight decay

$$L(\hat{g}, \theta | \mathcal{D}_{train}, \lambda) = -\frac{1}{n^*} \sum_{i=1}^{n} \Delta_i \Big[\Big(\hat{g}(X_i) - \log \sum_{j \in \mathcal{R}_{t_i})} \exp(\hat{g}(X_j) \Big) \Big] + \lambda \|\theta\|_2^2$$

where $n^* := \sum_{i=1}^{n} \mathbb{I}(\Delta_i = 1)$ is the number of uncensored observations and $\hat{g}(X_i) = \phi(X_i|\theta)$ is the same prediction object as the ANN-COX. State-of-the-art methods are used for data pre-processing and model training. The model architecture uses a combination of fully-connected and dropout layers. Benchmark experiments by the authors indicate that DeepSurv can outperform the Cox PH in ranking tasks (J. Katzman et al. 2016; J. L. Katzman et al. 2018) although independent experiments do not confirm this (Zhao and Feng 2020).

Cox-Time {#mod-coxtime}\ Kvamme *et al.* (Kvamme, Borgan, and Scheel 2019) build on these models by allowing time-varying effects. The loss function to minimise, with regularization, is given by

$$L(\hat{g}, \theta | \mathcal{D}_{train}, \lambda) = \frac{1}{n} \sum_{i:\Delta_i=1} \log \Big(\sum_{j \in \mathcal{R}_{t_i}} \exp[\hat{g}(X_j, T_i) - \hat{g}(X_i, T_i)] \Big) + \lambda \sum_{i:\Delta_i=1} \sum_{j \in \mathcal{R}_{t_i}} |\hat{g}(X_j, T_i)|$$

where $\hat{g} = \hat{g}_1, ..., \hat{g}_n$ is the same non-linear predictor but with a time interaction and λ is the regularization parameter. The model is trained with stochastic gradient descent and the risk set, $\mathcal{R}_{t_i}$, in the equation above is instead reduced to batches, as opposed to the complete dataset. ReLU activations (Nair and Hinton 2010) and dropout are employed in training. Benchmark experiments indicate good performance of Cox-Time, though no formal statistical comparisons are provided and hence no comment about general performance can be made.

ANN-CDP {#mod-anncdp}\ One of the earliest ANNs that was noted by Schwarzer *et al.* (Schwarzer, Vach, and Schumacher 2000) was developed by Liestøl *et al.* (Liestol, Andersen, and Andersen 1994) and predicts conditional death probabilities (hence 'ANN-CDP'). The model first partitions the continuous survival times into disjoint intervals $\mathcal{I}_k, k = 1, ..., m$ such that $\mathcal{I}_k$ is the interval $(t_{k-1}, t_k]$. The model then studies the logistic Cox model (proportional odds) (Cox 1972) given by

$$\frac{p_k(\mathbf{x})}{q_k(\mathbf{x})} = \exp(\eta + \theta_k)$$

where $p_k = 1 - q_k$, $\theta_k = \log(p_k(0)/q_k(0))$ for some baseline probability of survival, $q_k(0)$, to be estimated; η is the usual linear predictor, and $q_k = P(T \geq T_k | T \geq T_{k-1})$ is the conditional survival probability at time T_k given survival at time T_{k-1} for $k = 1, ..., K$ total time intervals. A logistic activation function is used to predict $\hat{g}(X^*) = \phi(\eta + \theta_k)$, which provides an estimate for $\hat{p}_k$.

The model is trained on discrete censoring indicators D_{ki} such that $D_{ki} = 1$ if individual i dies in interval $\mathcal{I}_k$ and 0 otherwise. Then with K output nodes and maximum likelihood estimation to find the model parameters, $\hat{\eta}$, the final prediction provides an estimate for the conditional death probabilities $\hat{p}_k$. The negative log-likelihood to optimise is given by

$$L(\hat{g}, \theta | \mathcal{D}_{train}) = \sum_{i=1}^{n} \sum_{k=1}^{m_i} [D_{ki} \log(\hat{p}_k(X_i)) + (1 - D_{ki}) \log(\hat{q}_k(X_i))]$$

where m_i is the number of intervals in which observation i is not censored.

Liestøl *et al.*{} discuss different weighting options and how they correspond to the PH assumption. In the most generalised case, a weight-decay type regularization is applied to the model weights given by

$$\alpha \sum_l \sum_k (w_{kl} - w_{k-1,l})^2$$

where w are weights, and α is a hyper-parameter to be tuned, which can be used alongside standard weight decay. This corresponds to penalizing deviations from proportionality thus creating a model with approximate proportionality. The authors also suggest the possibility of fixing the weights to be equal in some nodes and different in others; equal weights strictly enforces the proportionality assumption. Their simulations found that removing the proportionality assumption completely, or strictly enforcing it, gave inferior results. Comparing their model to a standard Cox PH resulted in a 'better' negative log-likelihood, however this is not a precise evaluation metric and an independent simulation would be

preferred. Finally Listøl *et al.* included a warning "The flexibility is, however, obtained at unquestionable costs: many parameters, difficult interpretation of the parameters and a slow numerical procedure' ' (Liestol, Andersen, and Andersen 1994).

PLANN {#mod-plann}\ Biganzoli *et al.* (1998) (E. Biganzoli et al. 1998) studied the same proportional-odds model as the ANN-CDP (Liestol, Andersen, and Andersen 1994). Their model utilises partial logistic regression (Efron 1988) with added hidden nodes, hence 'PLANN'. Unlike ANN-CDP, PLANN predicts a smoothed hazard function by using smoothing splines. The continuous time outcome is again discretised into disjoint intervals $t_m, m = 1, ..., M$. At each time-interval, t_m, the number of events, d_m, and number of subjects at risk, n_m, can be used to calculate the discrete hazard function,[4]

$$\hat{h}_m = \frac{d_m}{n_m}, m = 1, ..., M \tag{15.5}$$

This quantity is used as the target to train the neural network. The survival function is then estimated by the Kaplan-Meier type estimator,

$$\hat{S}(\tau) = \prod_{m:t_m \leq \tau} (1 - \hat{h}_m) \tag{15.6}$$

The model is fit by employing one of the more 'usual' survival reduction strategies in which an observation's survival time is treated as a covariate in the model (Tutz and Schmid 2016). As this model uses discrete time, the survival time is discretised into one of the M intervals. This approach removes the proportional odds constraint as interaction effects between time and covariates can be modelled (as time-updated covariates). Again the model makes predictions at a given time m, $\phi(\theta_m + \eta)$, where η is the usual linear predictor, θ is the baseline proportional odds hazard $\theta_m = \log(h_m(0)/(1 - h_m(0))$. The logistic activation provides estimates for the discrete hazard,

$$h_m(X_i) = \frac{\exp(\theta_m + \hat{\eta})}{1 + \exp(\theta_m + \hat{\eta})}$$

which is smoothed with cubic splines (Efron 1988) that require tuning.

A cross-entropy error function is used for training

$$L(\hat{h}, \theta | \mathcal{D}_{train}, a) = -\sum_{m=1}^{M} \Big[\hat{h}_m \log\Big(\frac{h_l(X_i, a_l)}{\hat{h}_m}\Big) + (1 - \hat{h}_m) \log\Big(\frac{1 - h_l(X_i, a_l)}{1 - \hat{h}_m}\Big)\Big] n_m$$

where $h_l(X_i, a_l)$ is the discrete hazard h_l with smoothing at mid-points a_l. Weight decay can be applied and the authors suggest $\lambda \approx 0.01 - 0.1$ (E. Biganzoli et al. 1998), though they make use of an AIC type criterion instead of cross-validation.

This model makes smoothed hazard predictions at a given time-point, τ, by including τ in the input covariates X_i. Therefore the model first requires transformation of the input data by replicating all observations and replacing the single survival indicator Δ_i, with a time-dependent indicator D_{ik}, the same approach as in ANN-CDP. Further developments have extended the PLANN to Bayesian modelling, and for competing risks (E. M. Biganzoli, Ambrogi, and Boracchi 2009).

No formal comparison is made to simpler model classes. The authors recommend ANNs primarily for exploration, feature selection, and understanding underlying patterns in the data (E. M. Biganzoli, Ambrogi, and Boracchi 2009).

[4]Derivation of this as a 'hazard' estimator follows trivially by comparison to the Nelson-Aalen estimator.

Nnet-survival {#mod-nnetsurvival}\ Aspects of the PLANN algorithm have been generalised into discrete-time survival algorithms in several papers (Michael F. Gensheimer and Narasimhan 2019; **Kvamme2019?**; Mani et al. 1999; Street 1998). Various estimates have been derived for transforming the input data to a discrete hazard or survival function. Though only one is considered here as it is the most modern and has a natural interpretation as the 'usual' Kaplan-Meier estimator for the survival function. Others by Street (1998) (Street 1998) and Mani (1999) (Mani et al. 1999) are acknowledged. The discrete hazard estimator (15.5), $\hat{h}$, is estimated and these values are used as the targets for the ANN. For the error function, the mean negative log-likelihood for discrete time (**Kvamme2019?**) is minimised to estimate $\hat{h}$,

$$L(\hat{h}, \theta | \mathcal{D}_{train}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k(T_i)} (\mathbb{I}(T_i = \tau_j, \Delta_i = 1) \log[\hat{h}_i(\tau_j)] + (1 - \mathbb{I}(T_i = \tau_j, \Delta_i = 1)) \log(1 - \hat{h}_i(\tau_j)))$$

where $k(T_i)$ is the time-interval index in which observation i dies/is censored, τ_j is the jth discrete time-interval, and the prediction of $\hat{h}$ is obtained via

$$\hat{h}(\tau_j | \mathcal{D}_{train}) = [1 + \exp(-\hat{g}_j(\mathcal{D}_{train}))]^{-1}$$

where $\hat{g}_j$ is the jth output for $j = 1, ..., m$ discrete time intervals. The number of units in the output layer for these models corresponds to the number of discrete-time intervals. Deciding the width of the time-intervals is an additional hyper-parameter to consider.

Gensheimer and Narasimhan's 'Nnet-survival' (Michael F. Gensheimer and Narasimhan 2019) has two different implementations. The first assumes a PH form and predicts the linear predictor in the final layer, which can then be composed to a distribution. Their second 'flexible' approach instead predicts the log-odds of survival in each node, which are then converted to a conditional probability of survival, $1 - h_j$, in a given interval using the sigmoid activation function. The full survival function can be derived with (15.6). The model has been demonstrated not to outperform the Cox PH with respect to Harrell's C or the Graf (Brier) score (Michael F. Gensheimer and Narasimhan 2019).

PC-Hazard {#mod-pchazard}\ Kvamme and Borgan deviate from nnet-survival in their 'PC-Hazard' (**Kvamme2019?**) by first considering a discrete-time approach with a softmax activation function influenced by multi-class classification. They expand upon this by studying a piecewise constant hazard function in continuous time and defining the mean negative log-likelihood as

$$L(\hat{g}, \theta | \mathcal{D}_{train}) = -\frac{1}{n} \sum_{i=1}^{n} \left(\Delta_i X_i \log \tilde{\eta}_{k(T_i)} - X_i \tilde{\eta}_{k(T_i)} \rho(T_i) - \sum_{j=1}^{k(T_i)-1} \tilde{\eta}_j X_i \right)$$

where $k(T_i)$ and τ_i is the same as defined above, $\rho(t) = \frac{t - \tau_{k(t)-1}}{\Delta \tau_{k(t)}}$, $\Delta \tau_j = \tau_j - \tau_{j-1}$, and $\tilde{\eta}_j := \log(1 + \exp(\hat{g}_j(X_i))$ where again $\hat{g}_j$ is the jth output for $j = 1, ..., m$ discrete time intervals. Once the weights have been estimated, the predicted survival function is given by

$$\hat{S}(\tau, X^* | \mathcal{D}_{train}) = \exp(-X^* \tilde{\eta}_{k(\tau)} \rho(\tau)) \prod_{j=1}^{k(\tau)-1} \exp(-\tilde{\eta}_j(X^*))$$

Benchmark experiments indicate similar performance to nnet-survival (**Kvamme2019?**), an unsurprising result given their implementations are identical with the exception of the

loss function (**Kvamme2019?**), which is also similar for both models. A key result found that varying values for interval width lead to significant differences and therefore should be carefully tuned.

DNNSurv {#mod-dnnsurv}\ A very recent (pre-print) approach (Zhao and Feng 2020) instead first computes 'pseudo-survival probabilities' and uses these to train a regression ANN with sigmoid activation and squared error loss. These pseudo-probabilities are computed using a jackknife-style estimator given by

$$\tilde{S}_{ij}(T_{j+1}, \mathcal{R}_{t_j}) = n_j \hat{S}(T_{j+1}|\mathcal{R}_{t_j}) - (n_j - 1)\hat{S}^{-i}(T_{j+1}|\mathcal{R}_{t_j})$$

where $\hat{S}$ is the IPCW weighted Kaplan-Meier estimator (defined below) for risk set $\mathcal{R}_{t_j}$, $\hat{S}^{-i}$ is the Kaplan-Meier estimator for all observations in $\mathcal{R}_{t_j}$ excluding observation i, and $n_j := |\mathcal{R}_{t_j}|$. The IPCW weighted Kaplan-Meier estimate is found via the IPCW Nelson-Aalen estimator,

$$\hat{H}(\tau|\mathcal{D}_{train}) = \sum_{i=1}^{n} \int_0^{\tau} \frac{\mathbb{I}(T_i \leq u, \Delta_i = 1)\hat{W}_i(u)}{\sum_{j=1}^{n} \mathbb{I}(T_j \geq u)\hat{W}_j(u)} \, du$$

where $\hat{W}_i, \hat{W}_j$ are subject specific IPC weights.

In their simulation studies, they found no improvement over other proposed neural networks. Arguably the most interesting outcome of their paper are comparisons of multiple survival ANNs at specific time-points, evaluated with C-index and Brier score. Their results indicate identical performance from all models. They also provide further evidence of neural networks not outperforming a Cox PH when the PH assumption is valid. However, in their non-PH dataset, DNNSurv appears to outperform the Cox model (no formal tests are provided). Data is replicated similarly to previous models except that no special indicator separates censoring and death, this is assumed to be handled by the IPCW pseudo probabilities.

DeepHit {#mod-deephit}\ DeepHit (C. Lee et al. 2018) was originally built to accommodate competing risks, but only the non-competing case is discussed here (Kvamme, Borgan, and Scheel 2019). The model builds on previous approaches by discretising the continuous time outcome, and makes use of a composite loss. It has the advantage of making no parametric assumptions and directly predicts the probability of failure in each time-interval (which again correspond to different terminal nodes), i.e. $\hat{g}(\tau_k|\mathcal{D}_{test}) = \hat{P}(T^* = \tau_k|X^*)$ where again $\tau_k, k = 1, ..., K$ are the distinct time intervals. The estimated survival function is found with $\hat{S}(\tau_K|X^*) = 1 - \sum_{k=1}^{K} \hat{g}_i(\tau_k|X^*)$. ReLU activations were used in all fully connected layers and a softmax activation in the final layer. The losses in the composite error function are given by

$$L_1(\hat{g}, \theta|\mathcal{D}_{train}) = -\sum_{i=1}^{N} [\Delta_i \log(\hat{g}_i(T_i)) + (1 - \Delta_i) \log(\hat{S}_i(T_i))]$$

and

$$L_2(\hat{g}, \theta|\mathcal{D}_{train}, \sigma) = \sum_{i \neq j} \Delta_i \mathbb{I}(T_i < T_j)\sigma(\hat{S}_i(T_i), \hat{S}_j(T_i))$$

for some convex loss function σ and where $\hat{g}_i(t) = \hat{g}(t|X_i)$. Again these can be seen to be a cross-entropy loss and a ranking loss. Benchmark experiments demonstrate the model outperforming the Cox PH and RSFs (C. Lee et al. 2018) with respect to separation, and an independent experiment supports these findings (Kvamme, Borgan, and Scheel 2019). However, the same independent study demonstrated worse performance than a Cox PH with respect to the integrated Brier score (Graf et al. 1999).

15.0.2.2 Deterministic Survival Models

Whilst the vast majority of survival ANNs have focused on probabilistic predictions (often via ranking), a few have also tackled the deterministic or 'hybrid' problem.

RankDeepSurv {#mod-rankdeepsurv}\ Jing *et al.* (Jing et al. 2019) observed the past two decades of research in survival ANNs and then published a completely novel solution, RankDeepSurv, which makes predictions for the survival time $\hat{T} = (\hat{T}_1, ..., \hat{T}_n)$. They proposed a composite loss function

$$L(\hat{T}, \theta | \mathcal{D}_{train}, \alpha, \gamma, \lambda) = \alpha L_1(\hat{T}, T, \Delta) + \gamma L_2(\hat{T}, T, \Delta) + \lambda \|\theta\|_2^2$$

where θ are the model weights, $\alpha, \gamma \in \mathbb{R}_{>0}$, λ is the shrinkage parameter, by a slight abuse of notation $T = (T_1, ..., T_n)$ and $\Delta = (\Delta_1, ..., \Delta_n)$, and

$$L_1(\hat{T}, \theta | \mathcal{D}_{train}) = \frac{1}{n} \sum_{\{i: I(i)=1\}} (\hat{T}_i - T_i)^2; \quad I(i) = \begin{cases} 1, & \Delta_i = 1 \cup (\Delta_i = 0 \cap \hat{T}_i \leq T_i) \\ 0, & \text{otherwise} \end{cases}$$

$$L_2(\hat{T}, \theta | \mathcal{D}_{train}) = \frac{1}{n} \sum_{\{i,j: I(i,j)=1\}}^{n} [(T_j - T_i) - (\hat{T}_j - \hat{T}_i)]^2; \quad I(i,j) = \begin{cases} 1, & T_j - T_i > \hat{T}_j - \hat{T}_i \\ 0, & \text{otherwise} \end{cases}$$

where $\hat{T}_i$ is the predicted survival time for observation i. A clear contrast can be made between these loss functions and the constraints used in SSVM-Hybrid (Van Belle et al. 2011) (Section 13.2). L_1 is the squared second constraint in 13.2.3 and L_2 is the squared first constraint in 13.2.3. However L_1 in RankDeepSurv discards the squared error difference for all censored observations when the prediction is lower than the observed survival time; which is problematic as if someone is censored at time T_i then it is guaranteed that their true survival time is greater than T_i (this constraint may be more sensible if the inequality were reversed). An advantage to this loss is, like the SSVM-Hybrid, it enables a survival time interpretation for a ranking optimised model; however these 'survival times' should be interpreted with care.

The authors propose a model architecture with several fully connected layers with the ELU (Clevert, Unterthiner, and Hochreiter 2015) activation function and a single dropout layer. Determining the success of this model is not straightforward. The authors claim superiority of RankDeepSurv over Cox PH, DeepSurv, and RSFs however this is an unclear comparison (RM2) {sec:car_reduxstrats_mistakes} that requires independent study.

15.0.3 Conclusions

There have been many advances in neural networks for survival analysis. It is not possible to review all proposed survival neural networks without diverting too far from the book scope. This survey of ANNs should demonstrate two points: firstly that the vast majority (if not all) of survival ANNs are reduction models that either find a way around censoring via imputation or discretisation of time-intervals, or by focusing on partial likelihoods only; secondly that no survival ANN is fully accessible or transparent.

