Introdução ao Aprendizado de Máquina para Análise de Sobrevivência

VII Workshop em Análise de Sobrevivência e Aplicações, Brasília, DF, 23 a 25 de outubro de 2024

AUTORES
Anderson Ara
Marcelo Ferreira
Agatha Rodrigues
Raydonal Ospina

AFILIAÇÕES
Departamento de Estatística, UFPR
Departamento de Estatística, UFPB
Departamento de Estatística, UFES
Departamento de Estatística, UFBA

RESUMO
Devido aos recentes avanços em aprendizado de máquina (ML), têm surgido novas oportunidades para estimar a estrutura de associação entre tempos de sobrevivência e covariáveis por meio de técnicas mais flexíveis. Este tutorial aborda os conceitos de ML para análise de sobrevivência, com foco específico nos métodos *Random Survival Forest* (RSF) e *Support Vector Censored Regression* (SVCR). Além de explorar os fundamentos dessas abordagens, o tutorial fornece exemplos práticos utilizando a linguagem `R`, uma ferramenta amplamente adotada na comunidade estatística e de ciência de dados. Ao final, espera-se que o leitor esteja apto a aplicar essas técnicas em seus próprios conjuntos de dados, ciente das vantagens e limitações associadas.

1 Introdução

A análise de sobrevivência é uma área da estatística voltada para o estudo do tempo até a ocorrência de um evento de interesse, comumente chamado de "tempo de falha" ou "tempo de sobrevivência". Inicialmente, foi desenvolvida para analisar a sobrevida (probabilidade de o evento não ocorrer até certo momento) e identificar os fatores que influenciam o risco de óbito. Com o tempo, essa abordagem foi amplamente adotada em diversas áreas além da medicina. Na engenharia, é conhecida como "análise de confiabilidade", avaliando o risco de falha de componentes mecânicos e elétricos (Yang et al. 2022). Em sociologia, é utilizada para prever taxas de desemprego, retenção acadêmica e outras questões sociais (Box-Steffensmeier et al. 2015). Por exemplo, na criminologia, estuda-se o tempo entre a libertação de presos e a reincidência em crimes, analisando os riscos associados (Benda 2003).

Uma característica distintiva dos dados de sobrevivência é a presença de **censura**, que ocorre quando o tempo até o evento de interesse não é totalmente observado para todos os indivíduos. Isso pode acontecer porque o estudo termina antes que o evento ocorra para alguns participantes, ou porque eles são perdidos durante o acompanhamento. Assim, a análise de sobrevivência deve lidar com a informação parcial disponível, contrastando com problemas clássicos de regressão onde a variável resposta é observada integralmente (Collett 2023; Klein e Moeschberger 2006; Cavalcante et al. 2023).

Devido aos recentes avanços em aprendizado de máquina (ML), surgiram novas oportunidades para estimar a estrutura de associação entre os tempos de sobrevivência e covariáveis por meio de técnicas mais flexíveis. Métodos tradicionais, como o modelo de Cox (Cox 1972), embora poderosos, impõem suposições específicas sobre os dados, como a proporcionalidade dos riscos, forçando uma ligação

específica entre as covariáveis e a resposta. Embora as interações entre as covariáveis possam ser incorporadas, elas devem ser especificadas pelo analista (Bou-Hamad, Larocque, e Ben-Ameur 2011). Quando o analista não deseja impor tal ligação desde o início, abordagens mais flexíveis de ML estão disponíveis. Tais abordagens oferecem diversas ferramentas para capturar relações complexas e não lineares sem depender de fortes premissas paramétricas.

Este tutorial aborda, de forma introdutória, os conceitos teóricos de métodos de aprendizado de máquina para análise de sobrevivência, com foco em **Random Survival Forests** (RSF) e **Support Vector Censored Regression** (SVCR). O RSF é uma extensão das florestas aleatórias para dados censurados, capaz de modelar interações complexas entre covariáveis e capturar estruturas não lineares (Hemant Ishwaran et al. 2008). O SVCR adapta as máquinas de vetor de suporte para lidar com censura, permitindo a modelagem eficiente de dados de sobrevivência em altas dimensões (Van Belle et al. 2011).

No entanto, a aplicação de técnicas de ML na análise de sobrevivência apresenta desafios únicos:

1. **Censura dos Dados**: A maioria dos algoritmos de ML convencionais não é projetada para lidar com censura, exigindo adaptações ou extensões específicas (P. Wang, Li, e Reddy 2019).

2. **Dados de Alta Dimensão e Pequenas Amostras**: Em muitos estudos clínicos, o número de covariáveis pode ser grande em comparação com o número de eventos observados, levando a problemas de superajuste (Ambrogi e Scheike 2016).

3. **Interpretação dos Modelos**: Enquanto modelos paramétricos tradicionais oferecem interpretações claras dos efeitos das covariáveis, modelos de ML podem ser vistos como "caixas-pretas", dificultando a interpretação clínica (Langbein et al. 2024; Rahman e Purushotham 2022).

4. **Validação e Generalização**: Garantir que os modelos generalizem bem para novos dados é crucial, especialmente em contextos médicos onde decisões críticas são tomadas com base nas previsões (Cabitza et al. 2021; Pfisterer et al. 2022; Guan et al. 2021).

Apesar desses desafios, a integração de ML na análise de sobrevivência tem o potencial de melhorar significativamente a precisão das previsões e a compreensão dos fatores que influenciam o tempo até a ocorrência do evento de interesse. Por exemplo, a identificação precoce de pacientes com alto risco de complicações pode levar a intervenções preventivas e melhorar os resultados clínicos.

2 Aprendizado Estatístico de Máquina

Métodos de **Aprendizado Estatístico de Máquina (AEM)** ou *Statistical Machine Learning (SML)* referem-se a um conjunto de algoritmos e modelos que, uma vez implementados em computadores, podem descobrir características importantes dos dados, servindo de base para a tomada de decisão, tais métodos são flexíveis e adaptáveis a diversos tipos de conjuntos de dados (Hastie, Tibshirani, e Friedman 2001; Bishop 2007; Sambasivan, Das, e Sahu 2020).

Aplicações de aprendizado de máquina estão por toda parte, sendo essenciais em diversas áreas, tais como, finanças, comércio, transporte, engenharia, comunicação, saúde, medicina, entre outras. Métodos de aprendizado de máquina têm sido aplicados com sucesso em sistemas biométricos (reconhecimento de voz, reconhecimento de faces, reconhecimento de iris, reconhecimento de assinaturas, etc.), aterrisagem automática de aeronaves, automóveis autônomos, diagnóstico médico assistido (previsão de doenças a partir de dados clínicos, resultados de exames, imagens, etc.), entre outros.

Para muitas dessas aplicações, o objetivo principal é obter uma boa estimativa do tempo em que um determinado evento de interesse ocorre. Na análise de sobrevivência, isso se traduz na modelagem do tempo até a ocorrência de eventos como falha de equipamentos, recaída de pacientes ou tempos de vida de organismos. Um dos principais desafios para esse tipo de dado é a presença de observações em que o evento de interesse não é observado devido a limitações de tempo ou falta de acompanhamento durante o período observado, resultando em dados censurados (P. Wang, Li, e Reddy 2019).

O sucesso em tantas áreas diferentes e a flexibilidade oferecida pelos métodos de AEM fazem com que surjam novas oportunidades para estimar a estrutura de associação entre os tempos de sobrevivência e covariáveis por meio de técnicas mais flexíveis. Assim, diversos métodos de aprendizado de máquina têm sido propostos para o contexto de análise de sobrevivência. Em Spooner et al. (2020), diversos métodos de AEM foram comparados na tarefa de predição da sobrevivência de pacientes com demência, enquanto que Štěpánek et al. (2020) utiliza métodos de AEM para previsão da sobrevivência de pacientes com câncer de estômago. Huang et al. (2023) apresenta uma análise de diversas aplicações de AEM para dados de sobrevivência. Em Kvamme, Borgan, e Scheel (2019), uma comparação entre redes neurais e o modelo de Cox é apresentada no contexto de predição de tempo até a ocorrência de eventos. Mais recentemente, métodos de aprendizado profundo (*deep learning*) têm sido propostos para a análise de dados de sobrevivência (Wiegrebe et al. 2023).

Esses métodos podem ser classificados de acordo com várias dimensões, incluindo se são supervisionados ou não supervisionados, e se são métodos preditivos ou descritivos. Na análise de sobrevivência, o foco geralmente está em métodos supervisionados, onde o objetivo é prever o tempo até o evento com base em covariáveis observadas.

Os **métodos supervisionados** utilizam dados de treinamento com respostas conhecidas (por exemplo, tempos de sobrevivência observados) para aprender um modelo que possa prever a resposta para novas observações. Exemplos incluem árvores de decisão, florestas aleatórias e modelos de regressão. Na análise de sobrevivência, esses métodos precisam ser adaptados para lidar com censura e outras particularidades dos dados de tempo. Já, os **métodos não supervisionados**, por outro lado, buscam encontrar estruturas ou padrões intrínsecos nos dados sem utilizar respostas conhecidas. Embora menos comuns na análise de sobrevivência, técnicas como agrupamento (clustering) podem ser usadas para identificar subgrupos de indivíduos com perfis de risco semelhantes.

A Figura 1 exibe uma visão geral sobre os principais métodos já propostos na literatura para análise de sobrevivência.

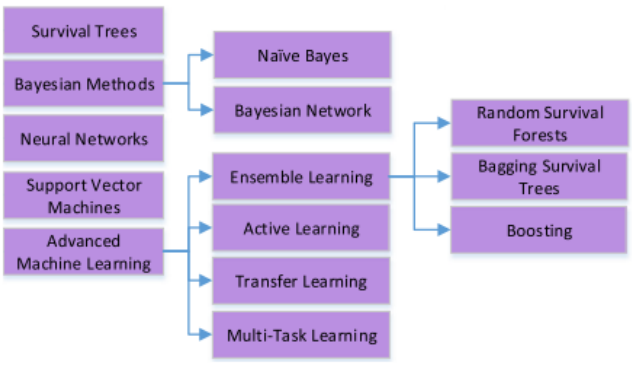


Fig 1. Visão geral dos principais métodos de aprendizado estatístico de máquina para análise de sobrevivência. Adaptado de Wang et al., 2019.

Entre os métodos supervisionados para análise de sobrevivência, destacam-se:

- **Árvores de Sobrevivência**: Extensão das árvores de decisão clássicas para lidar com dados censurados. Elas particionam o espaço de covariáveis em regiões homogêneas em termos de risco ou tempo de sobrevivência (Gordon e Olshen 1985). As árvores de sobrevivência são métodos preditivos e supervisionados, pois aprendem a partir de dados rotulados para prever o risco ou a sobrevida em novas observações.

- **Florestas Aleatórias de Sobrevivência**: Conjunto de árvores de sobrevivência construídas a partir de diferentes subconjuntos de dados e variáveis. As previsões são agregadas para melhorar a precisão e reduzir a variância do modelo (Hemant Ishwaran et al. 2008). Este método é altamente preditivo e supervisionado, capaz de capturar interações complexas entre covariáveis e modelar relações não lineares.

- **Regressão Censurada por Vetores de Suporte**: Adaptação das Máquinas de Vetores de Suporte (SVM) para lidar com dados censurados. O objetivo é encontrar uma função de predição que minimize um risco empírico modificado adequado para dados de sobrevivência (Shivaswamy, Chu, e Jansche 2007). Este método é preditivo e supervisionado, aproveitando o poder das SVM para lidar com alta dimensionalidade e problemas não lineares.

Esses métodos são preditivos, pois visam prever o tempo até o evento de interesse, e são supervisionados, pois dependem de dados de treinamento com tempos de sobrevivência observados (ainda que censurados). A censura nos dados adiciona complexidade à modelagem, exigindo adaptações nos algoritmos tradicionais de aprendizado de máquina.

Além disso, os métodos de aprendizado de máquina para análise de sobrevivência precisam lidar com questões como alta dimensionalidade, relações não lineares entre covariáveis e tempos de sobrevivência, e interação entre variáveis. Os métodos mencionados, como Florestas Aleatórias de Sobrevivência e Regressão Censurada por Vetores de Suporte, são capazes de capturar tais complexidades devido à sua flexibilidade e poder de modelagem.

O objetivo deste tutorial é introduzir os conceitos básicos sobre alguns desses métodos de aprendizado de máquina e ilustrar como podem ser utilizados no contexto de estimação de tempos até a ocorrência de eventos na presença de censura. A seguir, são resumidos alguns conceitos fundamentais de análise de sobrevivência, essenciais para a compreensão e implementação de técnicas de ML na área.

2.1 Métodos Baseados em Árvores e Florestas Aleatórias

Métodos baseados em árvores (L. Breiman et al. 1984) foram desenvolvidos tanto no contexto de classificação como no de regressão. A ideia básica é estratificar ou segmentar o espaço das variáveis preditoras em regiões. Isso leva a resultados que são interpretáveis. A predição para uma dada observação é feita, tipicamente, através da média (regressão) ou da moda (classificação) dos valores da variável resposta para as observações de treinamento na região a que essas observações pertencem.

Considere a tarefa de classificação e suponha que queremos dividir nosso conjunto de dados em M regiões $R_1, R_2, \ldots, R_M$. A Figura 2 exibe um exemplo de uma partição do espaço de entrada de $\mathbf{x}$_1 e $\mathbf{x}$ _2 em 5 regiões.

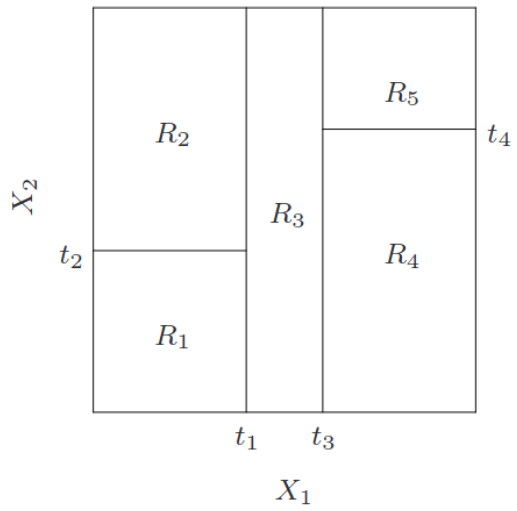


Fig 2. Exemplo de partição do espaço de entrada

Na região R_m, com N_m observações, podemos calcular a proporção da classe k como

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} \mathbb{1}(y_i = k),$$

em que

$$\mathbb{1}(y_i = k) = \begin{cases} 1, & \text{se } y_i = k, \\ 0, & \text{caso contrário.} \end{cases}$$

Dadas novas observações na região R_m, iremos classificá-las de acordo com $\hat{y}(m) = \arg\max_k \hat{p}_{mk}$. Podemos então calcular o grau de impureza de uma região R_m através da taxa de erro de classificação, definida por:

$$Q_m = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} \mathbb{1}(y_i \neq \hat{y}(m)) = 1 - \max_k \hat{p}_{mk};$$

Em geral, a taxa de erro não é sensível o bastante ao crescimento da árvore, e, na prática, duas outras medidas são utilizadas: o índice de Gini e a entropia cruzada, que medem, de modo similar, a variância total nas K classes. Essas medidas são dadas, respectivamente, por:

- Índice de Gini:

$$Q_m = \sum_k^K \hat{p}_{mk}(1 - \hat{p}_{mk});$$

- Entropia cruzada:

$$Q_m = -\sum_k^K \hat{p}_{mk} \log \hat{p}_{mk}.$$

A impureza é utilizada para decidir que variável e que ponto de corte deveremos usar no próximo nível da árvore. Começando com todas as observações e dados uma variável j e um ponto de corte s, definimos um par de regiões $R_1(j, s) = \{\mathbf{x} | x_j \leq s\}$ e $R_2(j, s) = \{\mathbf{x} | x_j > s\}$. Buscamos a variável j e o ponto de

corte s que minimizam

$$\frac{N_1 Q_1 + N_2 Q_2}{N_1 + N_2}.$$

Para encontrar j e s, podemos ordenar os valores observados de cada variável j e testá-los sequencialmente como pontos de corte. Alternativamente, uma busca aleatória pode ser considerada. Uma vez encontrada a melhor partição, fazemos o mesmo processo em cada região resultante, gerando um processo recursivo.

Note que, se não pararmos o processo de construção da árvore, ele seguirá até que todas as regiões sejam puras ou até que não haja ganho de informação, o que resultaria em uma classificação perfeita do conjunto de treinamento (super-ajuste). Podemos definir uma profundidade máxima para a árvore ou um número mínimo de observações em um nó. A fim de se evitar o super-ajuste, deve-se realizar um procedimento de poda (**pruning**). Poda, em sentido literal, consiste na remoção de certas partes de uma árvore ou planta, tais como galhos e raízes fracas, para melhorar a estrutura da árvore e promover um crescimento saudável. Com árvores de classificação e regressão isso é exatamente o que a poda promove, reduzindo o tamanho da árvore, o que pode aumentar ligeiramente o erro de treinamento, mas, ao mesmo tempo, pode reduzir significativamente o erro de teste. Ou seja, poda melhora o poder de generalização das árvores. Um dos métodos mais utilizados de poda é conhecido por poda por custo-complexidade (Mingers 1989), cujo objetivo é encontrar o equilíbrio entre complexidade e acurácia.

Para a tarefa de regressão a árvore é construída de maneira similar, ou seja, reduzindo a impureza dos nós recursivamente. O objetivo é encontrar regiões $R_1, R_2, \ldots, R_M$ que minimizem a soma dos quadrados dos resíduos, dada por

$$Q_m = \sum_{\mathbf{x}_i \in R_m} (y_i - \hat{y}_{R_m})^2,$$

em que $\hat{y}_{R_m} = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} y_i$.