Despite ANNs being highly performant in other areas of supervised learning, there is strong evidence that the survival ANNs above are inferior to a Cox PH when the data follows the PH assumption or when variables are linearly related (Michael F. Gensheimer and Narasimhan 2018; Luxhoj and Shyur 1997; Ohno-Machado 1997; Puddu and Menotti 2012; Xiang et al. 2000; Yang 2010; Yasodhara, Bhat, and Goldenberg 2018; Zhao and Feng 2020). There are not enough experiments to make conclusions in the case when the data is non-PH.

Experiments in (R. E. B. Sonabend 2021) support the finding that survival ANNs are not performant.

There is evidence that many papers introducing neural networks do not utilise proper methods of comparison or evaluation (Király, Mateen, and Sonabend 2018) and in conducting this survey, these findings are further supported. Many papers made claims of being 'superior' to the Cox model based on unfair comparisons (RM2){sec:car_reduxstrats_mistakes} or miscommunicating (or misinterpreting) results (e.g. (Fotso 2018)). At this stage, it does not seem possible to make any conclusions about the effectiveness of neural networks in survival analysis. Moreover, even the authors of these models have pointed out problems with transparency (E. M. Biganzoli, Ambrogi, and Boracchi 2009; Liestol, Andersen, and Andersen 1994), which was further highlighted by Schwarzer *et al.* (Schwarzer, Vach, and Schumacher 2000).

Finally, accessibility of neural networks is also problematic. Many papers do not release their code and instead just state their networks architecture and available packages. In theory, this is enough to build the models however this does not guarantee the reproducibility that is usually expected. For users with a technical background and good coding ability, many of the models above could be implemented in one of the neural network packages in R, such as **nnet** (N. Venables and D. Ripley 2002) and **neuralnet** (Fritsch, Guenther, and N. Wright 2019); though in practice the only package that does contain these models, **survivalmodels**, does not directly implement the models in R (which is much slower than Python) but provides a method for interfacing the Python implementations in **pycox** (Kvamme 2018).

Further reading

- Schwarzer, Vach, and Schumacher (2000) provided an early survey of neural networks, focusing on ways in which neural networks have been 'misused' in the context of survival analysis. Whilst neural networks have moved on substantially since, their early observations remain valid today.

Part IV

Reduction Techniques

16

Reductions

TODO (150-200 WORDS)

> ! Major changes expected!
>
> **This page is a work in progress and major changes will be made over time.**

In this chapter, composition and reduction are formally introduced, defined and demonstrated within survival analysis. Neither of these are novel concepts in general or in survival, with several applications already seen earlier when reviewing models (particularly in neural networks), however a lack of formalisation has led to much repeated work and at times questionable applications (Chapter 15). The primary purpose of this chapter is to formalise composition and reduction for survival and to unify references and strategies for future use. These strategies are introduced in the context of minimal 'workflows' and graphical 'pipelines' in order to maximise their generalisability.

A *workflow* is a generic term given to a series of sequential operations. For example a standard ML workflow is fit/predict/evaluate, which means a model is fit, predictions are made, and these are evaluated. In this book, a *pipeline* is the name given to a concrete workflow. Section 16.1 demonstrates how pipelines are represented in this book.

Composition (Section 16.2) is a general process in which an object is built (or composed) from other objects and parameters. Reduction (Section 16.3) is a closely related concept that utilises composition in order to transform one problem into another. Concrete strategies for composition and reduction are detailed in sections Section 16.4 and Section 16.5.

Notation and Terminology

The notation introduced in Chapter 4 is recapped for use in this chapter: the generative survival template for the survival setting is given by (X, T, Δ, Y, C) *t.v.i.* $\mathcal{X}\times\mathcal{T}\times\{0,1\}\times\mathcal{T}\times\mathcal{T}$ where $\mathcal{X} \subseteq \mathbb{R}^p$ and $\mathcal{T} \subseteq \mathbb{R}_{\geq 0}$, where C, Y are unobservable, $T := \min\{Y, C\}$, and $\Delta = \mathbb{I}(Y = T)$. Random survival data is given by $(X_i, T_i, \Delta_i, Y_i, C_i) \overset{i.i.d.}{\sim} (X, T, \Delta, Y, C)$. Usually data will instead be presented as a training dataset, $\mathcal{D}_{train} = \{(X_1, T_1, \Delta_1), ..., (X_n, T_n, \Delta_n)\}$ where $(X_i, T_i, \Delta_i) \overset{i.i.d.}{\sim} (X, T, \Delta)$, and some test data $\mathcal{D}_{test} = (X^*, T^*, \Delta^*) \sim (X, T, \Delta)$.

For regression models the generative template is given by (X, Y) t.v.i. $\mathcal{X} \subseteq \mathbb{R}^p$ and $Y \subseteq \mathbb{R}$. As with the survival setting, a regression training set is given by $\{(X_1, Y_1), ..., (X_n, Y_n)\}$ where $(X_i, Y_i) \overset{i.i.d.}{\sim} (X, Y)$ and some test data $(X^*, Y^*) \sim (X, Y)$.

16.1 Representing Pipelines

Before introducing concrete composition and reduction algorithms, this section briefly demonstrates how these pipelines will be represented in this book.

Pipelines are represented by graphs designed in the following way: all are drawn with operations progressing sequentially from left to right; graphs are comprised of nodes (or 'vertices') and arrows (or 'directed edges'); a rounded rectangular node represents a process such as a function or model fitting/predicting; a (regular) rectangular node represents objects such as data or hyper-parameters. Output from rounded nodes are sometimes explicitly drawn but when omitted the output from the node is the input to the next.

These features are demonstrated in **?@fig-car-example**. Say $y = 2$ and $a = 2$, then: data is provided ($y = 2$) and passed to the shift function ($f(x) = x + 2$), the output of this function ($y = 4$) is passed directly to the next ($h(x|a) = x^a$), this function requires a parameter which is also input ($a = 2$), finally the resulting output is returned ($y^* = 16$). Programmatically, $a = 2$ would be a hyper-parameter that is stored and passed to the required function when the function is called.

This pipeline is represented as a pseudo-algorithm in (**alg-car-ex?**), though of course is overly complicated and in practice one would just code $(y + 2)\hat{}a$.

16.2 Introduction to Composition

This section introduces composition, defines a taxonomy for describing compositors (Section 16.2.1), and provides some motivating examples of composition in survival analysis (Section 16.2.2).

In the simplest definition, a model (be it mathematical, computational, machine learning, etc.) is called a *composite model* if it is built of two or more constituent parts. This can be simplest defined in terms of objects. Just as objects in the real-world can be combined in some way, so can mathematical objects. The exact 'combining' process (or 'compositor') depends on the specific composition, so too do the inputs and outputs. By example, a wooden table can be thought of as a composite object (Figure 16.1). The inputs are wood and nails, the combining process is hammering (assuming the wood is pre-chopped), and the output is a surface for eating. In mathematics, this process is mirrored. Take the example of a shifted linear regression model. This is defined by a linear regression model, $f(x) = \beta_0 + x\beta_1$, a shifting parameter, α, and a compositor $g(x|\alpha) = f(x) + \alpha$. Mathematically this example is overly trivial as this could be directly modelled with $f(x) = \alpha + \beta_0 + x\beta_1$, but algorithmically there is a difference. The composite model g, is defined by first fitting the linear regression model, f, and then applying a shift, α; as opposed to fitting a directly shifted model.

Why Composition?

Tables tend to be better surfaces for eating your dinner than bundles of wood. Or in modelling terms, it is well-known that ensemble methods (e.g. random forests) will generally outperform their components (e.g. decision trees). All ensemble methods are composite models and this demonstrates one of the key use-cases of composition: improved predictive

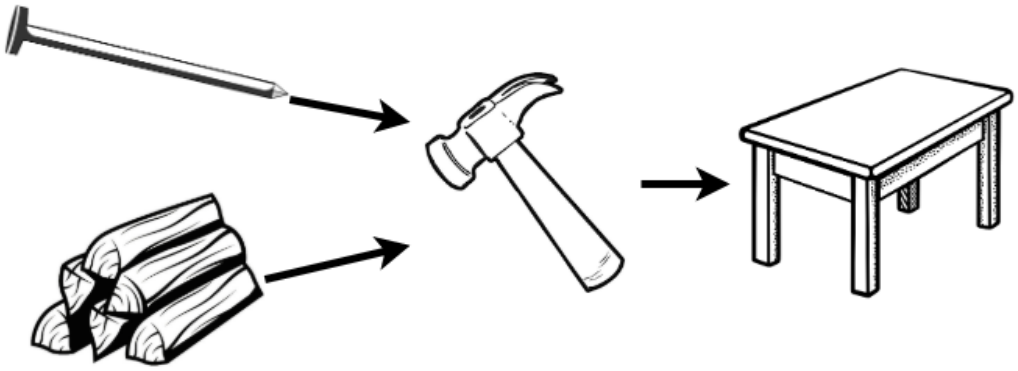


Figure 16.1: Visualising composition in the real-world. A table is a composite object built from nails and wood, which are combined with a hammer 'compositor'. Figure not to scale.

performance. The second key use-case is reduction, which is fully discussed in Section 16.3. Section 16.2.2 motivates composition in survival analysis by demonstrating how it is already prevalent but requires formalisation to make compositions more transparent and accessible.

Composite Model vs. Sub-models

A bundle of wood and nails is not a table and $1,000$ decision trees are not a random forest, both require a compositor. The compositor in a composite model combines the components into a single model. Considering a composite model as a single model enables the hyper-parameters of the compositor and the component model(s) to be efficiently tuned whilst being evaluated as a single model. This further allows the composite to be compared to other models, including its own components, which is required to justify complexity instead of parsimony in model building (**?@sec-eval-why-why**).

16.2.1 Taxonomy of Compositors

Just as there are an infinite number of ways to make a table, composition can come in infinite forms. However there are relatively few categories that these can be grouped into. Two primary taxonomies are identified here. The first is the 'composition type' and relates to the number of objects composed:

[i)] i. Single-Object Composition (SOC) – This form of composition either makes use of parameters or a transformation to alter a single object. The shifted linear regression model above is one example of this, another is given in Section 16.4.3. i. Multi-Object Composition (MOC) – In contrast, this form of composition combines multiple objects into a single one. Both examples in Section 16.2.2 are multi-object compositions.

The second grouping is the 'composition level' and determines at what 'level' the composition takes place:

[i)] i. Prediction Composition – This applies at the level of predictions; the component models could be forgotten at this point. Predictions may be combined from multiple models

(MOC) or transformed from a single model (SOC). Both examples in Section 16.2.2 are prediction compositions. i. Task Composition – This occurs when one task (e.g. regression) is transformed to one or more others (e.g. classification), therefore always SOC. This is seen mainly in the context of reduction (Section 16.3). i. Model Composition – This is commonly seen in the context of wrappers (Section 16.5.1.4), in which one model is contained within another. i. Data Composition – This is transformation of training/testing data types, which occurs at the first stage of every pipeline.

16.2.2 Motivation for Composition

Two examples are provided below to demonstrate common uses of composition in survival analysis and to motivate the compositions introduced in Section 16.4.

Example 1: Cox Proportional Hazards

Common implementations of well-known models can themselves be viewed as composite models, the Cox PH is the most prominent example in survival analysis. Recall the model defined by

$$h(\tau|X_i) = h_0(\tau)\exp(\beta X_i)$$

where h_0 is the baseline hazard and β are the model coefficients.

This can be seen as a composite model as Cox defines the model in two stages (Cox 1972): first fitting the β-coefficients using the partial likelihood and then by suggesting an estimate for the baseline distribution. This first stage produces a linear predictor return type (Chapter 6) and the second stage returns a survival distribution prediction. Therefore the Cox model for linear predictions is a single (non-composite) model, however when used to make distribution predictions then it is a composite. Cox implicitly describes the model as a composite by writing ''alternative simpler procedures would be worth having'' (Cox 1972), which implies a decision in fitting (a key feature of composition). This composition is formalised in Section 16.4.1 as a general pipelins. The Cox model utilises the pipeline with a PH form and Kaplan-Meier baseline.

Example 2: Random Survival Forests

Fully discussed in Chapter 12, random survival forests are composed from many individual decision trees via a prediction composition algorithm ((**alg-rsf-pred?**)). In general, random forests perform better than their component decision trees, which tends to be true of all ensemble methods. Aggregation of predictions in survival analysis requires slightly more care than other fields due to the multiple prediction types, however this is still possible and is formalised in Section 16.4.4.

16.3 Introduction to Reduction

This section introduces reduction, motivates its use in survival analysis (Section 16.3.1), details an abstract reduction pipeline and defines the difference between a complete/incomplete reduction (Section 16.3.2), and outlines some common mistakes that have been observed in the literature when applying reduction (Section 16.3.3).

Reduction is a concept found across disciplines with varying definitions. This report uses the Langford definition: reduction is ‘’a complex problem decomposed into simpler subproblems so that a solution to the subproblems gives a solution to the complex problem’ ’ (Langford et al. 2016). Generalisation (or induction) is a common real-world use of reduction, for example sampling a subset of a population in order to estimate population-level results. The true answer (population-level values) may not always be found in this way but very good approximations can be made with simpler sub-problems (sub-sampling).

Reductions are workflows that utilise composition. By including hyper-parameters, even complex reduction strategies can remain relatively flexible. To illustrate reduction by example, recall the table-building example (Section 16.2) in which the task of interest is to acquire a table. The most direct but complex solution is to fell a tree and directly saw it into a table (Figure 16.2, top), clearly this is not a sensible process. Instead the problem can be reduced into simpler sub-problems: saw the tree into bundles of wood, acquire nails, and then use the ‘hammer compositor’ (Figure 16.1) to create a table (Figure 16.2, bottom).

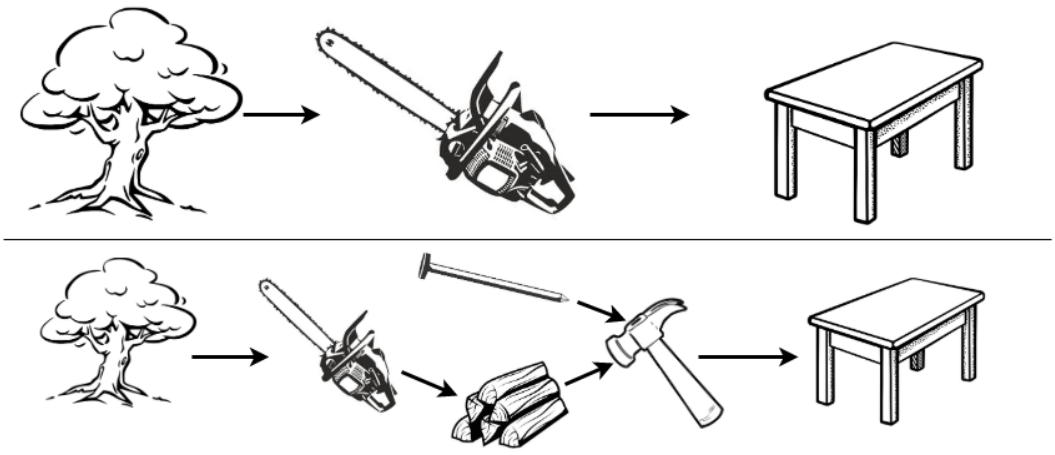


Figure 16.2: Visualising reduction in the real-world. The complex process (top) of directly sawing a tree into a table is inefficient and unnecessarily complex. The reduction (bottom) that involves first creating bundles of wood is simpler, more efficient, and yields the same result, though technically requiring more steps.

In a modelling example, predicting a survival distribution with the Cox model can be viewed as a reduction in which two sub-problems are solved and composed:

i. predict continuous ranking;
ii. estimate baseline hazard; and
iii. compose with (Section 16.4.1).

This is visualised as a reduction strategy in **?@fig-car-cargraph**. The entire process from defining the original problem, to combining the simpler sub-solutions (in green), is the reduction (in red).

16.3.1 Reduction Motivation

Formalisation of reduction positively impacts upon accessibility, transparency, and predictive performance. Improvements to predictive performance have already been demonstrated when comparing random forests to decision trees. In addition, a reduction with multiple stages and many hyper-parameters allows for fine tuning for improved transparency and model performance (usual overfitting caveat applies, as does the trade-off described in **?@sec-car-pipelines-trade**).

The survey of ANNs (Chapter 15) demonstrated how reduction is currently utilised without transparency. Many of these ANNs are implicitly reductions to probabilistic classification (Section 16.5.1.6) however none include details about how the reduction is performed. Furthermore in implementation, none provide interface points to the reduction hyper-parameters. Formalisation encourages consistent terminology, methodology and transparent implementation, which can only improve model performance by exposing further hyper-parameters.

Accessibility is improved by formalising specific reduction workflows that previously demanded expert knowledge in deriving, building, and running these pipelines.

Finally there is an economic and efficiency advantage to reduction. A reduction model is relatively 'cheap' to explore as they utilise pre-established models and components to solve a new problem. Therefore if a certain degree of predictive ability can be demonstrated from reduction models, it may not be worth the expense of pursuing more novel ideas and hence reduction can help direct future research.

16.3.2 Task, Loss, and Data Reduction

Reduction can be categorised into task, loss, and data reduction, often these must be used in conjunction with each other. The direction of the reductions may be one- or two-way; this is visualised in **?@fig-car-reduxdiag**. This diagram should not be viewed as a strict fit/predict/evaluation workflow but instead as a guidance for which tasks, T, data, D, models, M, and losses, L, are required for each other. The subscript O refers to the original object 'level' before reduction, whereas the subscript R is in reference to the reduced object.

The individual task, model, and data compositions in the diagram are listed below, the reduction from survival to classification (Section 16.5.1) is utilised as a running example to help exposition.

- $T_O \to T_R$: By definition of a machine learning reduction, task reduction will always be one way. A more complex task, T_O, is reduced to a simpler one, T_R, for solving. T_R could also be multiple simpler tasks. For example, solving a survival task, T_O, by classification, T_R (Section 16.5.1).
- $T_R \to M_R$: All machine learning tasks have models that are designed to solve them. For example logistic regression, M_R, for classification tasks, T_R.
- $M_R \to M_O$: The simpler models, M_R, are used for the express purpose to solve the original task, T_O, via solving the simpler ones. To solve T_O, a compositor must be applied, which may transform one (SOC) or multiple models (MOC) at a model- or prediction-level, thus creating M_O. For example predicting survival probabilities with logistic regression, M_R, at times $1, ..., \tau^*$ for some $\tau^* \in \mathbb{N}_{>0}$ (Section 16.5.1.4).
- $M_O \to T_O$: The original task should be solvable by the composite model. For example predicting a discrete survival distribution by concatenating probabilistic predictions at the times $1, ..., \tau^*$ (Section 16.5.1.6).
- $D_O \to D_R$: Just as the tasks and models are reduced, the data required to fit these must

likewise be reduced. Similarly to task reduction, data reduction can usually only take place in one direction, to see why this is the case take an example of data reduction by summaries. If presented with 10 data-points $\{1, 1, 1, 5, 7, 3, 5, 4, 3, 3\}$ then these could be reduced to a single point by calculating the sample mean, 3.3. Clearly given only the number 3.3 there is no strategy to recover the original data. There are very few (if any) data reduction strategies that allow recovery of the original data. Continuing the running example, survival data, D_O, can be binned (Section 16.5.1.1) to classification data, D_R.