A árvore é construída exatamente como fazemos para a tarefa de classificação. Ou seja, buscamos a variável j e o ponto de corte s que minimizam $\frac{N_1 Q_1 + N_2 Q_2}{N_1 + N_2}$.

A Figura 3 exibe um exemplo da superfície de previsão de uma árvore de regressão.

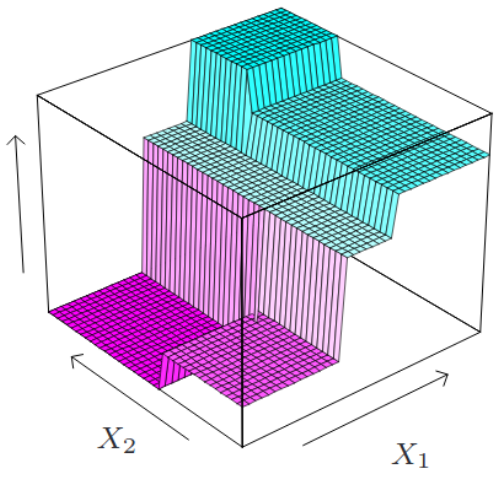


Fig 3. Exemplo de superfície de previsão

2.1.0.1 Importância das Variáveis

Devido a sua construção, modelos de árvore de decisão podem ser utilizados para avaliar a importância de cada atributo para o resultado predito.

A importância de cada atributo é calculada como a diminuição da impureza ponderada pela probabilidade de chegar ao nó (estimada por N_m/N).

Para um atributo j, sua importância $g(j)$ é a média da redução de impureza de cada nó em que j foi escolhido

$$\delta(m) = \frac{1}{N}(N_m Q_m - N_{m_1} Q_{m_1} - N_{m_2} Q_{m_2}),$$

$$g(j) = \frac{\sum_{m \in T_j} \delta(m)}{\sum_{w \in T} \delta(w)},$$

em que T e T_j indicam os conjuntos dos nós da árvore e dos nós vinculados ao atributo j, respectivamente.

2.1.1 Florestas Aleatórias

Uma característica forte das árvores de classificação e regressão é que elas são interpretáveis. Contudo, costumam apresentar baixo poder preditivo quando comparadas a outras classes de modelos.

Floresta aleatória (Leo Breiman 2001a) é um método de aprendizado de máquina da classe *ensemble learning*. A ideia é combinar múltiplas árvores (de regressão ou de classificação) construídas em versões ligeiramente modificadas de forma aleatória do conjunto de treinamento para obter melhores predições. Para a tarefa de regressão, a predição combinada será dada por uma média ponderada das predições de todas as árvores. No caso de classificação, a classe predita é determinada por votação majoritária. Em ambos os casos, é considerada a ponderação relativa a capacidade preditiva para os dados que não são selecionados pela amostra bootstrap, chamada de amostra *out-of-bag* (OBB).

A aleatoriedade é introduzida de duas formas: primeiro, cada árvore é ajustada em uma amostra *bootstrap* do conjunto de treinamento original; segundo, em cada nó de cada árvore, um subconjunto das variáveis é selecionado de forma aleatória. Uma das vantagens em se utilizar florestas aleatórias é a sua habilidade em corrigir super-ajustamento, o que ocorre frequentemente no ajuste de árvores individuais. Tipicamente, as florestas produzem melhores resultados do que árvores individuais.

A Figura 4 fornece o esquema geral do método de Floresta Aleatória.

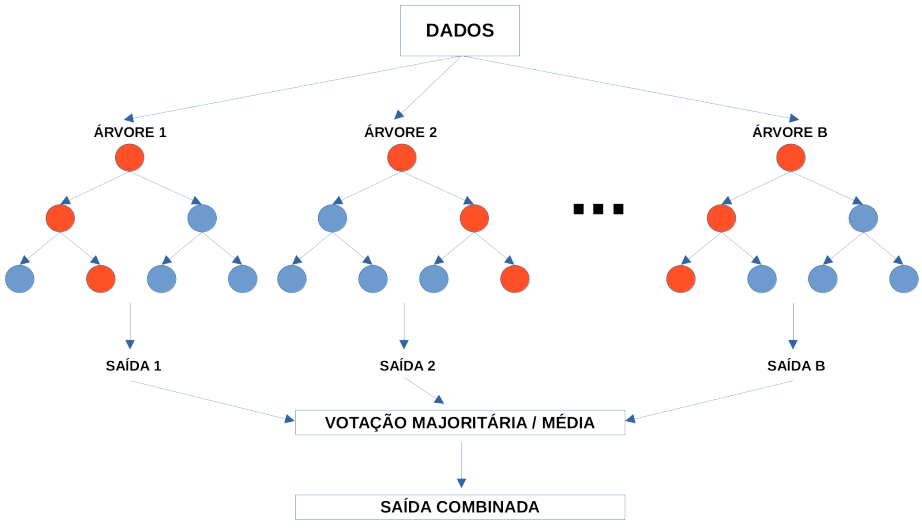


Fig 4. Ilustração de uma Floresta Aleatória

2.2 Máquinas de Vetores de Suporte

Máquinas de Vetores de Suporte (SVM - *Support Vector Machines*) foram propostas no contexto de classificação por V. Vapink (V. Vapnik 1998) e posteriormente modificadas para problemas de regressão (Smola e Schölkopf 2004).

Para o contexto de classificação binária, o SVM tenta encontrar um hiperplano ótimo, tornando-o um classificador linear binário não probabilístico. Em sua forma mais básica, SVM de margens rígidas, encontrar o hiperplano ótimo de separação é o mesmo que maximizar a distância entre as margens e separação e este problema pode ser escrito com um problema de otimização não linear como apresentado abaixo.

$$\min_{\mathbf{w},b} f(\mathbf{w}, b) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}).$$

sujeito às restrições,

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) - 1 \geq 0,$$

sendo $\mathbf{w} \in \mathbb{R}^p$ e $b \in \mathbb{R}$ os parâmetros a serem estimados, $\mathbf{x}_i \in \mathbb{R}^p$ o vetor composto por p variáveis explicativas, $y_i \in \{-1, +1\}$ o rótulo de classe, para $i = 1, \ldots, n$ observações da amostra de treinamento.

Este problema de otimização é ilustrado na Figura 5.

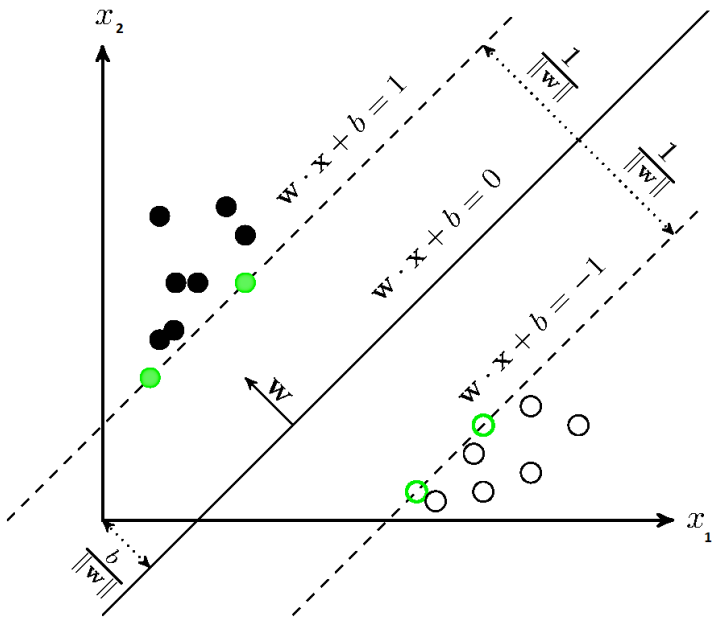


Fig 5. SVM de margens rígidas

Essa maximização pode ser feita por meio de multiplicadores de Lagrange e pode englobar separações lineares e não lineares, contemplando *kernels* polinomiais ou gaussianos.

Para o contexto de regressão, é similar ao problema de classificação, porém o interesse é focado em como os dados se comportam dentro de um hipertubo (um hipertubo é uma margem que separa duas classes de dados em um espaço multidimensional). Neste caso, as restrições serão baseadas em

$$|y_i - f(\mathbf{x}_i)| \leq \epsilon, \ \forall \ i = 1, \ldots, n$$

em que $f(\mathbf{x}_i) = \mathbf{w} \cdot \mathbf{x} + b$. Tais condições advém do uso da função perda ϵ-insensível (V. N. Vapnik 2000):

$$L_\epsilon(f, y) = \begin{cases} 0, & \text{se } |f(\mathbf{x}) - y)| < \epsilon \\ |f(\mathbf{x}) - y)| - \epsilon & \text{caso contrário} \end{cases}$$

com $\epsilon > 0$.

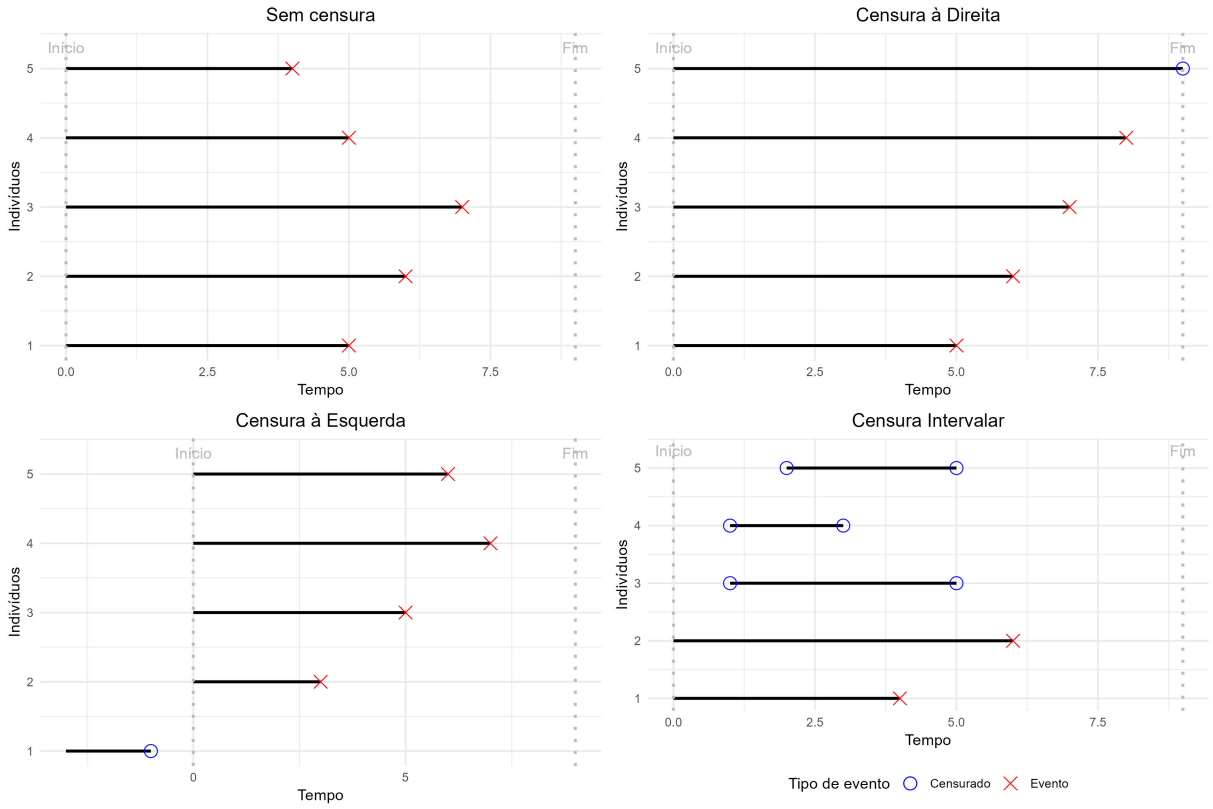
3 Análise de Sobrevivência

Estudos envolvendo dados de tempo até o evento são numerosos e surgem em todas as áreas de pesquisa. A análise de sobrevivência é um campo bem estabelecido na modelagem estatística tradicional e está relacionada ao estudo dos dados de tempo até a ocorrência de um determinado evento.

Neste caso, o tempo de ocorrência de um evento não é necessariamente observado para todas as observações amostrais, gerando observações parciais que são denominadas *observações censuradas*. Mesmo sendo parciais, as observações censuradas devem ser utilizadas na análise estatística, pois ainda sim fornecem informações sobre o tempo até o evento de interesse, evitando conclusões viciadas.

Existem alguns tipos de censura, os quais são explicados a seguir:

- **Censura à direita**: ocorre quando o evento de interesse não foi observado até o término do experimento. Ou seja, sabe-se que o tempo de falha do indivíduo é maior que um determinado valor, mas não se sabe o tempo exato. Assim, o tempo de falha está à direita do tempo registrado.
- **Censura à esquerda**: ocorre quando o evento de interesse aconteceu antes do indivíduo entrar no experimento, mas não se sabe o tempo exato do evento. Ou seja, sabe-se que o tempo de falha do indivíduo é menor que determinado valor.
- **Censura intervalar**: ocorre quando o evento de interesse foi observado em algum momento dentro de um intervalo de tempo, mas não se sabe o tempo exato. Ou seja, sabe-se que o tempo de falha do indivíduo está entre dois valores.



Tipos de censura.

Para o cenário de censura à direita, em especial a censura aleatória (em que a censura ocorre de forma imprevisível), seja T uma variável aleatória que representa o tempo de falha e C outra variável aleatória independente de T que representa o tempo de censura de um indivíduo. Uma forma de representar os dados observados é a seguinte:

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

$$\delta = \begin{cases} 1, & \text{se } T \leq C, \\ 0, & \text{se } T > C. \end{cases}$$

Assim, $\delta = 1$ quando não há censura e $\delta = 0$ no caso de uma observação censurada.

No contexto de regressão e aprendizado de máquina, os dados podem ser representados de diferentes formas, dependendo do tipo de problema e da natureza das observações. Suponha que o tempo de ocorrência até o evento é dado por uma variável rotulada y_i e seja $\mathbf{x}_i \in \mathbb{R}^p$ o vetor de covariáveis da i -ésimo observação, para $i = 1, \ldots, n$. Desta forma, tem-se:

- **Rótulos de pontos únicos:** Este é o caso da regressão padrão, onde cada amostra $\mathbf{x}_i$ tem um alvo pontual $y_i \in \mathbb{R}$. Assim, as tuplas $(\mathbf{x}_i, y_i)$ com $i = 1, \ldots, n$ denotam um conjunto de dados para regressão típica.

- **Rótulos para Classe Binária**: Os rótulos das classes binárias são geralmente denotados por $\{\pm 1\}$. Neste caso, um conjunto de dados é o conjunto de tuplas $(\mathbf{x}_i, y_i)$ com $y_i \in \{\pm 1\}$.

- **Rótulos Intervalares:** são amostras para as quais temos um limite superior e um limite inferior na variável resposta, sendo $y_i \in (l_i, u_i)$. A tupla $(\mathbf{x}_i, l_i, u_i)$ com $l_i < u_i, l_i \in \mathbb{R}, u_i \in \mathbb{R}, \mathbf{x}_i \in \mathbb{R}^p$ denota uma rótulo intervalar.

- **Tempos de Sobrevivência**: Uma amostra não censurada na análise de sobrevivência é o mesmo que uma variável resposta de ponto único definida acima. Uma amostra censurada à direita é escrita como

$(x_i, l_i, +\infty)$ cujo tempo de sobrevivência é maior que $l_i \in \mathbb{R}$, amostras censuradas à esquerda são escritas como $(x_i, -\infty, u_i)$ cujo tempo de sobrevivência é no máximo $u_i \in \mathbb{R}$.

Os rótulos intervalos são o tipo mais geral de observações para análise de sobrevivência, pois englobam todos os tipos de censura, desde censura à esquerda e à direita até censura intervalar, desde que se saiba que um tempo t, $(y = t)$, até o ocorrência do evento satisfaz $l \leq t \leq u$.

A aplicação de aprendizado de máquina na análise de sobrevivência apresenta vário desafios, dos que podem,os destacar:

- **Tratamento de Dados Censurados**: A maioria dos algoritmos tradicionais de aprendizado de máquina não é projetada para lidar com censura. É necessário adaptar os métodos ou desenvolver novas técnicas que considerem adequadamente as informações censuradas.
- **Estimativa de Funções de Risco e Sobrevivência**: Em vez de prever um único valor numérico, muitas vezes é desejável estimar funções inteiras, como a função de sobrevivência $S(t)$ ou a função de risco $h(t)$, que descrevem a probabilidade de sobrevivência além do tempo t e o risco instantâneo de falha, respectivamente, e que serão tratados na próxima seção.
- **Avaliação de Modelos**: Métricas de desempenho convencionais (como erro quadrático médio ou acurácia) podem não ser adequadas para dados censurados. Métricas específicas, como o estimador de Kaplan-Meier ou o índice de C concordante, são usadas para avaliar modelos de sobrevivência.

No que segue, são apresentados os conceitos básicos de Análise de Sobrevivência.