There is no arrow between D_O and M_O as the composite model is never fit directly, only via composition from $M_R \to M_O$. However, the original data, D_O, is required when evaluating the composite model against the respective loss, L_O.[1] Reduction should be directly comparable to non-reduction models, hence this diagram does not include loss reduction and instead insists that all models are compared against the same loss L_O.

A reduction is said to be *complete* if there is a full pipeline from $T_O \to M_O$ and the original task is solved, otherwise it is *incomplete*. The simplest complete reduction is comprised of the pipeline $T_O \to T_R \to M_R \to M_O$. Usually this is not sufficient on its own as the reduced models are fit on the reduced data, $D_R \to M_R$.

A complete reduction can be specified by detailing:

i. the original task and the sub-task(s) to be solved, $T_O \to T_R$;
ii. the original dataset and the transformation to the reduced one, $D_O \to D_R$ (if required); and
iii. the composition from the simpler model to the complex one, $M_R \to M_O$.

16.3.3 Common Mistakes in Implementation of Reduction

In surveying models and measures, several common mistakes in the implementation of reduction and composition were found to be particularly prevalent and problematic throughout the literature. It is assumed that these are indeed mistakes (not deliberate) and result from a lack of prior formalisation. These mistakes were even identified 20 years ago (Schwarzer, Vach, and Schumacher 2000) but are provided in more detail in order to highlight their current prevalence and why they cannot be ignored.

RM1. Incomplete reduction. This occurs when a reduction workflow is presented as if it solves the original task but fails to do so and only the reduction strategy is solved. A common example is claiming to solve the survival task by using binary classification, e.g. erroneously claiming that a model predicts survival probabilities (which implies distribution) when it actually predicts a five year probability of death ((**box-task-classif?**)). This is a mistake as it misleads readers into believing that the model solves a survival task ((**box-task-surv?**)) when it does not. This is usually a semantic not mathematical error and results from misuse of terminology. It is important to be clear about model predict types (Chapter 6) and general terms such as 'survival predictions' should be avoided unless they refer to one of the three prediction tasks. RM2. Inappropriate comparisons. This is a direct consequence of (RM1) and the two are often seen together. (RM2) occurs when an incomplete reduction is directly compared to a survival model (or complete reduction model) using a measure appropriate for the reduction. This may lead to a reduction model appearing erroneously superior. For example, comparing a logistic regression to a random survival forest (RSF)

[1] A complete diagram would indicate that D_O is split into training data, which is subsequently reduced, and test data, which is passed to L_O. All reductions in this section can be applied to any data splitting process.

(Chapter 12) for predicting survival probabilities at a single time using the accuracy measure is an unfair comparison as the RSF is optimised for distribution predictions. This would be non-problematic if a suitable composition is clearly utilised. For example a regression SSVM predicting survival time cannot be directly compared to a Cox PH. However the SSVM can be compared to a CPH composed with the probabilistic to deterministic compositor, then conclusions can be drawn about comparison to the composite survival time Cox model (and not simply a Cox PH). RM3. Na"ive censoring deletion. This common mistake occurs when trying to reduce survival to regression or classification by simply deleting all censored observations, even if censoring is informative. This is a mistake as it creates bias in the dataset, which can be substantial if the proportion of censoring is high and informative. More robust deletion methods are described in Chapter 20. RM4. Oversampling uncensored observations. This is often seen when trying to reduce survival to regression or classification, and often alongside (RM3). Oversampling is the process of replicating observations to artificially inflate the sample size of the data. Whilst this process does not create any new information, it can help a model detect important features in the data. However, by only oversampling uncensored observations, this creates a source of bias in the data and ignores the potentially informative information provided by the proportion of censoring.

16.4 Composition Strategies for Survival Analysis

Though composition is common practice in survival analysis, with the Cox model being a prominent example, a lack of formalisation means a lack of consensus in simple operations. For example, it is often asked in survival analysis how a model predicting a survival distribution can be used to return a survival time prediction. A common strategy is to define the survival time prediction as the median of the predicted survival curve however there is no clear reason why this should be more sensible than returning the distribution mean, mode, or some random quantile. Formalisation allow these choices to be analytically compared both theoretically and practically as hyper-parameters in a workflow. Four prediction compositions are discussed in this section ((**tab-car-taxredcar?**)), three are utilised to convert prediction types between one another, the fourth is for aggregating multiple predictions. One data composition is discussed for converting survival to regression data. Each is first graphically represented and then the components are discussed in detail. As with losses in the previous chapter, compositions are discussed at an individual observation level but extend trivially to multiple observations.

Table 16.1: Compositions formalised in Section 16.4.

ID[1]	Composition	Type[2]	Level[3]
C1)	Linear predictor to distribution	MOC	Prediction
C2)	Survival time to distribution	MOC	Prediction
C3)	Distribution to survival time	SOC	Prediction
C4)	Survival model averaging	MOC	Prediction
C5)	Survival to regression	SOC	Data

1. ID for reference throughout this book. 2. Composition type. Multi-object composition (MOC) or single-object composition (SOC). 3. Composition level.

16.4.1 C1) Linear Predictor → Distribution

This is a prediction-level MOC that composes a survival distribution from a predicted linear predictor and estimated baseline survival distribution. The composition (**?@fig-car-comp-distr**) requires:

- $\hat{\eta}$: Predicted linear predictor. $\hat{\eta}$ can be tuned by including this composition multiple times in a benchmark experiment with different models predicting $\hat{\eta}$. In theory any continuous ranking could be utilised instead of a linear predictor though results may be less sensible (**?@sec-car-pipelines-trade**).
- $\hat{S}_0$: Estimated baseline survival function. This is usually estimated by the Kaplan-Meier estimator fit on training data, $\hat{S}_{KM}$. However any model that can predict a survival distribution can estimate the baseline distribution (caveat: see **?@sec-car-pipelines-trade**) by taking a uniform mixture of the predicted individual distributions: say $\xi_1, ..., \xi_m$ are m predicted distributions, then $\hat{S}_0(\tau) = \frac{1}{m}\sum_{i=1}^{m} \xi_i.S(\tau)$. The mixture is required as the baseline must be the same for all observations. Alternatively, parametric distributions can be assumed for the baseline, e.g. $\xi = \text{Exp}(2)$ and $\xi.S(t) = \exp(-2t)$. As with $\hat{\eta}$, this parameter is also tunable.
- M: Chosen model form, which theoretically can be any non-increasing right-continuous function but is usually one of:
- Proportional Hazards (PH): $S_{PH}(\tau|\eta, S_0) = S_0(\tau)^{\exp(\eta)}$
- Accelerated Failure Time (AFT): $S_{AFT}(\tau|\eta, S_0) = S_0(\frac{\tau}{\exp(\eta)})$
- Proportional Odds (PO): $S_{PO}(\tau|\eta, S_0) = \frac{S_0(\tau)}{\exp(-\eta)+(1-\exp(-\eta))S_0(\tau)}$

Models that predict linear predictors will make assumptions about the model form and therefore dictate sensible choices of M, for example the Cox model assumes a PH form. This does not mean other choices of M cannot be specified but that interpretation may be more difficult (**?@sec-car-pipelines-trade**). The model form can be treated as a hyper-parameter to tune. * C: Compositor returning the composed distribution, $\zeta := C(M, \hat{\eta}, \hat{S}_0)$ where ζ has survival function $\zeta.S(\tau) = M(\tau|\hat{\eta}, \hat{S}_0)$.

Pseudo-code for training ((**alg-car-comp-distr-fit?**)) and predicting ((**alg-car-comp-distr-pred?**)) this composition as a model 'wrapper' with sensible parameter choices (**?@sec-car-pipelines-trade**) is provided in appendix (**app-car?**).

16.4.2 C2) Survival Time → Distribution

This is a prediction-level MOC that composes a distribution from a predicted survival time and assumed location-scale distribution. The composition (**?@fig-car-comp-response**) requires:

- $\hat{T}$: A predicted survival time. As with the previous composition, this is tunable. In theory any continuous ranking could replace $\hat{T}$, though the resulting distribution may not be sensible (**?@sec-car-pipelines-trade**).
- ξ: A specified location-scale distribution, $\xi(\mu, \sigma)$, e.g. Normal distribution.
- $\hat{\sigma}$: Estimated scale parameter for the distribution. This can be treated as a hyper-parameter or predicted by another model.
- C: Compositor returning the composed distribution $\zeta := C(\xi, \hat{T}, \hat{\sigma}) = \xi(\hat{T}, \hat{\sigma})$.

Pseudo-code for training ((**alg-car-comp-response-fit?**)) and predicting ((**alg-car-comp-response-pred?**)) this composition as a model 'wrapper' with sensible parameter choices

(**?@sec-car-pipelines-trade**) is provided in appendix (**app-car?**).

16.4.3 C3) Distribution → Survival Time Composition

This is a prediction-level SOC that composes a survival time from a predicted distribution. Any paper that evaluates a distribution on concordance is implicitly using this composition in some manner. Not acknowledging the composition leads to unfair model comparison (Section 16.3.3). The composition (**?@fig-car-comp-crank**) requires:

- ζ: A predicted survival distribution, which again is 'tunable'.
- ϕ: A distribution summary method. Common examples include the mean, median and mode. Other alternatives include distribution quantiles, $\zeta.F^{-1}(\alpha),\backslash\alpha \in [0,1]$; α could be tuned as a hyper-parameter.
- C: Compositor returning composed survival time predictions, $\hat{T} := C(\phi, \zeta) = \phi(\zeta)$.

Pseudo-code for training ((**alg-car-comp-crank-fit?**)) and predicting ((**alg-car-comp-crank-pred?**)) this composition as a model 'wrapper' with sensible parameter choices (**?@sec-car-pipelines-trade**) is provided in appendix (**app-car?**).

16.4.4 C4) Survival Model Averaging

Ensembling is likely the most common composition in machine learning. In survival it is complicated slightly as multiple prediction types means one of two possible compositions is utilised to average predictions. The (**?@fig-car-comp-avg**) composition requires:

- $\rho = \rho_1, ..., \rho_B$: B predictions (not necessarily from the same model) of the same type: ranking, survival time or distribution; again 'tunable'.
- $w = w_1, ..., w_B$: Weights that sum to one.
- C: Compositor returning combined predictions, $\hat{\rho} := C(\rho, w)$ where $C(\rho, w) = \frac{1}{B}\sum_{i=1}^{B} w_i\rho_i$, if ρ are ranking of survival time predictions; or $C(\rho, w) = \zeta$ where ζ is the distribution defined by the survival function $\zeta.S(\tau) = \frac{1}{B}\sum_{i=1}^{B} w_i\rho_i.S(\tau)$, if ρ are distribution predictions.

Pseudo-code for training ((**alg-car-comp-avg-fit?**)) and predicting ((**alg-car-comp-avg-pred?**)) this composition as a model 'wrapper' with sensible parameter choices (**?@sec-car-pipelines-trade**) is provided in appendix (**app-car?**).

16.5 Novel Survival Reductions

This section collects the various strategies and settings discussed previously into complete reduction workflows. (**tab-car-reduxes?**) lists the reductions discussed in this section with IDs for future reference. All strategies are described by visualising a graphical pipeline and then listing the composition steps required in fitting and predicting.

This section only includes novel reduction strategies and does not provide a survey of pre-existing strategies. This limitation is primarily due to time (and page) constraints as every method has very distinct workflows that require complex exposition. Well-established strategies are briefly mentioned below and future research is planned to survey and compare all strategies with respect to empirical performance (i.e. in benchmark experiments).

Two prominent reductions are 'landmarking' (Van Houwelingen 2007) and piecewise expo-

nential models (M. Friedman 1982). Both are reductions for time-varying covariates and hence outside the scope of this book. Relevant to this book scope is a large class of strategies that utilise 'discrete time survival analysis' (Tutz and Schmid 2016); these strategies include reductions (R7) and (R8). Methodology for discrete time survival analysis has been seen in the literature for the past three decades (Liestol, Andersen, and Andersen 1994). The primary reduction strategy for discrete time survival analysis is implemented in the R package **discSurv**. (Welchowski and Schmid 2019); this is very similar to (R7) except that it enforces stricter constraints in the composition procedures and forces a 'discrete-hazard' instead of 'discrete-survival' representation (Section 16.5.1.2).

16.5.1 R7-R8) Survival → Probabilistic Classification

Two separate reductions are presented in **?@fig-car-R7R8** however as both are reductions to probabilistic classification and are only different in the very last step, both are presented in this section. Steps and compositions of the reduction (**?@fig-car-R7R8**):

Fit F1) A survival dataset, $\mathcal{D}_{train}$, is binned, B, with a continuous to discrete data composition (Section 16.5.1.1). F2) A multi-label classification model, with adaptations for censoring, $g_L(D_B|\theta)$, is fit on the transformed dataset, D_B. Optionally, g_L could be further reduced to binary, g_B, or multi-class classification, g_c, (Section 16.5.1.4). **Predict** P1) Testing survival data, $\mathcal{D}_{test}$, is passed to the trained classification model, $\hat{g}$, to predict pseudo-survival probabilities $\tilde{S}$ (or optionally hazards (Section 16.5.1.2)). P2a) Predictions can be composed, T_1, into a survival distribution prediction, $\zeta = \zeta_1, ..., \zeta_m$ (Section 16.5.1.6); or, P2b) Predictions can be composed, T_2, to survival time predictions, $\hat{T} = \hat{T}_1, ..., \hat{T}_m$ (Section 16.5.1.7).

Further details for binning, multi-label classification, and transformation of pseudo-survival probabilities are now provided.

16.5.1.1 Composition: Binning Survival Times

An essential part of the reduction is the transformation from a survival dataset to a classification dataset, which requires two separate compositions. The first (discussed here) is to discretise the survival times ($B(\mathcal{D}_{train}|w)$ in **?@fig-car-R7R8**) and the second is to merge the survival time and censoring indicator into a single outcome (Section 16.5.1.2).

Discretising survival times is achieved by the common 'binning' composition, in which a continuous outcome is discretised into 'bins' according to specified thresholds. These thresholds are usually determined by specifying the width of the bins as a hyper-parameter w.[2] This is a common transformation and therefore further discussion is not provided here. An example is given below with the original survival data on the left and the binned data on the right ($w = 1$).

X	Time (Cont.)	Died
1	1.56	0
2	2	1
3	3.3	1
4	3.6	0
5	4	0

[2]Binning is described here with equal widths but generalises to unequal widths trivially.

X	Time (Disc.)	Died
1	[1, 2)	0
2	[2, 3)	1
3	[3, 4)	1
4	[3, 4)	0
5	[4, 5)	0

16.5.1.2 Composition: Survival to Classification Outcome

The binned dataset still has the unique survival data format of utilising two outcomes for training (time and status) but only making a prediction for one outcome (distribution). In order for this to be compatible with classification, the two outcome variables are composed into a single variable.[3] This is achieved by casting the survival times into a 'wide' format and creating a new outcome indicator.[4] Two outcome transformations are possible, the first represents a discrete survival function and the second represents a discrete hazard function.[5]

Discrete Survival Function Composition

In this composition, the data in the transformed dataset represents the discrete survival function. The new indicator is defined as follows,

$$Y_{i;\tau} := \begin{cases} 1, & T_i > \tau \\ 0, & T_i \leq \tau \cap \Delta_i = 1 \\ -1, & T_i \leq \tau \cap \Delta_i = 0 \end{cases}$$

At a given discrete time τ, an observation, i, is either alive ($Y_{i;\tau} = 1$), dead ($Y_{i;\tau} = 0$), or censored ($Y_{i;\tau} = -1$). Therefore $\hat{P}(Y_{i;\tau} = 1) = \hat{S}_i(\tau)$, motivating this particular choice of representation.

This composition is demonstrated below with the binned data (left) and the composed classification data (right).

X	Time (Disc.)	Died
1	[1, 2)	0
2	[2, 3)	1
3	[3, 4)	1
4	[3, 4)	0
5	[4, 5)	0

X	[1,2)	[2,3)	[3,4)	[4,5)
1	-1	-1	-1	-1
2	1	0	0	0

[3]This is the first key divergence from other discrete-time classification strategies, which use the censoring indicator as the outcome and the time outcome as a feature.

[4]This is the second key divergence from other discrete-time classification strategies, which keep the data in a 'long' format.

[5]This is the final key divergence from other discrete-time classification strategies, which enforce the discrete hazard representation.

X	[1,2)	[2,3)	[3,4)	[4,5)
3	1	1	0	0
4	1	1	-1	-1
5	1	1	-1	-1

Discrete Hazard Function Composition

In this composition, the data in the transformed dataset represents the discrete hazard function. The new indicator is defined as follows,

$$Y^*_{i;\tau} := \begin{cases} 1, & T_i = \tau \cap \Delta_i = 1 \\ -1, & T_i = \tau \cap \Delta_i = 0 \\ 0, & \text{otherwise} \end{cases}$$

At a given discrete time τ, an observation, i, either experiences the event ($Y^*_{i;\tau} = 1$), experiences censoring ($Y_{i;\tau} = -1$), or neither ($Y_{i;\tau} = 0$). Utilising sequential multi-label classification problem transformation methods (Section 16.5.1.4) results in $\hat{P}(Y^*_{i;\tau} = 1) = \hat{h}_i(\tau)$. If methods are utilised that do not 'look back' at predictions then $\hat{P}(Y^*_{i;\tau} = 1) = \hat{p}_i(\tau)$ (Section 16.5.1.4).[6]

This composition is demonstrated below with the binned data (left) and the composed classification data (right).

X	Time (Disc.)	Died
1	[1, 2)	0
2	[2, 3)	1
3	[3, 4)	1
4	[3, 4)	0
5	[4, 5)	0

X	[1,2)	[2,3)	[3,4)	[4,5)
1	-1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	-1	0
5	0	0	0	-1

Multi-Label Classification Data

In both compositions, survival data t.v.i. $\mathbb{R}^p \times \mathbb{R}_{\geq 0} \times \{0, 1\}$ is transformed to multi-label classification data t.v.i. $\mathbb{R}^p \times \{-1, 0, 1\}^K$ for K binned time-intervals. The multi-label classification task is defined in Section 16.5.1.4 with possible algorithms.

The discrete survival representation has a slightly more natural interpretation and is 'easier' for classifiers to use for training as there are more positive events (i.e. more observations

[6]This important distinction is not required in other discrete-time reduction strategies that automatically condition the prediction by including time as a feature.

alive) to train on, whereas the discrete hazard representation will have relatively few events in each time-point. However the hazard representation leads to more natural predictions (Section 16.5.1.6).

A particular bias that may easily result from the composition of survival to classification data is now discussed.

16.5.1.3 Reduction to Classification Bias

The reduction to classification bias is commonly known (Zhou et al. 2005) but is reiterated briefly here as it must be accounted for in any automated reduction to classification workflow. This bias occurs when making classification predictions about survival at a given time and incorrectly censoring patients who have not been observed long enough, instead of removing them.

By example, say the prediction of interest is five-year survival probabilities after a particular diagnosis, clearly a patient who has only been diagnosed for three years cannot inform this prediction. The bias is introduced if this patient is censored at five-years instead of being removed from the dataset. The result of this bias is to artificially inflate the probability of survival at each time-point as an unknown outcome is treated as censored and therefore alive.