3.1 Conceitos básicos

3.1.1 Função de sobrevivência

A função de sobrevivência indica a probabilidade do evento de interesse não ocorrer até o tempo t, ou seja, a probabilidade de um indivíduo ou objeto sobreviver até um determinado tempo, sem que o evento de interesse (por exemplo, morte, falência, ou falha de um equipamento) ocorra. Essa função é denotada por:

$$S(t) = P(T \geq t)$$

Essa função é interpretada da seguinte forma, supondo que $S(4) = 0.6$, significa que há 60% de probabilidade de um indivíduo não apresentar o evento de interesse até o período de quatro unidades de tempo.

3.1.2 Função de taxa de falha

A taxa de falha em um intervalo $[t, t + \Delta t)$ definida como a probabilidade de que o evento de interesse ocorra neste intervalo de tempo, dado que não ocorreu antes de t, dividida pelo comprimento do intervalo. A probabilidade do evento ocorrer neste intervalo é calculada como $S(t) - S(t + \Delta t)$, enquanto o número de indivíduos que não apresentaram o evento até o tempo t é denotada como $\Delta t S(t)$. Dessa forma, a taxa de falha será

$$h(t) = \frac{S(t) - S(t + \Delta t)}{\Delta t S(t)}$$

Assumindo Δt bem pequeno, é possível obter a taxa de falha instantânea no tempo t, dado que o evento de interesse não ocorreu até o tempo t. Assim, é denotada como

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t}$$

3.1.3 Função de taxa de falha acumulada

A função de taxa de falha acumulada representa o total acumulado de risco de ocorrência de um evento até um determinado tempo t. Essa função é denotada por

$$H(t) = \int_0^t h(u)du$$

A função de taxa de falha acumulada pode ser útil na avaliação da função de taxa de falha $h(t)$. Na estimação não-paramétrica, por exemplo, $H(t)$ apresenta um estimador de propriedades ótimas, enquanto $h(t)$ é difícil de ser estimado.

3.1.4 Relações entre as funções

É possível estabelecer relações entre as funções apresentadas anteriormente, como apresentado abaixo.

$$h(t) = \frac{f(t)}{S(t)} = -\frac{d}{dt}[\log S(t)]$$

$$H(t) = \int_0^t h(u)du = -\log S(t)$$

$$S(t) = \exp\{-H(t)\} = \exp\{-\int_0^t h(u)du\}$$

Logo, a partir do conhecimento de uma das funções é possível calcular as demais.

3.2 Técnicas não-paramétricas

A análise de sobrevivência é voltada para o estudo do tempo até a ocorrência de um evento de interesse, porém a presença de censura nos dados é um problema para as técnicas de análises estatísticas convencionais, tornando o uso de métodos adequados crucial. Nesse contexto, duas abordagens não-paramétricas utilizadas são o estimador de Kaplan-Meier e o teste de Log-rank.

3.2.1 Estimador de Kaplan Meier

O estimador de Kaplan-Meier é uma técnica não paramétrica que estima a função de sobrevivência a partir de dados de sobrevivência, mesmo quando há censura. Esse estimador é uma adaptação da função de sobrevivência empírica, que na ausência de censuras é dada por

$$\hat{S}(t) = \frac{\text{número de observações que não falharam até o tempo } t}{\text{número total de observações no estudo}}$$

Assim, $\hat{S}(t)$ é uma função escada decrescente, com degraus de tamanho $1/n$ nos tempos de falha observados, sendo n o tamanho da amostra.

Também nota-se que para qualquer t, $S(t)$ pode ser escrito em função de probabilidades condicionais. Considerando que existam n indivíduos em um experimento e $k \leq n$ falhas nos tempos $t_1 < t_2 < \ldots < t_k$ e que $S(t)$ é uma função discreta com saltos, ou seja, com probabilidade maior que zero somente nos tempos de falha $t_j, j = 1, \ldots, k$, tem-se que:

$$S(t_j) = (1 - q_1)(1 - q_2)\ldots(1 - q_j),$$

em que q_j é a probabilidade de um indivíduo apresentar o evento de interesse no intervalo $[t_{j-1}, t_j)$, sabendo que o evento não ocorreu até o tempo t_{j-1}. Assim, o estimador de Kaplan Meier estima essa probabilidade como

$$\hat{q}_j = \frac{\text{número de falhas em } t_{j-1}}{\text{número de observações sob risco em } t_{j-1}}$$

A partir disso, é possível obter a expressão geral do estimador de Kaplan Meier como

$$\hat{S}(t) = \prod_{j:t_j<t} \left(\frac{n_j - d_j}{n_j} \right) = \prod_{j:t_j<t} \left(1 - \frac{d_j}{n_j} \right),$$

em que:

- $t_1, \ldots, t_k$, são os k tempos distintos e ordenados de falha,
- d_j o número de falhas em t_j, , $j = 1, \ldots, k$, e
- n_j o número de indivíduos sob risco em t_j.

3.2.2 Estimador de Nelson-Aalen

O estimador de Nelson-Aalen é uma ferramenta fundamental em análise de sobrevivência, usada para estimar a função de risco cumulativo de um conjunto de dados de tempo até um evento. Ele se concentra na **função de risco cumulativo**, que mede o acúmulo do risco ao longo do tempo.

Para um conjunto de dados que inclui tempos de falha e censura, o estimador de Nelson-Aalen estima a função de risco cumulativo $H(t)$, representando o risco acumulado até um tempo t. A fórmula do estimador é dada por:

$$\hat{H}(t) = \sum_{t_i \leq t} \frac{d_i}{n_i},$$

Onde:

- t_i são os tempos observados de falha,
- d_i é o número de eventos (falhas) observados no tempo t_i,
- n_i é o número de indivíduos em risco imediatamente antes do tempo t_i.

3.2.3 Teste de Log-rank

O teste Log-rank é uma técnica não paramétrica usada para comparar as curvas de sobrevivência entre dois ou mais grupos. Ele testa a hipótese nula de que as funções de sobrevivência dos grupos são iguais em todos os tempos, ou seja, testa $H_0 : S_1(t) = S_2(t) = \ldots = S_k(t)$, sendo k o número de estratos. Logo,

rejeitar a hipótese nula significa que pelo menos uma curva difere das outras, significativamente, em algum momento do tempo.

Para isso, o teste compara a distribuição da ocorrência dos eventos observados em cada estrato com a distribuição esperada se a sobrevida fosse igual em todos os estratos. Se a distribuição observada for equivalente à esperada, dizemos que a curva de sobrevivência dos estratos é equivalente à curva de sobrevivência geral, ou seja, a variável que define os estratos não afeta a sobrevivência.

Para a realização do teste, é necessário estimar o número de eventos esperados para cada estrato k, segundo a hipótese nula de sobrevivência igual em todos os estratos, esse número será denotado $E_k(t)$. Também é preciso calcular a estatística do teste, que segue uma distribuição χ^2, com $k-1$ graus de liberdade quando H_0 é verdadeira.

O cálculo do número de eventos esperados para cada estrato é feito proporcionalmente ao número de indivíduos em cada estrato. Assim, sendo $N(t)$ o número total de eventos observados em t, $R_k(t)$ o número de indivíduos sob risco no estrato k no tempo t e $R(t)$ o número total de indivíduos sob risco no estudo no tempo t:

$$E_k(t) = N(t)\frac{R_k(t)}{R(t)}$$

A estatística log-rank será calculada a partir da diferença entre o número total de eventos observados e o número total de eventos esperados no estrato. Para um caso com apenas dois estratos, a estatística do teste é dada por:

$$\text{Log-rank } = \frac{(O_1 - E_1)^2}{Var(E_1)} \sim \chi^2_1, \text{em que}$$

- O_1 é o total de eventos observados no estrato 1,
- E_1 é o total de eventos esperados no estrato 1, e
- $Var(E_1)$ é a variância do número de eventos esperados.

3.3 Modelo de Cox

O modelo de regressão de Cox, também conhecido como modelo de riscos proporcionais de Cox, é o modelo mais utilizado e conhecido na área de Análise de Sobrevivência. Sua principal vantagem é que ele não assume uma distribuição específica para a variável de tempo, permitindo uma flexibilidade maior na modelagem, além de seu forte poder interpretativo a partir do risco relativo, que será descrito no que segue.

Sem perda de generalidade, considere um cenário em que pacientes são selecionados em dois grupos: o grupo 0, que recebe o tratamento padrão, e o grupo 1, que recebe um novo tratamento.

$$x = \begin{cases} 0, & \text{se grupo } 0, \\ 1, & \text{se grupo } 1. \end{cases}$$

A taxa de falha do grupo zero será representada por $h_0(t)$ e do grupo um $h_1(t)$. Assumindo proporcionalidade entre essas funções, tem-se

$$h(t) = \begin{cases} h_1(t) = h_0(t)k, & \text{se } x = 1, \\ h_0(t), & \text{se } x = 0 \end{cases}$$

Se $k = \exp\{\beta x\}$, então o risco de indivíduos com $x = 1$ é $\exp\{\beta x\}$ vezes o risco de indivíduos com $x = 0$.

A expressão acima representa o modelo de Cox para uma única covariável, porém é possível generalizá-lo para p covariáveis, de forma que x seja um vetor com os componentes $X' = (x_1, \ldots, x_p)$. Para encontrar a expressão geral do modelo de regressão de Cox, será considerado $h(t) = h_0(t)g(X\prime\beta)$, em que g é uma função não-negativa que deve ser especificada, de forma que $g(0) = 1$.

O modelo de Cox é composto por uma parte **não paramétrica**, que estima a função de risco base $h_0(t)$, e uma parte **paramétrica**, que modela o efeito das covariáveis na taxa de falha através do termo exponencial $g(X'\beta) = \exp\{X'\beta\} = \exp\{\beta_1 x_1 + \ldots + \beta_p x_p\}$. A função de risco base $h_0(t)$ não precisa de especificação explícita (não paramétrica), enquanto o impacto das covariáveis é estimado parametricamente pelos coeficientes β.

Uma das principais suposições do modelo de Cox é a de riscos proporcionais, que implica que as razões de taxa de falha entre dois indivíduos diferentes são constantes ao longo do tempo. Ou seja, dado dois indivíduos i e j,

$$\frac{h_i(t)}{h_j(t)} = \frac{h_0(t)\exp\{X_i'\beta\}}{h_0(t)\exp\{X_j'\beta\}} = \exp\{X_i' - X_j'\}.$$

Assim, é possível notar que a razão não depende do tempo t, sendo então esta razão de risco a mesma para todo o período de acompanhamento.

Para estimar as quantidades do modelo de Cox, utiliza-se o método de máxima verossimilhança parcial, que permite estimar β sem precisar conhecer $h_0(t)$, e, a partir das estimativas de β, pode-se obter a estimativa da função de risco base usando métodos não paramétricos, como o estimador de Breslow.

4 Aprendizado de Máquina para Dados Censurados

Eles oferecem grande flexibilidade e podem detectar automaticamente certos tipos de interação sem a necessidade de especificá-los previamente.

4.1 Árvores de Sobrevivência

Dado o poder interpretativo que árvores de regressão possuem, a adaptação desse método para o cenário em que os dados são possivelmente censurados se apresenta como uma evolução natural. Seria necessário, então, encontrar uma regra de decisão e uma regra de parada que levassem em conta a presença de censura nas observações. O que acontece, entretanto, é que dados de sobrevivência tipicamente não possuem medidas naturais de “impureza” ou de homogeneidade entre nós, o que dificulta a adaptação da regra de divisão de “redução de impureza” do algoritmo CART. Pela mesma razão, uma função de perda que estime o custo do desvio entre um valor predito e seu valor verdadeiro não pode ser facilmente definida, o que impede que o custo-complexidade de uma árvore, um elemento essencial no processo de podagem, seja avaliado.

A primeira ideia de adaptação do algoritmo CART para o contexto da sobrevivência foi proposta por Gordon e Olshen (1985). Nessa tentativa, os autores criaram uma medida de impureza baseada no estimador de Kaplan-Meier, definindo três possíveis formatos de curvas de sobrevivência que seriam consideradas "puras" e calculando a impureza de um nó a partir da diferença entre alguma dessas curvas e a curva de sobrevivência desse nó. Desde então, mais de dez algoritmos de árvores de sobrevivência, que em geral podem ser agrupados em duas categorias, foram propostos. A primeira categoria, dentro da qual se encontra a proposta de Gordon e Olshen (1985), é formada por algoritmos cujas regras de divisão visam maximizar a homogeneidade dentro de um nó, assim como acontece no algoritmo CART. A segunda delas, por outro lado, é composta por propostas cujas regras de divisão visam maximizar a heterogeneidade entre os nós, utilizando, por exemplo, a já apresentada estatística de logrank, como em Segal (1988). Para mais informações a respeito de cada um desses algoritmos, recomendamos a leitura de Zhou e McArdle (2015) e Hu e Steingrimsson (2018).

No R (R Core Team 2023), implementações de árvores de sobrevivência estão disponíveis nos pacotes `rpart` (Therneau e Atkinson 2022) e `party` (Zeileis, Hothorn, e Hornik 2008). Em sua implementação, o pacote `rpart` utiliza a regra de divisão proposta por LeBlanc e Crowley (1992), que assume um modelo de riscos proporcionais para estimar a impureza de um nó a utilizando a verossimilhança. O pacote `party`, por outro lado, implementa um procedimento de inferência condicional proposto por Hothorn, Hornik, e Zeileis (2006), que utiliza testes de permutação tanto para a regra de decisão quanto para o critério de parada. Se a hipótese nula do teste de permutação for rejeitada, o nó é dividido utilizando a valor da covariável que maximiza a estatística de logrank; caso contrário, a divisão da árvore é parada. Ambos os pacotes permitem que o critério de parada seja definido, também, através dos argumentos "minsplit", que controla o número mínimo de observações que um nó precisa ter que uma divisão seja testada, e "minbucket", que controla o número mínimo de observações que um nó terminal da árvore pode ter. Exemplos de ajustes de árvores de sobrevivência no R, construídos utilizando parte de um conjunto de dados de pacientes com câncer do colo do útero que apresentaremos posteriormente, podem ser vistos nas Figuras 6 e 7.

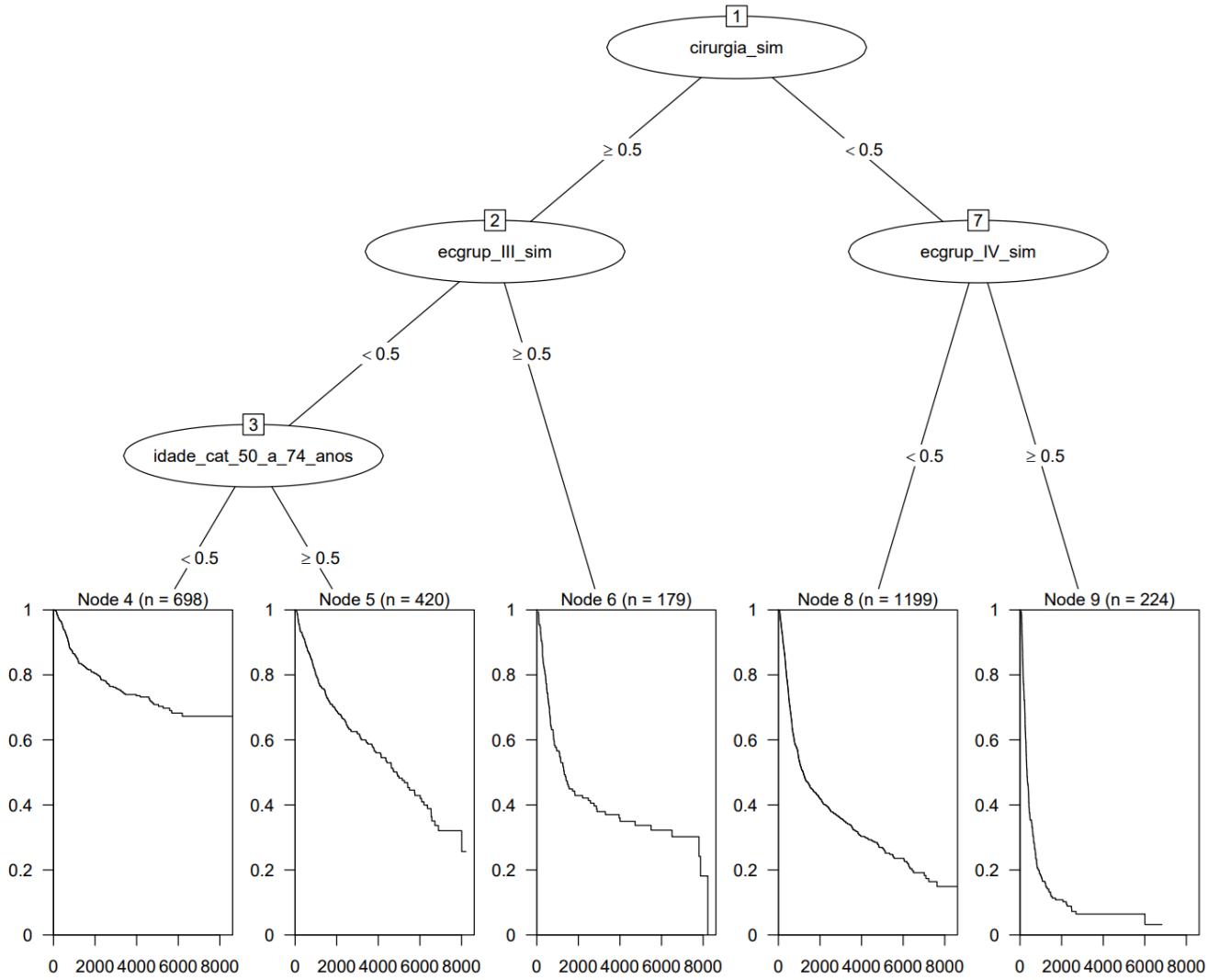


Fig 6. Exemplo de árvore de sobrevivência obtida pelo pacote `rpart`. Para uma melhor visualização, o objeto de classe `rpart` que continha o modelo ajustado foi convertido para um objeto de classe `party` através da função `as.party()`, do pacote `partykit`. Os gráficos nos nós terminais representam a curva de sobrevivência estimada via Kaplan-Meier para as observações de cada nó. A variável resposta considerada foi o tempo, em dias, até a morte por câncer do colo do útero.