This bias is simply dealt with by removing patients who have not been alive 'long enough'.[7] Paradoxically, even if a patient is observed to die before the time-point of interest, they should still be removed if they have not been in the dataset 'long enough' as failing to do so will result in a bias in the opposite direction, thus over-inflating the proportion of dead observations.

Accounting for this bias is particularly important in the multi-label reduction as the number of observable patients will decrease over time due to censoring.

16.5.1.4 Multi-Label Classification Algorithms

As the work in this section is completely out of the book scope, the full text is in appendix (**app-mlc?**). The most important contributions from this section are:

- Reviewing problem transformation methods (Tsoumakas and Katakis 2007) for multi-label classification;
- Identifying that only binary relevance, nested stacking, and classifier chains are appropriate in this reduction; and
- Generalising these methods into a single wrapper for any binary classifier, the 'LWrapper'.

16.5.1.5 Censoring in Classification

Classification algorithms cannot natively handle the censoring that is included in the survival reduction, but this can be incorporated using one of two approaches.

Multi-Class Classification

All multi-label datasets can also handle multi-class data, hence the simplest way in which to handle censoring is to make multi-class predictions in each label for the outcome Y_τ $t.v.i.\{-1, 0, 1\}$. Many off-shelf classification learners can make multi-class predictions natively and simple reductions exist for those that cannot. As a disadvantage to this method,

[7] Accounting for this bias is only possible if the study start and end dates are known, as well as the date the patient entered the study.

classifiers would then predict if an individual is dead or alive or censored (each mutually exclusive), and not simply alive or dead. Though this could be perceived as an advantage when censoring is informative as this will accurately reflect a real-world competing-risks set-up.

Subsetting/Hurdle Models

For this approach, the multi-class task is reduced to two binary class tasks: first predict if a subject is censored or not (dead or alive) and only if the prediction for censoring is below some threshold, $\alpha \in [0, 1]$, then predict if the subject is alive or not (dead or censored). If the probability of censoring is high in the first task then the probability of being alive is automatically set to zero in the final prediction, otherwise the prediction from the second task is used. Any classifier can utilise this approach and it has a meaningful interpretation, additionally α is a tunable hyper-parameter. The main disadvantage is increases to storage and run-time requirements as double the number of models may be fit.

Once the datasets have been composed to classification datasets and censoring is suitably incorporated by either approach, then any probabilistic classification model can be fit on the data. Predictions from these models can either be composed to a distribution prediction (R7) or a survival time prediction (R8).

16.5.1.6 R7) Probabilistic Survival → Probabilistic Classification

This final part of the (R7) reduction is described separately for discrete hazard and survival representations of the data (Section 16.5.1.2).

Discrete Hazard Representation

In this representation recall that predictions of the positive class, $P(Y_\tau = 1)$, are estimating the quantity $h(\tau)$. These predictions provide a natural and efficient transformation from predicted hazards to survival probabilities. Let $\hat{h}_i$ be a predicted hazard function for some observation i, then the survival function for that observation can be found with a Kaplan-Meier type estimator,

$$\tilde{S}_i(\tau^*) = \prod_\tau 1 - \hat{h}_i(\tau)$$

Now predictions are for a pseudo-survival function, which is 'pseudo' as it is not right-continuous. Resolving this is discussed below.

Discrete Survival Representation

In this representation, $P(Y_\tau = 1)$ is estimating $S(\tau)$, which means that predictions from a classification model result in discrete point predictions and not a right-continuous function. More importantly, there is no guarantee that a non-increasing function will be predicted, i.e. there is no guarantee that $P(Y_j = 1) < P(Y_i = 1)$, for time-points $j > i$.

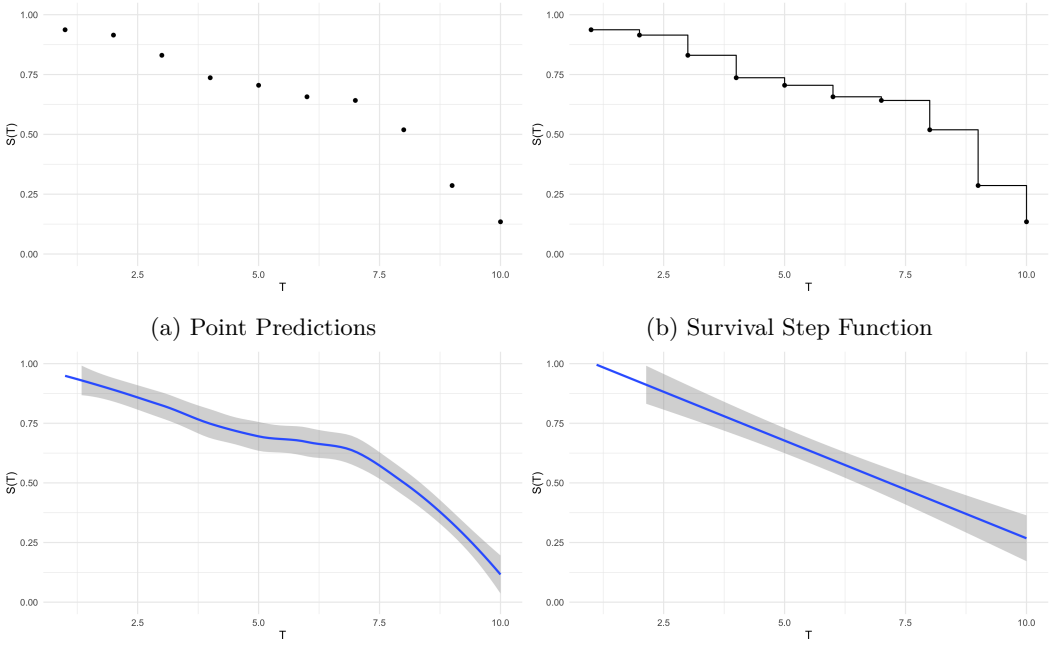
Unfortunately there is no optimal way of dealing with predictions of this sort and 'mistakes' of this kind have been observed in some software implementation. One point to note is that in practice these are quite rare as the probability of survival will always decrease over time. Therefore the 'usual' approach is quite 'hacky' and involves imputing increasing predictions with the previous prediction, formally,

$$\tilde{S}(i+1) := \min\{P(Y_{i+1} = 1), P(Y_i = 1)\}, \forall i = \mathbb{R}_{\geq 0}$$

assuming $\tilde{S}(0) = 1$. Future research should seek more robust alternatives.

Right-Continuous Survival Function

From either representation, a \ non-increasing but non-continuous pseudo-survival function, $\tilde{S}$, is now predicted. Creating a right-continuous function ('$T_1(\tilde{S})$' in **?@fig-car-R7**) from these point predictions (Figure 16.3 (a)) is relatively simple and well-known with accessible off-shelf software. At the very least, one can assume a constant hazard rate between predictions and cast them into a step function (Figure 16.3 (b)). This is a fairly common assumption and is usually valid as bin-width decreases. Alternatively, the point predictions can be smoothed into a continuous function with off-shelf software, for example with polynomial local regression smoothing (Figure 16.3 (c)) or generalised linear smoothing (Figure 16.3 (d)). Whichever method is chosen, the survival function is now non-increasing right-continuous and the (R7) reduction is complete.



(a) Point Predictions

(b) Survival Step Function

(c) Local polynomial regression smoothing

(d) Generalised linear smoothing

Figure 16.3: Survival function as a: point prediction (a), step function assuming constant risk (b), local polynomial regression smoothing (c), and generalised linear smoothing (d). (c) and (d) computed with **ggplot2** (Wickham 2016).

16.5.1.7 R8) Deterministic Survival → Probabilistic Classification

Predicting a deterministic survival time from the multi-label classification predictions is relatively straightforward and can be viewed as a discrete analogue to (C3) (Section 16.4.3). For the discrete hazard representation, one can simply take the predicted time-point for an individual to be time at which the predicted hazard probability is highest however this could easily be problematic as there may be multiple time-points at which the predicted hazard equals 1. Instead it is cleaner to first cast the hazard to a pseudo-survival probability (Section 16.5.1.6) and then treat both representations the same.

Let $\tilde{S}_i$ be the predicted multi-label survival probabilities for an observation i such that $\tilde{S}_i(\tau)$ corresponds with $\hat{P}(Y_{i;\tau} = 1)$ for label $\tau \in \mathcal{K}$ where $Y_{i;\tau}$ is defined in Section 16.5.1.2 and

$\mathcal{K} = \{1, ..., K\}$ is the set of labels for which to make predictions. Then the survival time transformation is defined by

$$T_2(\tilde{S}_i) = \inf\{\tau \in \mathcal{K} : \tilde{S}_i(\tau) \leq \beta\}$$

for some $\beta \in [0, 1]$.

This is interpreted as defining the predicted survival time as the first time-point in which the predicted probability of being alive drops below a certain threshold β. Usually $\beta = 0.5$, though this can be treated as a hyper-parameter for tuning. This composition can be utilised even if predictions are not non-increasing, as only the first time the predicted survival probability drops below the threshold is considered. With this composition the (R8) reduction is now complete.

16.6 Conclusions

This chapter introduced composition and reduction to survival analysis and formalised specific strategies. Formalising these concepts allows for better quality of research and most importantly improved transparency. Clear interface points for hyper-parameters and compositions allow for reproducibility that was previously obfuscated by unclear workflows and imprecise documentation for pipelines.

Additionally, composition and reduction improves accessibility. Reduction workflows vastly increase the number of machine learning models that can be utilised in survival analysis, thus opening the field to those whose experience is limited to regression or classification. Formalisation of workflows allows for precise implementation of model-agnostic pipelines as computational objects, as opposed to functions that are built directly into an algorithm without external interface points.

Finally, predictive performance is also increased by these methods, which is most prominently the case for the survival model averaging compositor (as demonstrated by RSFs).

17

Competing Risks Pipelines

TODO (150-200 WORDS)

> ! Page coming soon!
>
> **We are working on this page and it will be available soon!**

18

Discrete Time Survival Analysis

TODO (150-200 WORDS)

> ! Page coming soon!
>
> **We are working on this page and it will be available soon!**

19

Connections to Poisson Regression and Processes

TODO (150-200 WORDS)

! Page coming soon!

We are working on this page and it will be available soon!

20

Connections to Regression and Imputation

TODO (150-200 WORDS)

> ! Major changes expected!
>
> **This page is a work in progress and major changes will be made over time.**

TODO

- I think all these sections should have examples in implemented models, e.g., here we can point to SVM models and some neural nets
- We can also point to neural nets that use reduction to essentially just predict the linear predictor via regression or to use pseudovalues
- Add pseudovalues
- Add prediction of the observed outcome (not survival) time

This is a data-level SOC that transforms survival data to regression data by either removing censored observations or 'imputing' survival times. This composition is frequently incorrectly utilised (Section 16.3.3) and therefore more detail is provided here than previous compositions. Note that the previous compositions were prediction-level transformations that occur after a survival model makes a prediction, whereas this composition is on a data-level and can take place before model training or predicting.

In Statistics, there are only two methods for removing 'missing' values: deletion and imputation; both of these have been attempted for censoring.

Censoring can be beneficial, harmful, or neutral; each will affect the data differently if deleted or imputed. Harmful censoring occurs if the reason for censoring is negative, for example drop-out due to disease progression. Harmful censoring indicates that the true survival time is likely soon after the censoring time. Beneficial censoring occurs if censoring is positive, for example drop-out due to recovery. This indicates that the true survival time is likely far from the censoring time. Finally neutral censoring occurs when no information can be gained about the true survival time from the censoring time. Whilst the first two of these can be considered to be dependent on the outcome, neutral censoring is often the case when censoring is independent of the outcome conditional on the data, which is a standard assumption for the majority of survival models and measures.

20.0.0.1 Deletion #{sec-redux-regr-del}

Deletion is the process of removing observations from a dataset. This is usually seen in 'complete case analysis' in which observations with 'missingness', covariates with missing values, are removed from the dataset. In survival analysis this method is somewhat riskier as the subjects to delete depend on the outcome and not the features. Three methods are considered, the first two are a more brute-force approach whereas the third allows for some flexibility and tuning.

Complete Deletion

Deleting all censored observations is simple to implement with no computational overhead. Complete deletion results in a smaller regression dataset, which may be significantly smaller if the proportion of censoring is high. If censoring is uninformative, the dataset is suitably large and the proportion of censoring suitably low, then this method can be applied without further consideration. However if censoring is informative then deletion will add bias to the dataset, although the 'direction' of bias cannot be known in advance. If censoring is harmful then censored observations will likely have a similar profile to those that died, thus removing censoring will artificially inflate the proportion of those who survive. Conversely if censoring is beneficial then censored observations may be more similar to those who survive, thus removal will artificially inflate the proportion of those who die.

Omission

Omission is the process of omitting the censoring indicator from the dataset, thus resulting in a regression dataset that assumes all observations experienced the event. Complete deletion results in a smaller dataset of dead patients, omission results in no sample size reduction but the outcome may be incorrect. This reduction strategy is likely only justified for harmful censoring. In this case the true survival time is likely close to the censoring time and therefore treating censored observations as dead may be a fair assumption.

IPCW

If censoring is conditionally-outcome independent then deletion of censored events is possible by using Inverse Probability of Censoring Weights (IPCW). This method has been seen several times throughout this book in the context of models and measures. It has been formalised as a composition technique by Vock *et al.* (2016) (Vock et al. 2016) although their method is limited to binary classification. Their method weights the survival time of uncensored observations by $w_i = 1/\hat{G}_{KM}(T_i)$ and deletes censored observations, where $\hat{G}_{KM}$ is the Kaplan-Meier estimate of the censoring distribution fit on training data. As previously discussed, one could instead consider the Akritas (or any other) estimator for $\hat{G}_{KM}$.

Whilst this method does provide a 'safer' way to delete censored observations, there is not a necessity to do so. Instead consider the following weights

$$w_i = \frac{\Delta_i + \alpha(1 - \Delta_i)}{\hat{G}_{KM}(T_i)} \tag{20.1}$$

where $\alpha \in [0, 1]$ is a hyper-parameter to tune. Setting $\alpha = 1$ equally weights censored and uncensored observations and setting $\alpha = 0$ recovers the setting in which censored observations are deleted. It is assumed $\hat{G}_{KM}$ is set to some very small ϵ when $\hat{G}_{KM}(T_i) = 0$. When $\alpha \neq 0$ this becomes an imputation method, other imputation methods are now discussed.

20.0.0.2 Imputation

Imputation methods estimate the values of missing data conditional on non-missing data and other covariates. Whilst the true value of the missing data can never be known, by carefully conditioning on the 'correct' covariates, good estimates for the missing value can be obtained to help prevent a loss of data. Imputing outcome data is more difficult than imputing covariate data as models are then trained on 'fake' data. However a poor imputation should still be clear when evaluating a model as testing data remains un-imputed. By imputing censoring times with estimated survival times, the censoring indicator can be removed and the dataset becomes a regression dataset.

Gamma Imputation

Gamma imputation (Jackson et al. 2014) incorporates information about whether censoring is harmful, beneficial, or neutral. The method imputes survival times by generating times from a shifted proportional hazards model

$$h(\tau) = h_0(\tau) \exp(\eta + \gamma)$$

where η is the usual linear predictor and $\gamma \in \mathbb{R}$ is a hyper-parameter determining the 'type' of censoring such that $\gamma > 0$ indicates harmful censoring, $\gamma < 0$ indicates beneficial censoring, and $\gamma = 0$ is neutral censoring. This imputation method has the benefit of being tunable as γ is a hyper-parameter and there is a choice of variables to condition the imputation. No independent experiments exist studying how well this method performs, nor discussing the theoretical properties of the method.

MRL

The Mean Residual Lifetime (MRL) estimator has been previously discussed in the context of SVMs (Section 13.2). Here the estimator is extended to serve as an imputation method. Recall the MRL function, $MRL(\tau|\hat{S}) = \int_\tau^\infty \hat{S}(u)\ du / \hat{S}(\tau)$, where $\hat{S}$ is an estimate of the survival function of the underlying survival distribution (e.g. $\hat{S}_{KM}$). The MRL is interpreted as the expected remaining survival time after the time-point τ. This serves as a natural imputation strategy where given the survival outcome (T_i, Δ_i), the new imputed time T_i' is given by

$$T_i' = T_i + (1 - \Delta_i) MRL(T_i|\hat{S})$$

where $\hat{S}$ would be fit on the training data and could be an unconditional estimator, such as Kaplan-Meier, or conditional, such as Akritas. The resulting survival times are interpreted as the true times for those who died and the expected survival times for those who were censored.

Buckley-James

Buckley-James (Buckley and James 1979) is another imputation method discussed earlier (**?@sec-surv-ml-models-boost**). The Buckley-James method uses an iterative procedure to impute censored survival times by the conditional expectation given censoring times and covariates (Z. Wang and Wang 2010). Given the survival tuple for an outcome (T_i, Δ_i), the new imputed time T_i' is

$$T_i' = \begin{cases} T_i, & \Delta_i = 1 \\ X_i\hat{\beta} + \frac{1}{\hat{S}_{KM}(e_i)} \sum_{e_i < e_k} \hat{p}_{KM}(e_k) e_k & \Delta_i = 0 \end{cases}$$

where $\hat{S}_{KM}$ is the Kaplan-Meier estimator of the survival distribution estimated on training data and with associated pmf $\hat{p}_{KM}$ and $e_i = T_i - X_i\hat{\beta}$ where $\hat{\beta}$ are estimated coefficients of a linear regression model fit on (X_i, T_i). Given the least squares approach, more parametric assumptions are made than other imputation methods and it is more complex to separate model fitting from imputation. Hence, this imputation may only be appropriate on a limited number of data types.

Alternative Methods

Other methods have been proposed for 'imputing' censored survival times though with either less clear discussion or to no benefit. Multiple imputation by chained equations (MICE) has been demonstrated to perform well with covariate data and even outcome data (in a non-survival setting). However no adaptations have been developed to incorporate censoring times into the imputation and therefore is less informative than Gamma imputation.

Re-calibration of censored survival times (Vinzamuri, Li, and Reddy 2017) uses an iterative update procedure to 're-calibrate' censoring times however the motivation behind the method is not sufficiently clear to be of interest in general survival modelling tasks outside of the authors' specific pipelines.

Finally parametric imputation is defined by making random draws from truncated probability distributions and adding these to the censoring time (P. Royston 2001; Patrick Royston, Parmar, and Altman 2008). Whilst this method is arguably the simplest method and will lead to a sufficiently random sample, i.e. not one skewed by the imputation process, in practice the randomness leads to unrealistic results, with some imputed times being very far from the original censoring times and some being very close.

20.0.0.3 The Decision to Impute or Delete

Deletion methods are simple to implement and fast to compute however they can lead to biasing the data or a significant sample reduction if used incorrectly. Imputation methods can incorporate tuning and have more relaxed assumptions about the censoring mechanism, though they may lead to over-confidence in the resulting outcome and therefore add bias into the dataset. In some cases, the decision to impute or delete is straightforward, for example if censoring is uninformative and only few observations are censored then complete deletion is appropriate. If it is unknown if censoring is informative then this can crudely be estimated by a benchmark experiment. Classification models can be fit on $\{(X_1, \Delta_1), ..., (X_n, \Delta_n)\}$ where $(X_i, \Delta_i) \in \mathcal{D}_{train}$. Whilst not an exact test, if any model significantly outperforms a baseline, then this may indicate censoring is informative. This is demonstrated in (**tab-car-predcens?**), in which a logistic regression outperforms a featureless baseline in correctly predicting if an observation is censored when censoring is informative, but is no better than the baseline when censoring is uninformative.