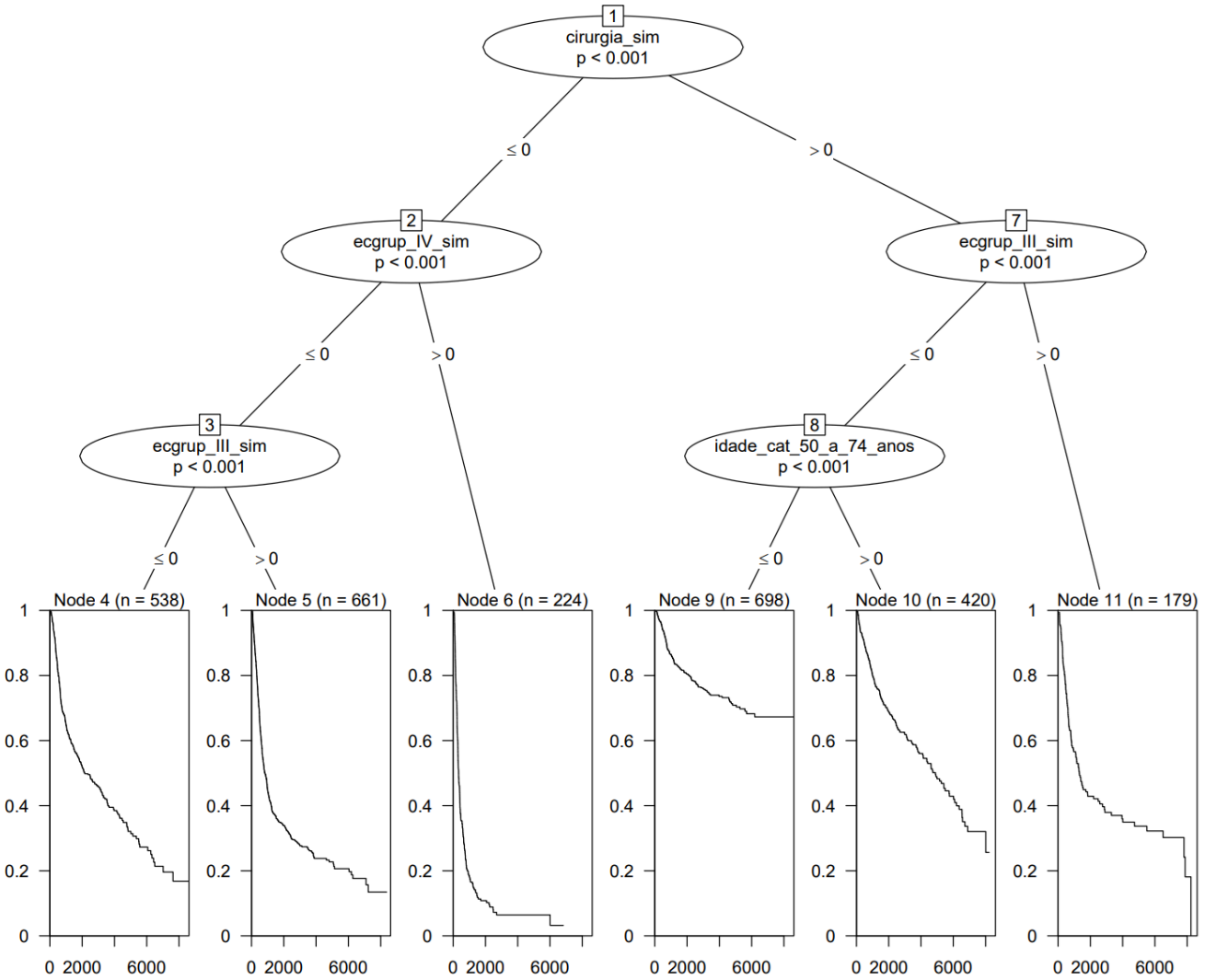


Fig 7. Exemplo de árvore de sobrevivência obtida pelo pacote `party`. Os gráficos nos nós terminais representam a curva de sobrevivência estimada via Kaplan-Meier para as observações de cada nó. A variável resposta considerada foi o tempo, em dias, até a morte por câncer do colo do útero.

Ademais, assim como ocorre com as árvores de regressão e classificação, as árvores de sobrevivência são candidatas ideais para a combinação por meio de um método ensemble e podem, assim, ser transformadas em ferramentas preditivas com maior capacidade de predição, como as florestas aleatórias de sobrevivência. Dessa forma, como já apresentamos o algoritmo das árvores de regressão e classificação, e considerando que as principais diferenças estariam na forma como a resposta é estimada e nas regras de decisão e de parada, utilizaremos o algoritmo das florestas aleatórias de sobrevivência, que será apresentado na seção seguinte, para traçar alguns paralelos entre a forma como as árvores e as florestas de sobrevivência são construídas.

4.2 Florestas Aleatórias de Sobrevivência

Propostas por Hemant Ishwaran et al. (2008) como uma extensão das florestas aleatórias de Breiman (Leo Breiman 2001b), as florestas aleatórias de sobrevivência são um método ensemble de árvores para a análise de dados de sobrevivência com censura à direita. Em contraste com os métodos de análise de sobrevivência mais comumente utilizados, as florestas aleatórias de sobrevivência não dependem de suposições restritivas como a de riscos proporcionais, e podem facilmente lidar com dificuldades que

frequentemente surgem quando utilizamos métodos paramétricos, como efeitos não lineares e interações entre variáveis.

4.2.1 O algoritmo

Considere que os dados sejam denotados por $(T_i, \mathbf{x}_i, \delta_i)$, onde $\mathbf{x}_i$ é o vetor de covariáveis para o indivíduo i e T_i e δ_i são o tempo observado e a variável indicadora de censura para $i, i = 1, 2, \ldots, n$. O algoritmo das florestas aleatórias de sobrevivência, como proposto por Hemant Ishwaran et al. (2008), é dado como se segue:

1. Do conjunto de dados original, retire B amostras bootstrap. Em média, cada amostra bootstrap irá excluir aproximadamente 37% dos dados. Essa parte dos dados é chamada de dados *out-of-bag* (dados OOB), e é utilizada para se estimar o erro de predição do modelo;

2. Para cada amostra bootstrap, ajuste uma árvore de sobrevivência. Em cada nó da árvore, selecione aleatoriamente p variáveis candidatas. Divida o nó utilizando o valor da variável candidata que **maximize a diferença de sobrevivência entre os nós filhos**, utilizando uma regra de divisão (*splitting rule*) previamente decidida. Note que, nessa etapa, os algoritmos das árvores e das florestas de sobrevivência se diferenciam, uma vez que, para as florestas, apenas parte das covariáveis é candidata à divisão em cada nó da árvore, e a regra de decisão busca sempre obter nós heterogêneos entre si;

3. Cresça a árvore até o seu tamanho máximo, sob a restrição de que um nó terminal deve conter um mínimo de $d_0 > 0$ eventos únicos;

4. Para uma árvore completamente crescida, calcule a função de risco acumulado (CHF) para um dado tempo t utilizando o estimador de Nelson-Aalen,

$$\hat{H}_h(t) = \sum_{t_{j,h} \leq t} \frac{d_{j,h}}{Y_{j,h}},$$

onde $t_{1,h} < t_{2,h} < \cdots < t_{N(h),h}$ são os $N(h)$ tempos de evento distintos no nó terminal h, e $d_{j,h}$ e $Y_{j,h}$ são os números de eventos e de indivíduos sob risco no tempo $t_{j,h}$. Para estimar a CHF *in bag* de um indivíduo com um dado vetor de covariáveis $\mathbf{x}_i$, jogue $\mathbf{x}_i$ pela árvore. Por conta da natureza binária de uma árvore, $\mathbf{x}_i$ irá pertencer a um único nó terminal h. A CHF *in bag* estimada para $\mathbf{x}_i$ será igual ao estimador de Nelson-Aalen para o nó terminal de $\mathbf{x}_i$:

$$\hat{H}^{IB}(t|\mathbf{x}_i) = \hat{H}_h(t), \quad \text{se } \mathbf{x}_i \in h.$$

Para obter uma estimativa emsemble para a CHF *in bag* do indivíduo i, obtenha a média das CHFs *in bag* de cada uma das B árvores de sobrevivência, ou seja,

$$\hat{H}_e^{IB}(t|\mathbf{x}_i) = \frac{1}{B} \sum_{b=1}^{B} \hat{H}_b^{IB}(t|\mathbf{x}_i).$$

5. De maneira similar, seja $I_i \in \{0, 1\}$ uma variável indicadora que indique se o indivíduo i é *in bag* ou *out-of-bag* em uma dada árvore (ou seja, se o indivíduo i está na amostra utilizada para a construção da árvore ou não). Jogue i pela árvore e denote por h o nó terminal do indivíduo i. O estimador da CHF *out-of-bag* para i nessa árvore é dado por

$$\hat{H}^{OOB}(t|\mathbf{x}_i) = I_i \hat{H}_h(t).$$

Para obter uma estimativa ensemble para a CHF *out-of-bag* do indivíduo i, obtenha a média das CHFs das árvores em que i é OOB, ou seja,

$$\hat{H}_e^{OOB}(t|\mathbf{x}_i) = \frac{\sum_{b=1}^{B} \hat{H}_b^{OOB}(t|\mathbf{x}_i)}{\sum_{b=1}^{B} I_{i,b}}.$$

Observe que o estimador ensemble da CHF *in bag* utiliza todas as árvores de sobrevivência, enquanto o estimador OOB utiliza apenas aquelas em que i não estava presente na amostra bootstrap.