Table 20.1: Estimating censoring dependence by prediction. `Sim1` is informative censoring and `Sim7` is uninformative. Logistic regression is compared to a featureless baseline with the Brier score with standard errors. Censoring can be significantly predicted to 95% confidence when informative (`Sim1`) but not when uninformative (`Sim7`).

Data	Baseline	Logistic Regression
`Sim1`	0.20 (0.14, 0.26)	0.02 (0.01, 0.03)
`Sim7`	0.19 (0.14, 0.24)	0.16 (0.13, 0.19)

21

FAQs and Outlook

! Page coming soon!

We are working on this page and it will be available soon!

21.1 Common problems in survival analysis

21.1.1 Data cleaning

Events at t=0

Throughout this book we have defined survival times taking values in the non-negative Reals (zero inclusive) $\mathbb{R}_{\geq 0}$. In practice, model implementations assume time is over the positive Reals (zero exclusive). One must therefore consider how to deal with subjects that experience the outcome at 0. There is no established best practice for dealing with this case as the answer may be data-dependent. Possible choices include:

1. Deleting all data where the outcome occurs at $t = 0$, this may be appropriate if it only happens in a small number of observations and therefore deletion is unlikely to bias predictions;
2. Update the survival time to the next smallest observed survival time. For example, if the first observation to experience the event after $t = 0$ happens at $t = 0.1$, then set 0.1 as the survival time for any observation experiencing the event at $t = 0$. Note this method will not be appropriate when data is over a long period, for example if measuring time over years, then there could be a substantial difference between $t = 0$ and $t = 1$;
3. Update the survival time to a very small value ϵ that makes sense given the context of the data, e.g., $\epsilon = 0.0001$.

Continuous v Discrete Time

We defined survival tasks throughout this book assuming continuous time predictions in $\mathbb{R}_{\geq 0}$. In practice, many outcomes in survival analysis are recorded on a discrete scale, such as in medical statistics where outcomes are observed on a yearly, daily, monthly, hourly, etc. basis. Whilst discrete-time survival analysis exists for this purpose (Chapter 18), software implementations overwhelming use theory from the 'continuous-time setting. There has not been a lot of research into whether discrete-time methods outperform continuous-time methods when correctly applied to discrete data, however available experiments do not indicate that discrete methods outperform their continuous counterparts (Suresh, Severn,

and Ghosh 2022). Therefore it is recommended to use available software implementations, even when data is recorded on a discrete scale.

21.1.2 Evaluation and prediction

- Which time points to make predictions for?
-

21.1.3 Choosing models and measures

Choosing models

In contrast to measure selection, selecting models is more straightforward and the same heuristics from regression and classification largely apply to survival analysis. Firstly, for low-dimensional data, many experiments have demonstrated that machine learning may not improve upon more standard statistical methods (Christodoulou et al. 2019) and the same holds for survival analysis (Burk et al. 2024). Therefore the cost that comes with using machine learning – lower interpretability, longer training time – is unlikely to provide any performance benefits when a dataset has relatively few covariates. In settings where machine learning is more useful, then the choice largely falls into the four model classes discussed in this book: random forests, support vector machines, boosting, and neural networks (deep learning). If you have access to sufficient computational resources, then it is always worthwhile including at least one model from each class in a benchmark experiment, as models perform differently depending on the data type. However, without significant resources, the rules-of-thumb below can provide a starting point for smaller experiments.

Random survival forests and boosting methods are both good all-purpose methods that can handle different censoring types and competing risks settings. In single-event settings both have been shown to perform well on high-dimensional data, outperforming other model classes (Spooner et al. 2020). Forests require less tuning than boosting methods and the choice of hyperparameters if often more intuitive. Therefore, we generally recommend forests as the first choice for high-dimensional data. Given more resources, boosting methods such as *xgboost* are powerful methods to improve the predictive performance of classical measures. Survival support vector machines do not appear to work well in practice and to-date we have not seen any real-world use of SSVMs, therefore we generally do not recommend use of SVMs without robust training and testing first.

Neural networks are incredibly data-dependent. Moreover, given a huge increase in research into this area (Wiegrebe et al. 2024), there are no clear heuristics for recommending when to use neural networks and then which particular algorithms to use. With enough fine-tuning we have found that neural networks can work well but still without outperforming other methods. Where neural networks may shine is going beyond tabular data to incorporate other modalities, but again this area of research for survival analysis is still nascent.

Choosing measures

There are many survival measures to choose from and selecting the right one for the task might seem daunting. We have put together a few heuristics to support decision making. Evaluation should always be according to the goals of analysis, which means using discrimination measures to evaluate rankings, calibration measures to evaluate average performance, and scoring rules to evaluate overall performance and distribution predictions.

For discrimination measures, we recommend Harrell's and Uno's C. Whilst others can assess

time-dependent trends, these are also captured in scoring rules. In practice the choice of measure matters less than ensuring your reporting is transparent and honest (Therneau and Atkinson 2024; R. Sonabend, Whittles, et al. 2021).

To assess a single model's calibration, graphical comparisons to the Kaplan-Meier provide a useful and interpretable method to quickly see if a model is a good fit to the data (Section 8.2.1). When choosing between models, we recommend D-calibration, which can be meaningful optimized and thus used for comparison.

When picking scoring rules, we recommend using both the ISBS and RCLL. If a model outperforms another with respect to both measures then that can be a strong indicator of performance. When reporting scoring rules, we recommend the ERV representation which provides a meaningful interpretation as 'performance increase over baseline'.

Given the lack of research, if you are interested in survival time predictions then treat evaluation with caution and check for new developments in the literature.

For automated model optimization, we recommend tuning with a scoring rule, which should capture discrimination and calibration simultaneously [Rindt et al. (2022); Yanagisawa (2023); FIXME ECML]. Though if you are only ever using a model for ranking, then we recommend tuning with Uno's C. Whilst it does have higher variance compared to other concordance measures (Rahman et al. 2017; Schmid and Potapov 2012), it performs better than Harrell's C as censoring increases (Rahman et al. 2017).

Interpreting survival models

Interpreting models is increasingly important as we rely on more complex 'black-box' models (Molnar 2019). Classic methods that test if a model is fit well to data, such as the AIC and BIC, have been extended to survival models however are limited in application to 'classical' models (Chapter 11) only (Liang and Zou 2008; Volinsky and Raftery 2000). As a more flexible alternative, any of the calibration measures in Chapter 8 can be used to evaluate a model's fit to data. To assess algorithmic fairness, the majority of measures discussed in Part II can be used to detect bias in a survival context (R. Sonabend et al. 2022). Gold-standard interpretability methods such as SHAP and LIME (Molnar 2019) can be extended to survival analysis off-shelf (Langbein et al. 2024), and time-dependent extensions also exist to observe the impact of variables on the survival probability over time (Krzyziński et al. 2023; Langbein et al. 2024).

21.2 What's next for MLSA?

Exercises

TODO (150-200 WORDS)

> ! Page coming soon!
>
> **We are working on this page and it will be available soon!**

References

Aalen, Odd. 1978. "Nonparametric Inference for a Family of Counting Processes." *The Annals of Statistics* 6 (4): 701–26.

Aalen, Odd O., and Søren Johansen. 1978. "An Empirical Transition Matrix for Non-Homogeneous Markov Chains Based on Censored Observations." *Scandinavian Journal of Statistics* 5 (3): 141–50. https://www.jstor.org/stable/4615704.

Akritas, Michael G. 1994. "Nearest Neighbor Estimation of a Bivariate Distribution Under Random Censoring." *Ann. Statist.* 22 (3): 1299–1327. https://doi.org/10.1214/aos/1176325630.

Akritas, Michael G., and Michael P. LaValley. 2005. "A Generalized Product-Limit Estimator for Truncated Data." *Nonparametric Statistics*, September. https://doi.org/10.1080/10485250500038637.

Alberge, Julie, Vincent Maladière, Olivier Grisel, Judith Abécassis, and Gaël Varoquaux. 2024. "Survival Models: Proper Scoring Rule and Stochastic Optimization with Competing Risks." https://arxiv.org/abs/2410.16765.

———. 2025. "P52 - Survival Models: Proper Scoring Rule and Stochastic Optimization with Competing Risks." *Journal of Epidemiology and Population Health* 73: 203083. https://doi.org/https://doi.org/10.1016/j.jeph.2025.203083.

Allignol, Arthur, Jan Beyersmann, and Martin Schumacher. 2008. "Mvna an r Package for the Nelson-Aalen Estimator in Multistate Models." *R News* 8 (2): 48–50.

Andersen, Per Kragh, Mette Gerster Hansen, and John P. Klein. 2004. "Regression Analysis of Restricted Mean Survival Time Based on Pseudo-Observations." *Lifetime Data Analysis* 10 (4): 335–50. https://doi.org/10.1007/s10985-004-4771-0.

Andres, Axel, Aldo Montano-Loza, Russell Greiner, Max Uhlich, Ping Jin, Bret Hoehn, David Bigam, James Andrew Mark Shapiro, and Norman Mark Kneteman. 2018. "A novel learning algorithm to predict individual survival after liver transplantation for primary sclerosing cholangitis." *PLOS ONE* 13 (3): e0193523. https://doi.org/10.1371/journal.pone.0193523.

Angus, John E. 1994. "The Probability Integral Transform and Related Results." *SIAM Review* 36 (4): 652–54. http://www.jstor.org/stable/2132726.

Antolini, Laura, Patrizia Boracchi, and Elia Biganzoli. 2005. "A time-dependent discrimination index for survival data." *Statistics in Medicine* 24 (24): 3927–44. https://doi.org/10.1002/sim.2427.

Austin, Peter C., and Jason P. Fine. 2017. "Practical Recommendations for Reporting Fine-Gray Model Analyses for Competing Risk Data." *Statistics in Medicine* 36 (27): 4391–4400. https://doi.org/https://doi.org/10.1002/sim.7501.

Austin, Peter C., Frank E. Harrell Jr, and David van Klaveren. 2020. "Graphical Calibration Curves and the Integrated Calibration Index (ICI) for Survival Models." *Statistics in Medicine* 39 (21): 2714–42. https://doi.org/https://doi.org/10.1002/sim.8570.

Austin, Peter C., Douglas S. Lee, and Jason P. Fine. 2016. "Introduction to the Analysis of Survival Data in the Presence of Competing Risks." *Circulation* 133 (6): 601–9. https://doi.org/10.1161/CIRCULATIONAHA.115.017719.

Austin, Peter C., Hein Putter, Douglas S. Lee, and Ewout W. Steyerberg. 2022. "Estimation

of the Absolute Risk of Cardiovascular Disease and Other Events: Issues with the Use of Multiple Fine-Gray Subdistribution Hazard Models." *Circulation: Cardiovascular Quality and Outcomes* 15 (2): e008368. https://doi.org/10.1161/CIRCOUTCOMES.121.008368.

Avati, Anand, Tony Duan, Sharon Zhou, Kenneth Jung, Nigam H. Shah, and Andrew Ng. 2020. "Countdown Regression: Sharp and Calibrated Survival Predictions." In *Proceedings of Machine Learning Research*, 145–55. https://proceedings.mlr.press/v115/avati20a.html http://arxiv.org/abs/1806.08324.

Beaulac, Cédric, Jeffrey S. Rosenthal, Qinglin Pei, Debra Friedman, Suzanne Wolden, and David Hodgson. 2020. "An Evaluation of Machine Learning Techniques to Predict the Outcome of Children Treated for Hodgkin-Lymphoma on the AHOD0031 Trial." *Applied Artificial Intelligence* 34 (14): 1100–1114. https://doi.org/10.1080/08839514.2020.1815151.

Becker, Marc, Lennart Schneider, and Sebastian Fischer. 2024. "Hyperparameter Optimization." In *Applied Machine Learning Using mlr3 in R*, edited by Bernd Bischl, Raphael Sonabend, Lars Kotthoff, and Michel Lang. CRC Press. https://mlr3book.mlr-org.com/hyperparameter_optimization.html.

Bello, Ghalib A, Timothy J W Dawes, Jinming Duan, Carlo Biffi, Antonio de Marvao, Luke S G E Howard, J Simon R Gibbs, et al. 2019. "Deep-learning cardiac motion analysis for human survival prediction." *Nature Machine Intelligence* 1 (2): 95–104. https://doi.org/10.1038/s42256-019-0019-2.

Benavoli, Alessio, Giorgio Corani, Janez Demšar, and Marco Zaffalon. 2017. "Time for a Change: A Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis." *Journal of Machine Learning Research* 18 (77): 1–36. http://jmlr.org/papers/v18/16-305.html.

Bender, Andreas, David Rügamer, Fabian Scheipl, and Bernd Bischl. 2021. "A General Machine Learning Framework for Survival Analysis." In *Machine Learning and Knowledge Discovery in Databases*, edited by Frank Hutter, Kristian Kersting, Jefrey Lijffijt, and Isabel Valera, 158–73. Lecture Notes in Computer Science. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-67664-3_10.

Bender, Andreas, and Fabian Scheipl. 2018. "pammtools: Piece-wise exponential Additive Mixed Modeling tools." *arXiv:1806.01042 [Stat]*. http://arxiv.org/abs/1806.01042.

Bennett, Steve. 1983. "Analysis of survival data by the proportional odds model." *Statistics in Medicine* 2 (2): 273–77. https://doi.org/https://doi.org/10.1002/sim.4780020223.

Beyersmann, Jan, Arthur Allignol, and Martin Schumacher. 2012. *Competing Risks and Multistate Models with R*. Use R! New York: Springer.

Biganzoli, E M, F Ambrogi, and P Boracchi. 2009. "Partial logistic artificial neural networks (PLANN) for flexible modeling of censored survival data." In *2009 International Joint Conference on Neural Networks*, 340–46. https://doi.org/10.1109/IJCNN.2009.5178824.

Biganzoli, Elia, Patrizia Boracchi, Luigi Mariani, and Ettore Marubini. 1998. "Feed forward neural networks for the analysis of censored survival data: a partial logistic regression approach." *Statistics in Medicine* 17 (10): 1169–86. https://doi.org/10.1002/(SICI)1097-0258(19980530)17:10%3C1169::AID-SIM796%3E3.0.CO;2-D.

Binder, Harald. 2013. "CoxBoost: Cox models by likelihood based boosting for a single survival endpoint or competing risks." CRAN.

Binder, Harald, Arthur Allignol, Martin Schumacher, and Jan Beyersmann. 2009. "Boosting for High-Dimensional Time-to-Event Data with Competing Risks." *Bioinformatics* 25 (7): 890–96. https://doi.org/10.1093/bioinformatics/btp088.

Binder, Harald, and Martin Schumacher. 2008. "Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models." *BMC Bioinformatics* 9 (1): 14. https://doi.org/10.1186/1471-2105-9-14.

Bischl, Bernd, O. Mersmann, H. Trautmann, and C. Weihs. 2012. "Resampling Meth-

ods for Meta-Model Validation with Recommendations for Evolutionary Computation." *Evolutionary Computation* 20 (2): 249–75. https://doi.org/10.1162/EVCO_a_00069.

Bischl, Bernd, Raphael Sonabend, Lars Kotthoff, and Michel Lang, eds. 2024. *Applied Machine Learning Using mlr3 in R*. CRC Press. https://mlr3book.mlr-org.com.

Bishop, Christopher M. 2006. *Pattern recognition and machine learning*. springer.

Blanche, Paul, Jean-François Dartigues, and Hélène Jacqmin-Gadda. 2013. "Review and comparison of ROC curve estimators for a time-dependent outcome with marker-dependent censoring." *Biometrical Journal* 55 (5): 687–704. https://doi.org/10.1002/bimj.201200045.

Blanche, Paul, Aurélien Latouche, and Vivian Viallon. 2012. "Time-dependent AUC with right-censored data: a survey study," October. https://doi.org/10.1007/978-1-4614-8981-8_11.

Bland, J Martin, and Douglas G. Altman. 2004. "The logrank test." *BMJ (Clinical Research Ed.)* 328 (7447): 1073. https://doi.org/10.1136/bmj.328.7447.1073.

Bommert, Andrea, Thomas Welchowski, Matthias Schmid, and Jörg Rahnenführer. 2021. "Benchmark of Filter Methods for Feature Selection in High-Dimensional Gene Expression Survival Data." *Briefings in Bioinformatics* 23 (1): bbab354. https://doi.org/10.1093/bib/bbab354.

Bou-Hamad, Imad, Denis Larocque, and Hatem Ben-Ameur. 2011. "A review of survival trees." *Statist. Surv.* 5: 44–71. https://doi.org/10.1214/09-SS047.

Bower, Hannah, Michael J Crowther, Mark J Rutherford, Therese M.-L. Andersson, Mark Clements, Xing-Rong Liu, Paul W Dickman, and Paul C Lambert. 2019. "Capturing simple and complex time-dependent effects using flexible parametric survival models: A simulation study." *Communications in Statistics - Simulation and Computation*, July, 1–17. https://doi.org/10.1080/03610918.2019.1634201.

Breiman, Leo. 1996. "Bagging Predictors." *Machine Learning* 24 (2): 123–40. https://doi.org/10.1023/A:1018054314350.

Breiman, Leo, and Philip Spector. 1992. "Submodel Selection and Evaluation in Regression. The X-Random Case." *International Statistical Review / Revue Internationale de Statistique* 60 (3): 291–319. https://doi.org/10.2307/1403680.

Breiman, L, J Friedman, C J Stone, and R A Olshen. 1984. *Classification and Regression Trees*. The Wadsworth and Brooks-Cole Statistics-Probability Series. Taylor & Francis. https://books.google.co.uk/books?id=JwQx-WOmSyQC.

Breslow, N. 1972. "Discussion following 'Regression models and life tables' by D. R. Cox." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 34 (2): 187–220.

Brier, Glenn. 1950. "Verification of forecasts expressed in terms of probability." *Monthly Weather Review* 78 (1): 1–3.

Broström, Göran. 1987. "The Influence of Mother's Death on Infant Mortality: A Case Study in Matched Data Survival Analysis." *Scandinavian Journal of Statistics* 14 (2): 113–23. https://www.jstor.org/stable/4616055.

———. 2024. *Eha: Event History Analysis*. https://cran.r-project.org/package=eha.

Buckley, Jonathan, and Ian James. 1979. "Linear Regression with Censored Data." *Biometrika* 66 (3): 429–36. https://doi.org/10.2307/2335161.

Buhlmann, Peter. 2006. "Boosting for high-dimensional linear models." *Ann. Statist.* 34 (2): 559–83. https://doi.org/10.1214/009053606000000092.

Buhlmann, Peter, and Torsten Hothorn. 2007. "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statist. Sci.* 22 (4): 477–505. https://doi.org/10.1214/07-STS242.

Bühlmann, Peter, and Bin Yu. 2003. "Boosting With the L2 Loss." *Journal of the American Statistical Association* 98 (462): 324–39. https://doi.org/10.1198/016214503000125.

Burk, Lukas, John Zobolas, Bernd Bischl, Andreas Bender, Marvin N. Wright, and Raphael Sonabend. 2024. "A Large-Scale Neutral Comparison Study of Survival Models on Low-Dimensional Data," June. http://arxiv.org/abs/2406.04098.

Casalicchio, Giuseppe, and Lukas Burk. 2024. "Evaluation and Benchmarking." In *Applied Machine Learning Using mlr3 in R*, edited by Bernd Bischl, Raphael Sonabend, Lars Kotthoff, and Michel Lang. CRC Press. https://mlr3book.mlr-org.com/chapters/chapter3/evaluation_and_benchmarking.html.

Chambless, Lloyd E, and Guoqing Diao. 2006. "Estimation of time-dependent area under the ROC curve for long-term risk prediction." *Statistics in Medicine* 25 (20): 3474–86. https://doi.org/10.1002/sim.2299.

Chandrashekar, Girish, and Ferat Sahin. 2014. "A Survey on Feature Selection Methods." *Computers & Electrical Engineering* 40 (1): 16–28. https://doi.org/https://doi.org/10.1016/j.compeleceng.2013.11.024.

Chapfuwa, Paidamoyo, Chenyang Tao, Chunyuan Li, Courtney Page, Benjamin Goldstein, Lawrence Carin Duke, and Ricardo Henao. 2018. "Adversarial Time-to-Event Modeling." In *Proceedings of the 35th International Conference on Machine Learning*, edited by Jennifer Dy and Andreas Krause, 80:735–44. Proceedings of Machine Learning Research. PMLR. https://proceedings.mlr.press/v80/chapfuwa18a.html.

Chen, Tianqi, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. 2020. "xgboost: Extreme Gradient Boosting." CRAN. https://cran.r-project.org/package=xgboost.

Chen, Yen-Chen, Wan-Chi Ke, and Hung-Wen Chiu. 2014. "Risk classification of cancer survival using ANN with gene expression data from multiple laboratories." *Computers in Biology and Medicine* 48: 1–7. https://doi.org/https://doi.org/10.1016/j.compbiomed.2014.02.006.

Chen, Yifei, Zhenyu Jia, Dan Mercola, and Xiaohui Xie. 2013. "A Gradient Boosting Algorithm for Survival Analysis via Direct Optimization of Concordance Index." Edited by Lev Klebanov. *Computational and Mathematical Methods in Medicine* 2013: 873595. https://doi.org/10.1155/2013/873595.

Ching, Travers, Xun Zhu, and Lana X Garmire. 2018. "Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data." *PLOS Computational Biology* 14 (4): e1006076. https://doi.org/10.1371/journal.pcbi.1006076.

Choodari-Oskooei, Babak, Patrick Royston, and Mahesh K. B. Parmar. 2012a. "A simulation study of predictive ability measures in a survival model I: Explained variation measures." *Statistics in Medicine* 31 (23): 2627–43. https://doi.org/10.1002/sim.4242.

———. 2012b. "A simulation study of predictive ability measures in a survival model II: explained randomness and predictive accuracy." *Statistics in Medicine* 31 (23): 2644–59. https://doi.org/10.1002/sim.4242.

Christodoulou, Evangelia, Jie Ma, Gary S Collins, Ewout W Steyerberg, Jan Y Verbakel, and Ben Van Calster. 2019. "A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models." *Journal of Clinical Epidemiology* 110 (June): 12–22. https://doi.org/10.1016/j.jclinepi.2019.02.004.

Ciampi, Antonio, Sheilah A Hogg, Steve McKinney, and Johanne Thiffault. 1988. "RECPAM: a computer program for recursive partition and amalgamation for censored survival data and other situations frequently occurring in biostatistics. I. Methods and program features." *Computer Methods and Programs in Biomedicine* 26 (3): 239–56. https://doi.org/https://doi.org/10.1016/0169-2607(88)90004-1.

Ciampi, Antonio, Johanne Thiffault, Jean Pierre Nakache, and Bernard Asselain. 1986. "Stratification by stepwise regression, correspondence analysis and recursive partition: a comparison of three methods of analysis for survival data with covariates." *Computational Statistics and Data Analysis* 4 (3): 185–204. https://doi.org/10.1016/0167-9473(86)90033-2.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. 2015. "Fast and accurate deep network learning by exponential linear units (elus)." *arXiv Preprint*

arXiv:1511.07289.

Collett, David. 2014. *Modelling Survival Data in Medical Research*. 3rd ed. CRC.

Cook, Richard J., and Jerald F. Lawless. 2007. *The Statistical Analysis of Recurrent Events*. Statistics for Biology and Health. New York, NY: Springer. https://doi.org/10.1007/978-0-387-69810-6.

Cortes, Corinna, and Vladimir Vapnik. 1995. "Support-Vector Networks." *Machine Learning* 20: 273–97. https://doi.org/10.1007/BF00994018.

Cox, D. R. 1972. "Regression Models and Life-Tables." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 34 (2): 187–220.

———. 1975. "Partial Likelihood." *Biometrika* 62 (August): 269–76. https://doi.org/10.1093/biomet/62.2.269.

Cui, Lei, Hansheng Li, Wenli Hui, Sitong Chen, Lin Yang, Yuxin Kang, Qirong Bo, and Jun Feng. 2020. "A deep learning-based framework for lung cancer survival analysis with biomarker interpretation." *BMC Bioinformatics* 21 (1): 112. https://doi.org/10.1186/s12859-020-3431-z.

Data Study Group Team. 2020. "Data Study Group Final Report: Great Ormond Street Hospital." https://doi.org/10.5281/zenodo.3670726.

Dawid, A P. 1984. "Present Position and Potential Developments: Some Personal Views: Statistical Theory: The Prequential Approach." *Journal of the Royal Statistical Society. Series A (General)* 147 (2): 278–92. https://doi.org/10.2307/2981683.

Dawid, A Philip. 1986. "Probability Forecasting." *Encyclopedia of Statistical Sciences* 7: 210–18.

Dawid, A Philip, and Monica Musio. 2014. "Theory and Applications of Proper Scoring Rules." *Metron* 72 (2): 169–83. https://arxiv.org/abs/arXiv:1401.0398v1.

de Wreede, Liesbeth C., Marta Fiocco, and Hein Putter. 2011. "mstate: An R Package for the Analysis of Competing Risks and Multi-State Models." *Journal of Statistical Software* 38 (7): 1–30. https://doi.org/10.18637/jss.v038.i07.

Demler, Olga V, Nina P Paynter, and Nancy R Cook. 2015. "Tests of calibration and goodness-of-fit in the survival setting." *Statistics in Medicine* 34 (10): 1659–80. https://doi.org/10.1002/sim.6428.

Demšar, Janez. 2006. "Statistical comparisons of classifiers over multiple data sets." *Journal of Machine Learning Research* 7 (1): 1–30.

Dietterich, Thomas G. 1998. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms." *Neural Computation* 10 (7): 1895–1923. https://doi.org/10.1162/089976698300017197.

Efron, Bradley. 1988. "Logistic Regression, Survival Analysis, and the Kaplan-Meier Curve." *Journal of the American Statistical Association* 83 (402): 414–25. https://doi.org/10.2307/2288857.

Efron, Bradley, and Robert Tibshirani. 1997. "Improvements on Cross-Validation: The .632+ Bootstrap Method." *Journal of the American Statistical Association* 92 (438): 548–60. http://www.jstor.org/stable/2965703.

Evers, Ludger, and Claudia-Martina Messow. 2008. "Sparse kernel methods for high-dimensional survival data." *Bioinformatics* 24 (14): 1632–38.

Faraggi, David, and Richard Simon. 1995. "A neural network model for survival data." *Statistics in Medicine* 14 (1): 73–82. https://doi.org/10.1002/sim.4780140108.

Fine, Jason P., and Robert J. Gray. 1999. "A Proportional Hazards Model for the Subdistribution of a Competing Risk." *Journal of the American Statistical Association* 94 (446): 496–509. http://www.jstor.org/stable/2670170.

Fleming, Thomas R, Judith R O'Fallon, Peter C O'Brien, and David P Harrington. 1980. "Modified Kolmogorov-Smirnov Test Procedures with Application to Arbitrarily Right-Censored Data." *Biometrics* 36 (4): 607–25. https://doi.org/10.2307/2556114.

Foss, Natalie, and Lars Kotthoff. 2024. "Data and Basic Modeling." In *Applied Machine Learning Using mlr3 in R*, edited by Bernd Bischl, Raphael Sonabend, Lars Kotthoff, and Michel Lang. CRC Press. https://mlr3book.mlr-org.com/data_and_basic_modeling.html.

Fotso, Stephane. 2018. "Deep Neural Networks for Survival Analysis Based on a Multi-Task Framework." *arXiv Preprint arXiv:1801.05512*, January. http://arxiv.org/abs/1801.05512.

Fouodo, Cesaire J K, I Konig, C Weihs, A Ziegler, and M Wright. 2018. "Support vector machines for survival analysis with R." *The R Journal* 10 (July): 412–23.

Freund, Yoav, and Robert E Schapire. 1996. "Experiments with a new boosting algorithm." In. Citeseer.

Friedman, Jerome. 1999. "Stochastic Gradient Boosting." *Computational Statistics & Data Analysis* 38 (March): 367–78. https://doi.org/10.1016/S0167-9473(01)00065-2.

Friedman, Jerome H. 2001. "Greedy Function Approximation: A Gradient Boosting Machine." *The Annals of Statistics* 29 (5): 1189–1232. http://www.jstor.org/stable/2699986.

Friedman, Michael. 1982. "Piecewise exponential models for survival data with covariates." *The Annals of Statistics* 10 (1): 101–13.

Fritsch, Stefan, Frauke Guenther, and Marvin N. Wright. 2019. "neuralnet: Training of Neural Networks." CRAN. https://cran.r-project.org/package=neuralnet.

Geloven, Nan van, Daniele Giardiello, Edouard F Bonneville, Lucy Teece, Chava L Ramspek, Maarten van Smeden, Kym I E Snell, et al. 2022. "Validation of Prediction Models in the Presence of Competing Risks: A Guide Through Modern Methods." *BMJ* 377. https://doi.org/10.1136/bmj-2021-069249.

Gensheimer, Michael F., and Balasubramanian Narasimhan. 2018. "A Simple Discrete-Time Survival Model for Neural Networks," 1–17. https://doi.org/arXiv:1805.00917v3.

Gensheimer, Michael F, and Balasubramanian Narasimhan. 2019. "A scalable discrete-time survival model for neural networks." *PeerJ* 7: e6257.

Gerds, Thomas A, and Martin Schumacher. 2006. "Consistent Estimation of the Expected Brier Score in General Survival Models with Right-Censored Event Times." *Biometrical Journal* 48 (6): 1029–40. https://doi.org/10.1002/bimj.200610301.

Géron, Aurélien. 2019. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly. https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

Giolo, Suely. 2004. "Turnbull's Nonparametric Estimator for Interval-Censored Data," January.

Giunchiglia, Eleonora, Anton Nemchenko, and Mihaela van der Schaar. 2018. "Rnn-surv: A deep recurrent model for survival analysis." In *International Conference on Artificial Neural Networks*, 23–32. Springer.

Gneiting, Tilmann, and Adrian E Raftery. 2007. "Strictly Proper Scoring Rules, Prediction, and Estimation." *Journal of the American Statistical Association* 102 (477): 359–78. https://doi.org/10.1198/016214506000001437.

Goli, Shahrbanoo, Hossein Mahjub, Javad Faradmal, Hoda Mashayekhi, and Ali-Reza Soltanian. 2016. "Survival Prediction and Feature Selection in Patients with Breast Cancer Using Support Vector Regression." Edited by Francesco Pappalardo. *Computational and Mathematical Methods in Medicine* 2016: 2157984. https://doi.org/10.1155/2016/2157984.

Goli, Shahrbanoo, Hossein Mahjub, Javad Faradmal, and Ali-Reza Soltanian. 2016. "Performance Evaluation of Support Vector Regression Models for Survival Analysis: A Simulation Study." *International Journal of Advanced Computer Science and Applications* 7 (June). https://doi.org/10.14569/IJACSA.2016.070650.

Gompertz, Benjamin. 1825. "On the Nature of the Function Expressive of the Law of Human Mortality, and on a New Mode of Determining the Value of Life Contingencies."

Philosophical Transactions of the Royal Society of London 115: 513–83.
Gönen, Mithat, and Glenn Heller. 2005. "Concordance Probability and Discriminatory Power in Proportional Hazards Regression." *Biometrika* 92 (4): 965–70.
Good, I J. 1952. "Rational Decisions." *Journal of the Royal Statistical Society. Series B (Methodological)* 14 (1): 107–14. http://www.jstor.org/stable/2984087.
Gordon, Louis, and Richard A Olshen. 1985. "Tree-structured survival analysis." *Cancer Treatment Reports* 69 (10): 1065–69.
Graf, Erika, Claudia Schmoor, Willi Sauerbrei, and Martin Schumacher. 1999. "Assessment and comparison of prognostic classification schemes for survival data." *Statistics in Medicine* 18 (17-18): 2529–45. https://doi.org/10.1002/(SICI)1097-0258(19990915/30)18:17/18%3C2529::AID-SIM274%3E3.0.CO;2-5.
Graf, Erika, and Martin Schumacher. 1995. "An Investigation on Measures of Explained Variation in Survival Analysis." *Journal of the Royal Statistical Society. Series D (The Statistician)* 44 (4): 497–507. https://doi.org/10.2307/2348898.
Gray, Robert J. 1988. "A Class of K-Sample Tests for Comparing the Cumulative Incidence of a Competing Risk." *The Annals of Statistics* 16 (3): 1141–54. https://doi.org/10.1214/aos/1176350951.
Gressmann, Frithjof, Franz J. Király, Bilal Mateen, and Harald Oberhauser. 2018. "Probabilistic supervised learning." https://doi.org/10.1002/iub.552.
Guyon, Isabelle, and André Elisseeff. 2003. "An Introduction to Variable and Feature Selection." *The Journal of Machine Learning Research* 3 (March): 1157–82.
Habibi, Danial, Mohammad Rafiei, Ali Chehrei, Zahra Shayan, and Soheil Tafaqodi. 2018. "Comparison of Survival Models for Analyzing Prognostic Factors in Gastric Cancer Patients." *Asian Pacific Journal of Cancer Prevention : APJCP* 19 (3): 749–53. https://doi.org/10.22034/APJCP.2018.19.3.749.
Haider, Humza, Bret Hoehn, Sarah Davis, and Russell Greiner. 2020. "Effective ways to build and evaluate individual survival distributions." *Journal of Machine Learning Research* 21 (85): 1–63.
Han, Ilkyu, June Hyuk Kim, Heeseol Park, Han-Soo Kim, and Sung Wook Seo. 2018. "Deep learning approach for survival prediction for patients with synovial sarcoma." *Tumor Biology* 40 (9): 1010428318799264. https://doi.org/10.1177/1010428318799264.
Han, Kyunghwa, and Inkyung Jung. 2022. "Restricted Mean Survival Time for Survival Analysis: A Quick Guide for Clinical Researchers." *Korean Journal of Radiology* 23 (5): 495–99. https://doi.org/10.3348/kjr.2022.0061.
Harrell, F E Jr, K L Lee, R M Califf, D B Pryor, and R A Rosati. 1984. "Regression modelling strategies for improved prognostic prediction." *Statistics in Medicine* 3 (2): 143–52. https://doi.org/10.1002/sim.4780030207.
Harrell, Frank E., Robert M. Califf, and David B. Pryor. 1982. "Evaluating the yield of medical tests." *JAMA* 247 (18): 2543–46. http://dx.doi.org/10.1001/jama.1982.03320430047030.
Hartman, Nicholas, Sehee Kim, Kevin He, and John D. Kalbfleisch. 2022. "Concordance Indices with Left-Truncated and Right-Censored Data." *Biometrics* 79 (3): 1624–34. https://doi.org/10.1111/biom.13714.
Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer New York Inc.
Heagerty, Patrick J., Thomas Lumley, and Margaret S. Pepe. 2000. "Time-Dependent ROC Curves for Censored Survival Data and a Diagnostic Marker." *Biometrics* 56 (2): 337–44. https://doi.org/10.1111/j.0006-341X.2000.00337.x.
Heagerty, Patrick J, and Yingye Zheng. 2005. "Survival Model Predictive Accuracy and ROC Curves." *Biometrics* 61 (1): 92–105. https://doi.org/10.1111/j.0006-341X.2005.030814.x.
Henderson, and Velleman. 1981. "Building multiple regression models interactively." *Biometrics* 37: 391–411.

Herrmann, Moritz, Philipp Probst, Roman Hornung, Vindi Jurinovic, and Anne-Laure Boulesteix. 2021. "Large-scale benchmark study of survival prediction methods using multi-omics data." *Briefings in Bioinformatics* 22 (3). https://doi.org/10.1093/bib/bbaa167.

Hielscher, Thomas, Manuela Zucknick, Wiebke Werft, and Axel Benner. 2010. "On the Prognostic Value of Gene Expression Signatures for Censored Data BT - Advances in Data Analysis, Data Handling and Business Intelligence." In, edited by Andreas Fink, Berthold Lausen, Wilfried Seidel, and Alfred Ultsch, 663–73. Berlin, Heidelberg: Springer Berlin Heidelberg.

Hinchliffe, Sally R, and Paul C Lambert. 2013. "Flexible Parametric Modelling of Cause-Specific Hazards to Estimate Cumulative Incidence Functions." *BMC Medical Research Methodology* 13: 13. https://doi.org/10.1186/1471-2288-13-13.

Hornung, Roman, Malte Nalenz, Lennart Schneider, Andreas Bender, Ludwig Bothmann, Bernd Bischl, Thomas Augustin, and Anne-Laure Boulesteix. 2023. "Evaluating Machine Learning Models in Non-Standard Settings: An Overview and New Findings." https://arxiv.org/abs/2310.15108.

Hosmer, David W, and Stanley Lemeshow. 1980. "Goodness of fit tests for the multiple logistic regression model." *Communications in Statistics-Theory and Methods* 9 (10): 1043–69.

Hosmer Jr, David W, Stanley Lemeshow, and Susanne May. 2011. *Applied survival analysis: regression modeling of time-to-event data*. Vol. 618. John Wiley & Sons.

Hothorn, Torsten, Peter Buehlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. 2020. "mboost: Model-Based Boosting." CRAN. https://cran.r-project.org/package=mboost.

Hothorn, Torsten, Peter Bühlmann, Sandrine Dudoit, Annette Molinaro, and Mark J Van Der Laan. 2005. "Survival ensembles." *Biostatistics* 7 (3): 355–73. https://doi.org/10.1093/biostatistics/kxj011.

Hothorn, Torsten, and Berthold Lausen. 2003. "On the exact distribution of maximally selected rank statistics." *Computational Statistics & Data Analysis* 43 (2): 121–37. https://doi.org/10.1016/S0167-9473(02)00225-6.

Hothorn, Torsten, Berthold Lausen, Axel Benner, and Martin Radespiel-Tröger. 2004. "Bagging survival trees." *Statistics in Medicine* 23 (1): 77–91. https://doi.org/10.1002/sim.1593.

Huang, Shigao, Jie Yang, Simon Fong, and Qi Zhao. 2020a. "Artificial intelligence in cancer diagnosis and prognosis: Opportunities and challenges." *Cancer Letters* 471: 61–71. https://doi.org/https://doi.org/10.1016/j.canlet.2019.12.007.

———. 2020b. "Artificial intelligence in cancer diagnosis and prognosis: Opportunities and challenges." *Cancer Letters* 471: 61–71. https://doi.org/https://doi.org/10.1016/j.canlet.2019.12.007.

Hung, Hung, and Chin-Tsang Chiang. 2010. "Estimation methods for time-dependent AUC models with survival data." *The Canadian Journal of Statistics / La Revue Canadienne de Statistique* 38 (1): 8–26. http://www.jstor.org/stable/27805213.

Hurvich, Clifford M, and Chih-Ling Tsai. 1989. "Regression and time series model selection in small samples." *Biometrika* 76 (2): 297–307. https://doi.org/10.1093/biomet/76.2.297.

Ishwaran, By Hemant, Udaya B Kogalur, Eugene H Blackstone, and Michael S Lauer. 2008. "Random survival forests." *The Annals of Statistics* 2 (3): 841–60. https://doi.org/10.1214/08-AOAS169.

Ishwaran, Hemant, Eugene H Blackstone, Claire E Pothier, and Michael S Lauer. 2004. "Relative Risk Forests for Exercise Heart Rate Recovery as a Predictor of Mortality." *Journal of the American Statistical Association* 99 (467): 591–600. https://doi.org/10.1198/016214504000000638.

Ishwaran, Hemant, and Udaya B Kogalur. 2018. "randomForestSRC." https://cran.r-project.org/package=randomForestSRC.

Ishwaran, Hemant, Udaya B Kogalur, Xi Chen, and Andy J Minn. 2011. "Random Survival Forests for High-Dimensional Data." *Statistical Analysis and Data Mining* 4 (1): 115–32. https://doi.org/10.1002/sam.

Jackson, Dan, Ian R. White, Shaun Seaman, Hannah Evans, Kathy Baisley, and James Carpenter. 2014. "Relaxing the independent censoring assumption in the Cox proportional hazards model using multiple imputation." *Statistics in Medicine* 33 (27): 4681–94. https://doi.org/10.1002/sim.6274.

Jacqmin-Gadda, Hélène, Paul Blanche, Emilie Chary, Célia Touraine, and Jean-François Dartigues. 2016. "Receiver Operating Characteristic Curve Estimation for Time to Event with Semicompeting Risks and Interval Censoring." *Statistical Methods in Medical Research* 25 (6): 2750–66. https://doi.org/10.1177/0962280214531691.

Jager, Kitty J, Paul C van Dijk, Carmine Zoccali, and Friedo W Dekker. 2008. "The analysis of survival data: the Kaplan–Meier method." *Kidney International* 74 (5): 560–65. https://doi.org/https://doi.org/10.1038/ki.2008.217.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An introduction to statistical learning*. Vol. 112. New York: Springer.

Jing, Bingzhong, Tao Zhang, Zixian Wang, Ying Jin, Kuiyuan Liu, Wenze Qiu, Liangru Ke, et al. 2019. "A deep survival analysis method based on ranking." *Artificial Intelligence in Medicine* 98: 1–9. https://doi.org/https://doi.org/10.1016/j.artmed.2019.06.001.

Johnson, Brent A, and Qi Long. 2011. "Survival ensembles by the sum of pairwise differences with application to lung cancer microarray studies." *Ann. Appl. Stat.* 5 (2A): 1081–101. https://doi.org/10.1214/10-AOAS426.

Kalbfleisch, J. D., and R. L. Prentice. 1973. "Marginal likelihoods based on Cox's regression and life model." *Biometrika* 60 (2): 267–78. https://doi.org/10.1093/biomet/60.2.267.

Kalbfleisch, John D, and Ross L Prentice. 2011. *The statistical analysis of failure time data*. Vol. 360. John Wiley & Sons.

Kamarudin, Adina Najwa, Trevor Cox, and Ruwanthi Kolamunnage-Dona. 2017. "Time-dependent ROC curve analysis in medical research: Current methods and applications." *BMC Medical Research Methodology* 17 (1): 1–19. https://doi.org/10.1186/s12874-017-0332-6.

Kaplan, E. L., and Paul Meier. 1958. "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association* 53 (282): 457–81. https://doi.org/10.2307/2281868.

Katzman, Jared L, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. 2018. "DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network." *BMC Medical Research Methodology* 18 (1): 24. https://doi.org/10.1186/s12874-018-0482-1.

Katzman, Jared, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. 2016. "Deep Survival: A Deep Cox Proportional Hazards Network," June.

Khan, Faisal M., and Valentina Bayer Zubek. 2008. "Support vector regression for censored data (SVRc): A novel tool for survival analysis." *Proceedings - IEEE International Conference on Data Mining, ICDM*, 863–68. https://doi.org/10.1109/ICDM.2008.50.

Király, Franz J, Bilal Mateen, and Raphael Sonabend. 2018. "NIPS - Not Even Wrong? A Systematic Review of Empirically Complete Demonstrations of Algorithmic Effectiveness in the Machine Learning and Artificial Intelligence Literature." *arXiv*, December. http://arxiv.org/abs/1812.07519.

Kirmani, S N U A, and Ramesh C Gupta. 2001. "On the Proportional Odds Model in Survival Analysis." *Annals of the Institute of Statistical Mathematics* 53 (2): 203–16. https://doi.org/10.1023/A:1012458303498.

Klein, John P, and Melvin L Moeschberger. 2003. *Survival analysis: techniques for censored and truncated data*. 2nd ed. Springer Science & Business Media.

Kleinbaum, David G, and Mitchel Klein. 1996. *Survival Analysis a Self-Learning Text*. Springer.

Kohavi, Ron. 1995. "A study of cross-validation and bootstrap for accuracy estimation and model selection." *Ijcai* 14 (2): 1137–45.

Korn, Edward L., and Richard Simon. 1990. "Measures of explained variation for survival data." *Statistics in Medicine* 9 (5): 487–503. https://doi.org/10.1002/sim.4780090503.

Korn, Edward L, and Richard Simon. 1991. "Explained Residual Variation, Explained Risk, and Goodness of Fit." *The American Statistician* 45 (3): 201–6. https://doi.org/10.2307/2684290.

Krzyziński, Mateusz, Mikołaj Spytek, Hubert Baniecki, and Przemysław Biecek. 2023. "SurvSHAP(t): Time-Dependent Explanations of Machine Learning Survival Models." *Knowledge-Based Systems* 262: 110234. https://doi.org/https://doi.org/10.1016/j.knosys.2022.110234.

Kuhn, Max, and Julia Silge. 2023. *Tidy Modeling with R*. https://www.tmwr.org/.

Kvamme, Håvard. 2018. "Pycox." https://pypi.org/project/pycox/.

Kvamme, Håvard, and Ørnulf Borgan. 2023. "The Brier Score Under Administrative Censoring: Problems and a Solution." *Journal of Machine Learning Research* 24 (2): 1–26. http://jmlr.org/papers/v24/19-1030.html.

Kvamme, Håvard, Ørnulf Borgan, and Ida Scheel. 2019. "Time-to-event prediction with neural networks and Cox regression." *Journal of Machine Learning Research* 20 (129): 1–30.

Land, Walker H, Xingye Qiao, Dan Margolis, and Ron Gottlieb. 2011. "A new tool for survival analysis: evolutionary programming/evolutionary strategies (EP/ES) support vector regression hybrid using both censored / non-censored (event) data." *Procedia Computer Science* 6: 267–72. https://doi.org/https://doi.org/10.1016/j.procs.2011.08.050.

Langbein, Sophie Hanna, Mateusz Krzyziński, Mikołaj Spytek, Hubert Baniecki, Przemysław Biecek, and Marvin N. Wright. 2024. "Interpretable Machine Learning for Survival Analysis." https://arxiv.org/abs/2403.10250.

Langford, John, Paul Mineiro, Alina Beygelzimer, and Hal Daume. 2016. "Learning Reductions that Really Work." *Proceedings of the IEEE* 104 (1).

Lao, Jiangwei, Yinsheng Chen, Zhi-Cheng Li, Qihua Li, Ji Zhang, Jing Liu, and Guangtao Zhai. 2017. "A Deep Learning-Based Radiomics Model for Prediction of Survival in Glioblastoma Multiforme." *Scientific Reports* 7 (1): 10353. https://doi.org/10.1038/s41598-017-10649-8.

LeBlanc, Michael, and John Crowley. 1992. "Relative Risk Trees for Censored Survival Data." *Biometrics* 48 (2): 411–25. https://doi.org/10.2307/2532300.

———. 1993. "Survival Trees by Goodness of Split." *Journal of the American Statistical Association* 88 (422): 457–67. https://doi.org/10.2307/2290325.

Lee, Changhee, William Zame, Jinsung Yoon, and Mihaela Van der Schaar. 2018. "DeepHit: A Deep Learning Approach to Survival Analysis With Competing Risks." *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (1). https://doi.org/10.1609/aaai.v32i1.11842.

Lee, Donald K K, Ningyuan Chen, and Hemant Ishwaran. 2019. "Boosted nonparametric hazards with time-dependent covariates." https://arxiv.org/abs/arXiv:1701.07926v6.

Li, Jialiang, and Shuangge Ma. 2011. "Time-Dependent ROC Analysis Under Diverse Censoring Patterns." *Statistics in Medicine* 30 (11): 1266–77. https://doi.org/https://doi.org/10.1002/sim.4178.

Li, Liang, Tom Greene, and Bo Hu. 2018. "A simple method to estimate the time-dependent receiver operating characteristic curve and the area under the curve with right censored

data." *Statistical Methods in Medical Research* 27 (8): 2264–78. https://doi.org/10.1177/0962280216680239.

Liang, Hua, and Guohua Zou. 2008. "Improved AIC Selection Strategy for Survival Analysis." *Computational Statistics & Data Analysis* 52 (5): 2538–48. https://doi.org/10.1016/j.csda.2007.09.003.

Liestol, Knut, Per Kragh Andersen, and Ulrich Andersen. 1994. "Survival analysis and neural nets." *Statistics in Medicine* 13 (12): 1189–1200. https://doi.org/10.1002/sim.4780131202.

Lundberg, Scott M, and Su-In Lee. 2017. "A Unified Approach to Interpreting Model Predictions." *Advances in Neural Information Processing Systems* 30.

Lundin, M, J Lundin, H B Burke, S Toikkanen, L Pylkkänen, and H Joensuu. 1999. "Artificial Neural Networks Applied to Survival Prediction in Breast Cancer." *Oncology* 57 (4): 281–86. https://doi.org/10.1159/000012061.

Luxhoj, James T., and Huan Jyh Shyur. 1997. "Comparison of proportional hazards models and neural networks for reliability estimation." *Journal of Intelligent Manufacturing* 8 (3): 227–34. https://doi.org/10.1023/A:1018525308809.

Ma, Shuangge, and Jian Huang. 2006. "Regularized ROC method for disease classification and biomarker selection with microarray data." *Bioinformatics (Oxford, England)* 21 (January): 4356–62. https://doi.org/10.1093/bioinformatics/bti724.

Mani, D R, James Drew, Andrew Betz, and Piew Datta. 1999. "Statistics and data mining techniques for lifetime value modeling." In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 94–103.

Mariani, L, D Coradini, E Biganzoli, P Boracchi, E Marubini, S Pilotti, B Salvadori, et al. 1997. "Prognostic factors for metachronous contralateral breast cancer: A comparison of the linear Cox regression model and its artificial neural network extension." *Breast Cancer Research and Treatment* 44 (2): 167–78. https://doi.org/10.1023/A:1005765403093.

Mayr, Andreas, Benjamin Hofner, and Matthias Schmid. 2016. "Boosting the discriminatory power of sparse survival models via optimization of the concordance index and stability selection." *BMC Bioinformatics* 17 (1): 288. https://doi.org/10.1186/s12859-016-1149-8.

Mayr, Andreas, and Matthias Schmid. 2014. "Boosting the concordance index for survival data–a unified framework to derive and evaluate biomarker combinations." *PloS One* 9 (1): e84483–83. https://doi.org/10.1371/journal.pone.0084483.

McGough, Sarah F., Devin Incerti, Svetlana Lyalina, Ryan Copping, Balasubramanian Narasimhan, and Robert Tibshirani. 2021. "Penalized Regression for Left-Truncated and Right-Censored Survival Data." *Statistics in Medicine* 40 (25): 5487–5500. https://doi.org/https://doi.org/10.1002/sim.9136.

McKinney, Scott Mayer, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, et al. 2020. "International evaluation of an AI system for breast cancer screening." *Nature* 577 (7788): 89–94. https://doi.org/10.1038/s41586-019-1799-6.

Meinshausen, Nicolai, and Peter Bühlmann. 2010. "Stability selection." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72 (4): 417–73. https://doi.org/10.1111/j.1467-9868.2010.00740.x.

Moghimi-dehkordi, Bijan, Azadeh Safaee, Mohamad Amin Pourhoseingholi, Reza Fatemi, Ziaoddin Tabeie, and Mohammad Reza Zali. 2008. "Statistical Comparison of Survival Models for Analysis of Cancer Data." *Asian Pacific Journal of Cancer Prevention* 9: 417–20.

Molnar, Christoph. 2019. *Interpretable Machine Learning*. https://christophm.github.io/interpretable-ml-book/.

Monterrubio-Gómez, Karla, Nathan Constantine-Cooke, and Catalina A. Vallejos. 2024. "A Review on Statistical and Machine Learning Competing Risks Methods." *Biometrical Journal* 66 (2): 2300060. https://doi.org/https://doi.org/10.1002/bimj.202300060.

Morales-Hernández, Alejandro, Inneke Van Nieuwenhuyse, and Sebastian Rojas Gonzalez. 2023. "A survey on multi-objective hyperparameter optimization algorithms for machine learning." *Artificial Intelligence Review* 56 (8): 8043–93. https://doi.org/10.1007/s10462-022-10359-2.

Murphy, Allan H. 1973. "A New Vector Partition of the Probability Score." *Journal of Applied Meteorology and Climatology* 12 (4): 595–600. https://doi.org/10.1175/1520-0450(1973)012%3C0595:ANVPOT%3E2.0.CO;2.

N. Venables, W, and B D. Ripley. 2002. *Modern Applied Statistics with S*. Springer. http://www.stats.ox.ac.uk/pub/MASS4.

Nadeau, Claude, and Yoshua Bengio. 2003. "Inference for the Generalization Error." *Machine Learning* 52 (3): 239–81. https://doi.org/10.1023/A:1024068626366.

Nair, Vinod, and Geoffrey E Hinton. 2010. "Rectified linear units improve restricted boltzmann machines." In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–14.

Nelson, Wayne. 1972. "Theory and Applications of Hazard Plotting for Censored Failure Data." *Technometrics* 14 (4): 945–66.

Ng, Ryan, Kathy Kornas, Rinku Sutradhar, Walter P. Wodchis, and Laura C. Rosella. 2018. "The current application of the Royston-Parmar model for prognostic modeling in health research: a scoping review." *Diagnostic and Prognostic Research* 2 (1): 4. https://doi.org/10.1186/s41512-018-0026-5.

Oh, Sung Eun, Sung Wook Seo, Min-Gew Choi, Tae Sung Sohn, Jae Moon Bae, and Sung Kim. 2018. "Prediction of Overall Survival and Novel Classification of Patients with Gastric Cancer Using the Survival Recurrent Network." *Annals of Surgical Oncology* 25 (5): 1153–59. https://doi.org/10.1245/s10434-018-6343-7.

Ohno-Machado, Lucila. 1996. "Medical applications of artificial neural networks: connectionist models of survival." Stanford University Stanford, Calif.

———. 1997. "A COMPARISON OF COX PROPORTIONAL HAZARDS AND ARTIFICIAL NEURAL NETWORK MODELS FOR MEDICAL PROGNOSIS The theoretical advantages and disadvantages of using different methods for predicting survival have seldom been tested in real data sets [1 , 2]. Althou." *Comput. Biol. Med* 27 (1): 55–65.

Patel, Katie, Richard Kay, and Lucy Rowell. 2006. "Comparing proportional hazards and accelerated failure time models: An application in influenza." *Pharmaceutical Statistics* 5 (3): 213–24. https://doi.org/10.1002/pst.213.

Pölsterl, Sebastian. 2020. "scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn." *Journal of Machine Learning Research* 21 (212): 1–6. http://jmlr.org/papers/v21/20-729.html.

Probst, Philipp, Anne-Laure Boulesteix, and Bernd Bischl. 2019. "Tunability: Importance of Hyperparameters of Machine Learning Algorithms." *Journal of Machine Learning Research* 20 (53): 1–32. http://jmlr.org/papers/v20/18-444.html.

Puddu, Paolo Emilio, and Alessandro Menotti. 2012. "Artificial neural networks versus proportional hazards Cox models to predict 45-year all-cause mortality in the Italian Rural Areas of the Seven Countries Study." *BMC Medical Research Methodology* 12 (1): 100. https://doi.org/10.1186/1471-2288-12-100.

Qi, Jiezhi. 2009. "Comparison of Proportional Hazards and Accelerated Failure Time Models." PhD thesis.

Qi, Shi-Ang, Neeraj Kumar, Mahtab Farrokh, Weijie Sun, Li-Hao Kuan, Rajesh Ranganath, Ricardo Henao, and Russell Greiner. 2023. "An Effective Meaningful Way to Evaluate Survival Models." In *Proceedings of the 40th International Conference on Machine Learning*, edited by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, 202:28244–76. Proceedings of Machine Learning Research. PMLR. https://proceedings.mlr.press/v202/qi23b.html.

R., Cox, and Snell J. 1968. "A General Definition of Residuals." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 30 (2): 248–75.

Rahman, M. Shafiqur, Gareth Ambler, Babak Choodari-Oskooei, and Rumana Z. Omar. 2017. "Review and evaluation of performance measures for survival prediction models in external validation settings." *BMC Medical Research Methodology* 17 (1): 1–15. https://doi.org/10.1186/s12874-017-0336-2.

Reid, Nancy. 1994. "A Conversation with Sir David Cox." *Statistical Science* 9 (3): 439–55. https://doi.org/10.1214/aos/1176348654.

Ridgeway, Greg. 1999. "The state of boosting." *Computing Science and Statistics* 31: 172–81.

Rietschel, Carl, Jinsung Yoon, and Mihaela van der Schaar. 2018. "Feature Selection for Survival Analysis with Competing Risks using Deep Learning." *arXiv Preprint arXiv:1811.09317*.

Rindt, David, Robert Hu, David Steinsaltz, and Dino Sejdinovic. 2022. "Survival Regression with Proper Scoring Rules and Monotonic Neural Networks," March. http://arxiv.org/abs/2103.14755.

Ripley, Brian D, and Ruth M Ripley. 2001. "Neural networks as statistical methods in survival analysis." In *Clinical Applications of Artificial Neural Networks*, edited by Richard Dybowski and Vanya Gant, 237–55. Cambridge: Cambridge University Press. https://doi.org/DOI: 10.1017/CBO9780511543494.011.

Ripley, R M, A L Harris, and L Tarassenko. 1998. "Neural network models for breast cancer prognosis." *Neural Computing & Applications* 7 (4): 367–75. https://doi.org/10.1007/BF01428127.

Royston, P. 2001. "The Lognormal Distribution as a Model for Survival Time in Cancer, With an Emphasis on Prognostic Factors." *Statistica Neerlandica* 55 (1): 89–104. https://doi.org/10.1111/1467-9574.00158.

Royston, Patrick, and Douglas G. Altman. 2013. "External validation of a Cox prognostic model: Principles and methods." *BMC Medical Research Methodology* 13 (1). https://doi.org/10.1186/1471-2288-13-33.

Royston, Patrick, Mahesh K B Parmar, and Douglas G Altman. 2008. "Visualizing Length of Survival in Time-to-Event Studies: A Complement to Kaplan–Meier Plots." *JNCI: Journal of the National Cancer Institute* 100 (2): 92–97. https://doi.org/10.1093/jnci/djm265.

Royston, Patrick, and Mahesh K. B. Parmar. 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): 2175–97. https://doi.org/10.1002/sim.1203.

Royston, Patrick, and Willi Sauerbrei. 2004. "A new measure of prognostic separation in survival data." *Statistics in Medicine* 23 (5): 723–48. https://doi.org/10.1002/sim.1621.

Sashegyi, Andreas, and David Ferry. 2017. "On the Interpretation of the Hazard Ratio and Communication of Survival Benefit." *The Oncologist* 22 (4): 484–86. https://doi.org/10.1634/theoncologist.2016-0198.

Schemper, Michael, and Robin Henderson. 2000. "Predictive Accuracy and Explained Variation in Cox Regression." *Biometrics* 56: 249–55. https://doi.org/10.1002/sim.1486.

Schemper, Michael, Samo Wakounig, and Georg Heinze. 2009. "The estimation of average hazard ratios by weighted Cox regression." *Statistics in Medicine* 28 (19): 2473–89. https://doi.org/10.1002/sim.3623.

Schmid, Matthias, Thomas Hielscher, Thomas Augustin, and Olaf Gefeller. 2011. "A Robust Alternative to the Schemper-Henderson Estimator of Prediction Error." *Biometrics* 67 (2): 524–35. https://doi.org/10.1111/j.1541-0420.2010.01459.x.

Schmid, Matthias, and Torsten Hothorn. 2008a. "Boosting additive models using component-wise P-splines." *Computational Statistics & Data Analysis* 53 (2): 298–311.

———. 2008b. "Flexible boosting of accelerated failure time models." *BMC Bioinformatics*

9 (February): 269. https://doi.org/10.1186/1471-2105-9-269.

Schmid, Matthias, and Sergej Potapov. 2012. "A comparison of estimators to evaluate the discriminatory power of time-to-event models." *Statistics in Medicine* 31 (23): 2588–2609. https://doi.org/10.1002/sim.5464.

Schmid, Matthias, Marvin Wright, and Andreas Ziegler. 2016. "On the Use of Harrell's c for Clinical Risk Prediction via Random Survival Forests." *Expert Systems with Applications* 63 (July). https://doi.org/10.1016/j.eswa.2016.07.018.

Schoop, Rotraut, Martin Schumacher, and Erika Graf. 2011. "Measures of prediction error for survival data with longitudinal covariates." *Biometrical Journal* 53 (2): 275–93. https://doi.org/10.1002/bimj.201000145.

Schwarzer, Guido, Werner Vach, and Martin Schumacher. 2000. "On the misuses of artificial neural networks for prognostic and diagnostic classification in oncology." *Statistics in Medicine* 19 (4): 541–61. https://doi.org/10.1002/(SICI)1097-0258(20000229)19:4%3C541::AID-SIM355%3E3.0.CO;2-V.

Segal, Mark Robert. 1988. "Regression Trees for Censored Data." *Biometrics* 44 (1): 35–47.

Seker, H., M. O. Odetayo, D. Petrovic, R. N. G. Naguib, C. Bartoli, L. Alasio, M. S. Lakshmi, G. V. Sherbet, and O. R. Hinton. 2002. "An Artificial Neural Network Based Feature Evaluation Index for the Assessment of Clinical Factors in Breast Cancer Survival Analysis." In *IEEE CCECE2002. Canadian Conference on Electrical and Computer Engineering. Conference Proceedings (Cat. No.02CH37373)*, 2:1211–1215 vol.2. https://doi.org/10.1109/CCECE.2002.1013121.

Seker, Huseyin, Michael O Odetayo, Dobrila Petrovic, Raouf N G Naguib, C Bartoli, L Alasio, M S Lakshmi, and G V Sherbet. 2002. "Assessment of nodal involvement and survival analysis in breast cancer patients using image cytometric data: statistical, neural network and fuzzy approaches." *Anticancer Research* 22 (1A): 433–38. http://europepmc.org/abstract/MED/12017328.

Shivaswamy, Pannagadatta K., Wei Chu, and Martin Jansche. 2007. "A support vector approach to censored targets." In *Proceedings - IEEE International Conference on Data Mining, ICDM*, 655–60. https://doi.org/10.1109/ICDM.2007.93.

Simon, Noah, Jerome H. Friedman, Trevor Hastie, and Rob Tibshirani. 2011. "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." *Journal of Statistical Software* 39 (5): 1–13. https://doi.org/10.18637/jss.v039.i05.

Simon, Richard. 2007. "Resampling Strategies for Model Assessment and Selection." In *Fundamentals of Data Mining in Genomics and Proteomics*, edited by Werner Dubitzky, Martin Granzow, and Daniel Berrar, 173–86. Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-47509-7_8.

Sonabend, Raphael. 2020. "survivalmodels: Models for Survival Analysis." CRAN. https://raphaels1.r-universe.dev/ui#package:survivalmodels.

Sonabend, Raphael Edward Benjamin. 2021. "A Theoretical and Methodological Framework for Machine Learning in Survival Analysis: Enabling Transparent and Accessible Predictive Modelling on Right-Censored Time-to-Event Data." PhD, University College London (UCL). https://discovery.ucl.ac.uk/id/eprint/10129352/.

Sonabend, Raphael, Andreas Bender, and Sebastian Vollmer. 2022. "Avoiding C-hacking when evaluating survival distribution predictions with discrimination measures." Edited by Zhiyong Lu. *Bioinformatics* 38 (17): 4178–84. https://doi.org/10.1093/bioinformatics/btac451.

Sonabend, Raphael, Franz J Király, Andreas Bender, Bernd Bischl, and Michel Lang. 2021. "mlr3proba: an R package for machine learning in survival analysis." Edited by Jonathan Wren. *Bioinformatics* 37 (17): 2789–91. https://doi.org/10.1093/bioinformatics/btab039.

Sonabend, Raphael, Florian Pfisterer, Alan Mishler, Moritz Schauer, Lukas Burk, Sumantrak Mukherjee, and Sebastian Vollmer. 2022. "Flexible Group Fairness Metrics for Sur-

vival Analysis." In *DSHealth 2022 Workshop on Applied Data Science for Healthcare at KDD2022*. http://arxiv.org/abs/2206.03256.

Sonabend, Raphael, Lilith K Whittles, Natsuko Imai, Pablo N Perez-Guzman, Edward S Knock, Thomas Rawson, Katy A M Gaythorpe, et al. 2021. "Non-pharmaceutical interventions, vaccination, and the SARS-CoV-2 delta variant in England: a mathematical modelling study." *The Lancet*. https://doi.org/https://doi.org/10.1016/S0140-6736(21)02276-5.

Sonabend, Raphael, John Zobolas, Riccardo Be Bin, Philipp Kopper, Lukas Burk, and Andreas Bender. 2025. "Examining Marginal Properness in the External Validation of Survival Models with Squared and Logarithmic Losses." https://arxiv.org/abs/2212.05260.

Song, Xiao, and Xiao-Hua Zhou. 2008. "A semiparametric approach for the covariate specific ROC curve with survival outcome." *Statistica Sinica* 18 (July): 947–65.

Spooner, Annette, Emily Chen, Arcot Sowmya, Perminder Sachdev, Nicole A Kochan, Julian Trollor, and Henry Brodaty. 2020. "A comparison of machine learning methods for survival analysis of high-dimensional clinical data for dementia prediction." *Scientific Reports* 10 (1): 20410. https://doi.org/10.1038/s41598-020-77220-w.

Spruance, Spotswood L, Julia E Reid, Michael Grace, and Matthew Samore. 2004. "Hazard ratio in clinical trials." *Antimicrobial Agents and Chemotherapy* 48 (8): 2787–92. https://doi.org/10.1128/AAC.48.8.2787-2792.2004.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. "Dropout: a simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research* 15 (1): 1929–58.

Stasinopoulos, Mikis, Bob Rigby, Vlasios Voudouris, and Daniil Kiose. 2020. "gamlss.add: Extra Additive Terms for Generalized Additive Models for Location Scale and Shape." CRAN. https://cran.r-project.org/package=gamlss.add.

Street, W Nick. 1998. "A Neural Network Model for Prognostic Prediction." In *Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco.

Suresh, Krithika, Cameron Severn, and Debashis Ghosh. 2022. "Survival prediction models: an introduction to discrete-time modeling." *BMC Medical Research Methodology* 22 (1): 207. https://doi.org/10.1186/s12874-022-01679-6.

Therneau, Terry M. 2015. "A Package for Survival Analysis in S." https://cran.r-project.org/package=survival.

Therneau, Terry M., and Elizabeth Atkinson. 2024. "Concordance." https://cran.r-project.org/web/packages/survival/vignettes/concordance.pdf.

Therneau, Terry M., Patricia M. Grambsch, and Thomas R. Fleming. 1990. "Martingale-based residuals for survival models." *Biometrika* 77 (1): 147–60. https://doi.org/10.1093/biomet/77.1.147.

Tibshirani, Robert. 1997. "The Lasso Method for Variable Selection in the Cox Model." *Statistics in Medicine* 16 (4): 385–95. https://doi.org/https://doi.org/10.1002/(SICI)1097-0258(19970228)16:4%3C385::AID-SIM380%3E3.0.CO;2-3.

Tsoumakas, Grigorios, and Ioannis Katakis. 2007. "Multi-Label Classification: An Overview." *International Journal of Data Warehousing and Mining* 3 (3): 1–13. https://doi.org/10.4018/jdwm.2007070101.

Tsouprou, Sofia. 2015. "Measures of Discrimination and Predictive Accuracy for Interval Censored Survival Data." Master's thesis.

Turnbull, Bruce W. 1974. "Nonparametric Estimation of a Survivorship Function with Doubly Censored Data." *Journal of the American Statistical Association* 69 (345): 169–73. https://doi.org/10.1080/01621459.1974.10480146.

Tutz, Gerhard, and Harald Binder. 2007. "Boosting Ridge Regression." *Computational Statistics & Data Analysis* 51 (February): 6044–59. https://doi.org/10.1016/j.csda.2006.

11.041.

Tutz, Gerhard, and Matthias Schmid. 2016. *Modeling Discrete Time-to-Event Data*. Springer Series in Statistics. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-28158-2.

Uno, Hajime, Tianxi Cai, Michael J. Pencina, Ralph B. D'Agostino, and L J Wei. 2011. "On the C-statistics for Evaluating Overall Adequacy of Risk Prediction Procedures with Censored Survival Data." *Statistics in Medicine* 30 (10): 1105–17. https://doi.org/10.1002/sim.4154.

Uno, Hajime, Tianxi Cai, Lu Tian, and L J Wei. 2007. "Evaluating Prediction Rules for t-Year Survivors with Censored Regression Models." *Journal of the American Statistical Association* 102 (478): 527–37. http://www.jstor.org/stable/27639883.

Ushey, Kevin, J J Allaire, and Yuan Tang. 2020. "reticulate: Interface to 'Python'." CRAN. https://cran.r-project.org/package=reticulate.

Vakulenko-Lagun, Bella, Micha Mandel, and Rebecca A. Betensky. 2020. "Inverse Probability Weighting Methods for Cox Regression with Right-Truncated Data." *Biometrics* 76 (2): 484–95. https://doi.org/10.1111/biom.13162.

Van Belle, Vanya, Kristiaan Pelckmans, Johan A K Suykens, and Sabine Van Huffel. 2008. "Survival SVM: a practical scalable algorithm." In *Proceedings of the 16th European Symposium on Artificial Neural Networks (ESANN)*, 89–94.

Van Belle, Vanya, Kristiaan Pelckmans, Johan A. K. Suykens, and Sabine Van Huffel. 2007. "Support Vector Machines for Survival Analysis." In *In Proceedings of the Third International Conference on Computational Intelligence in Medicine and Healthcare*. 1.

Van Belle, Vanya, Kristiaan Pelckmans, Sabine Van Huffel, and Johan A. K. Suykens. 2011. "Support vector methods for survival analysis: A comparison between ranking and regression approaches." *Artificial Intelligence in Medicine* 53 (2): 107–18. https://doi.org/10.1016/j.artmed.2011.06.006.

Van Houwelingen, Hans C. 2000. "Validation, calibration, revision and combination of prognostic survival models." *Statistics in Medicine* 19 (24): 3401–15. https://doi.org/10.1002/1097-0258(20001230)19:24%3C3401::AID-SIM554%3E3.0.CO;2-2.

———. 2007. "Dynamic prediction by landmarking in event history analysis." *Scandinavian Journal of Statistics* 34 (1): 70–85. https://doi.org/10.1111/j.1467-9469.2006.00529.x.

Vinzamuri, Bhanukiran, Yan Li, and Chandan K. Reddy. 2017. "Pre-processing censored survival data using inverse covariance matrix based calibration." *IEEE Transactions on Knowledge and Data Engineering* 29 (10): 2111–24. https://doi.org/10.1109/TKDE.2017.2719028.

Vock, David M, Julian Wolfson, Sunayan Bandyopadhyay, Gediminas Adomavicius, Paul E Johnson, Gabriela Vazquez-Benitez, and Patrick J O'Connor. 2016. "Adapting machine learning techniques to censored time-to-event health record data: A general-purpose approach using inverse probability of censoring weighting." *Journal of Biomedical Informatics* 61: 119–31. https://doi.org/https://doi.org/10.1016/j.jbi.2016.03.009.

Volinsky, Chris T, and Adrian E Raftery. 2000. "Bayesian Information Criterion for Censored Survival Models." *International Biometric Society* 56 (1): 256–62.

Wang, Ping, Yan Li, and Chandan K. Reddy. 2019. "Machine Learning for Survival Analysis." *ACM Computing Surveys* 51 (6): 1–36. https://doi.org/10.1145/3214306.

Wang, Zhu, and C Y Wang. 2010. "Buckley-James Boosting for Survival Analysis with High-Dimensional Biomarker Data." *Statistical Applications in Genetics and Molecular Biology* 9 (1). https://doi.org/https://doi.org/10.2202/1544-6115.1550.

Wei, L J. 1992. "The Accelerated Failure Time Model: A Useful Alternative to the Cox Regression Model in Survival Analysis." *Statistics in Medicine* 11: 1871–79.

Welchowski, Thomas, and Matthias Schmid. 2019. "discSurv: Discrete Time Survival Analysis." CRAN. https://cran.r-project.org/package=discSurv.

Wickham, Hadley. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wiegrebe, Simon, Philipp Kopper, Raphael Sonabend, Bernd Bischl, and Andreas Bender. 2024. "Deep learning for survival analysis: a review." *Artificial Intelligence Review* 57 (3): 65. https://doi.org/10.1007/s10462-023-10681-3.

Wilks, Daniel S. 1990. "On the Combination of Forecast Probabilities for Consecutive Precipitation Periods." *Weather and Forecasting* 5 (4): 640–50. https://doi.org/10.1175/1520-0434(1990)005%3C0640:OTCOFP%3E2.0.CO;2.

Wright, Marvin N., and Andreas Ziegler. 2017. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software* 77 (1): 1–17.

Wu, Ying, and Richard J Cook. 2020. "Assessing the Accuracy of Predictive Models with Interval-Censored Data." *Biostatistics* 23 (1): 18–33. https://doi.org/10.1093/biostatistics/kxaa011.

Wu, Yuan, Xiaofei Wang, Jiaxing Lin, Beilin Jia, and Kouros Owzar. 2020. "Predictive Accuracy of Markers or Risk Scores for Interval Censored Survival Data." *Statistics in Medicine* 39 (18): 2437–46. https://doi.org/https://doi.org/10.1002/sim.8547.

Xiang, Anny, Pablo Lapuerta, Alex Ryutov, Jonathan Buckley, and Stanley Azen. 2000. "Comparison of the performance of neural network methods and Cox regression for censored survival data." *Computational Statistics & Data Analysis* 34 (2): 243–57. https://doi.org/https://doi.org/10.1016/S0167-9473(99)00098-5.

Yanagisawa, Hiroki. 2023. "Proper Scoring Rules for Survival Analysis." In *Proceedings of the 40th International Conference on Machine Learning*, edited by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, 202:39165–82. Proceedings of Machine Learning Research. PMLR. https://proceedings.mlr.press/v202/yanagisawa23a.html.

Yang, Yanying. 2010. "Neural Network Survival Analysis." PhD thesis, Universiteit Gent.

Yasodhara, Angeline, Mamatha Bhat, and Anna Goldenberg. 2018. *Prediction of New Onset Diabetes after Liver Transplant*.

Zare, Ali, Mostafa Hosseini, Mahmood Mahmoodi, Kazem Mohammad, Hojjat Zeraati, and Kourosh Holakouie Naieni. 2015. "A Comparison between Accelerated Failure-time and Cox Proportional Hazard Models in Analyzing the Survival of Gastric Cancer Patients." *Iranian Journal of Public Health* 44 (8): 1095–1102. https://doi.org/10.1007/s00606-006-0435-8.

Zhang, Yucheng, Edrise M Lobo-Mueller, Paul Karanicolas, Steven Gallinger, Masoom A Haider, and Farzad Khalvati. 2020. "CNN-based survival model for pancreatic ductal adenocarcinoma in medical imaging." *BMC Medical Imaging* 20 (1): 11. https://doi.org/10.1186/s12880-020-0418-1.

Zhang, Yunwei, Germaine Wong, Graham Mann, Samuel Muller, and Jean Y H Yang. 2021. "SurvBenchmark: comprehensive benchmarking study of survival analysis methods using both omics data and clinical data." *bioRxiv*, January, 2021.07.11.451967. https://doi.org/10.1101/2021.07.11.451967.

Zhang, Zhigang, and Jianguo Sun. 2010. "Interval Censoring." *Statistical Methods in Medical Research* 19 (1): 53–70. https://doi.org/10.1177/0962280209105023.

Zhao, Lili, and Dai Feng. 2020. "Deep Neural Networks for Survival Analysis Using Pseudo Values." *IEEE Journal of Biomedical and Health Informatics* 24 (11): 3308–14. https://doi.org/10.1109/JBHI.2020.2980204.

Zhou, Zheng, Elham Rahme, Michal Abrahamowicz, and Louise Pilote. 2005. "Survival Bias Associated with Time-to-Treatment Initiation in Drug Effectiveness Evaluation: A Comparison of Methods." *American Journal of Epidemiology* 162 (10): 1016–23. https://doi.org/10.1093/aje/kwi307.

Zhu, Wan, Longxiang Xie, Jianye Han, and Xiangqian Guo. 2020. "The Application of Deep Learning in Cancer Prognosis Prediction." *Cancers* 12 (3): 603. https://doi.org/10.3390/cancers12030603.

Zhu, X, J Yao, and J Huang. 2016. "Deep convolutional neural network for survival analysis with pathological images." In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, 544–47. https://doi.org/10.1109/BIBM.2016.7822579.

Zou, Hui, and Trevor Hastie. 2005. "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67 (2): 301–20. https://doi.org/10.1111/j.1467-9868.2005.00503.x.