4.2.2 A mortalidade

Apresentados os estimadores ensemble para a CHF de um indivíduo, podemos definir o valor predito das florestas aleatórias de sobrevivência: a **mortalidade**. A mortalidade leva em conta um princípio de conservação de eventos que define que, para um nó da árvore, a soma da CHF estimada em todos os seus tempos de sobrevivência é igual ao número total de mortes nesse nó. A mortalidade é, então, definida como sendo o valor esperado da soma da CHF em todos os tempos T_j condicionado a um $\mathbf{x}_i$ específico, ou seja,

$$M_i = E_i\left(\sum_{j=1}^{n} H(T_j|\mathbf{x}_i)\right),$$

onde E_i é o valor esperado sob a hipótese nula de que todos os indivíduos são similares a i. Dessa forma, a mortalidade mede o número de mortes esperadas na amostra caso todos os indivíduos fossem semelhantes a i.

Como a estrutura de uma árvore de sobrevivência força a hipótese nula de comportamento de sobrevivência similar dentro de seus nós terminais, a mortalidade pode ser naturalmente estimada dentro do paradigma desse método. O estimador ensemble da mortalidade *in bag* é definido para um indivíduo i como sendo

$$\hat{M}_{e,i}^{IB} = \sum_{j=1}^{n} H_e^{IB}(T_j|\mathbf{x}_i).$$

De maneira similar, o estimador ensemble da mortalidade OOB é dado por

$$\hat{M}_{e,i}^{OOB} = \sum_{j=1}^{n} H_e^{OOB}(T_j|\mathbf{x}_i).$$

As estimativas ensemble da mortalidade OOB são utilizadas como o risco estimado pelo modelo para o cálculo do *Concordance Index* (C-Index), o qual, por sua vez, é utilizado para estimar o erro de predição das florestas aleatórias de sobrevivência. Definiremos o C-Index com maior precisão em uma seção subsequente. Por agora, é suficiente entender que essa estatística estima a capacidade que um modelo possui para ordenar quaisquer indivíduos da amostra em termos de seus riscos estimados.

É interessante notar que, para as florestas aleatórias de sobrevivência, não é necessária a utilização de métodos de validação cruzada em nenhuma etapa do ajuste, uma vez que o próprio algoritmo desse método garante que uma parcela da amostra original não seja utilizada na construção de cada árvore (a amostra OOB). Essa parcela dos dados pode ser, então, utilizada para verificar o desempenho preditivo de

cada árvore, e a média dos desempenhos de todas as árvores fornece uma estimativa do desempenho preditivo da floresta como um todo.

4.3 Regressão Censurada por Vetores de Suporte

A Regressão Censurada por Vetores de Suporte (SVCR - Support Vector Censored Regression) (Shivaswamy, Chu, e Jansche 2007; Van Belle et al. 2011; Maia et al. 2023) é um método de aprendizado de máquina baseado em SVR para dados intervalares.

Para este caso, é desejável que o valor predito de $\mathbf{x}_i$ esteja contido no intervalo (l_i, u_i). Assim, a função de perda ε-insensível (Figura 8) pode ser generalizada para esta situação. Tal generalização é exposta na Figura 9.

- Função de Perda ε-insensível tradicional

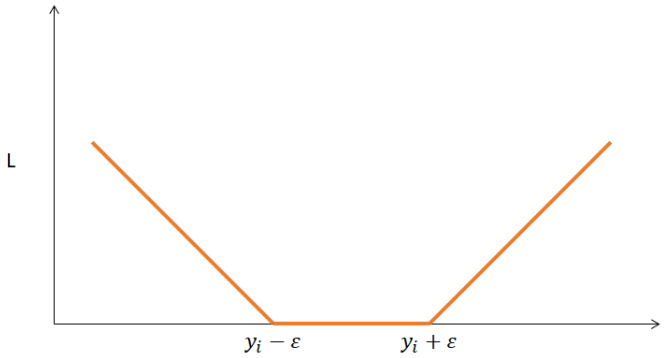


Fig 8. Função de Perda epsilon-insensível

- Função de Perda ε-insensível modificada para dados censurados

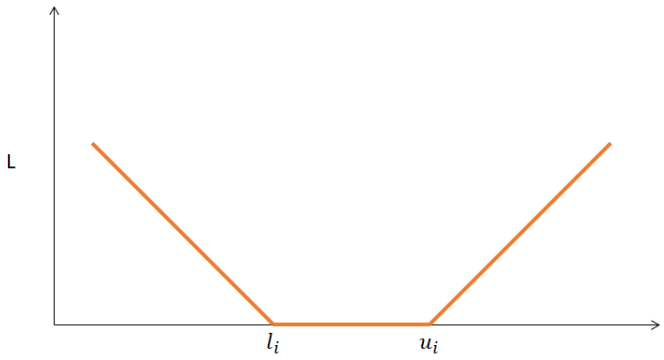


Fig 9. Função de perda epsilon-insensível para dados censurados

A função de perda modificada pode ser escrita da seguinte forma,

$$L = \max\left(l_i - f(\mathbf{x}_i), f(\mathbf{x}_i) - u_i\right)$$

quando $l_i = -\infty$ ou $u_i = +\infty$, a função de perda torna-se unilateral.

Desta forma, a estimação dos parâmetros do SVCR é modificada no processo de otimização levando em consideração a nova função de perda ε-insensível para dados censurados.

Seja

$$L = \{l_i, +\infty\}$$

$$U = \{-\infty, u_i\}$$

$$\min_{b,\mathbf{w},\xi,\xi^*} \frac{1}{2}\mathbf{w} \bullet \mathbf{w} + C\left(\sum_{i\in U}\xi_i + \sum_{i\in L}\xi_i^*\right)$$

$$\text{sujeito a } \begin{cases} \xi_i \geq 0 & \forall i \in U \\ \xi_i^* \geq 0 & \forall i \in L \\ b + \mathbf{w}\cdot\mathbf{x}_i - u_i \leq \xi_i^* & \forall i \in U \\ l_i - b - \mathbf{w}\cdot\mathbf{x}_i \leq \xi_i & \forall i \in L \end{cases}$$

O SVCR utiliza todas as observações completas ou censuradas disponíveis na amostra de treinamento.

Expandindo esta formulação através dos multiplicadores de Lagrange, é possível realizar o truque do kernel de semelhante ao SVR.

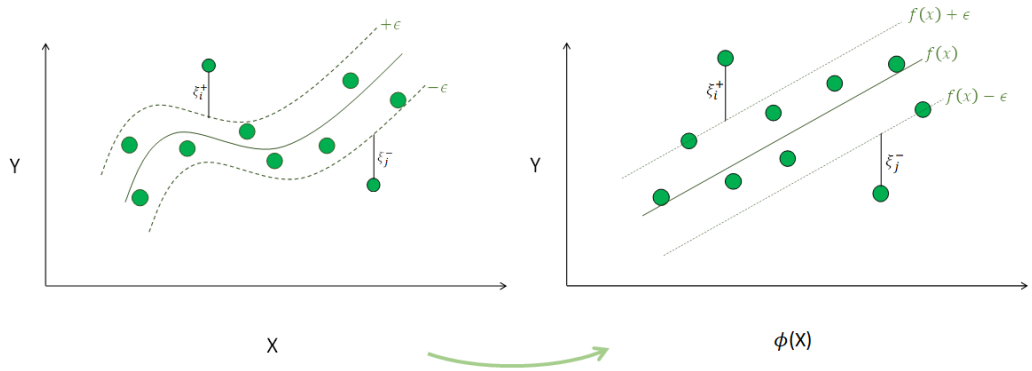


Fig 10. Representação de um hipertudo no espaço de características do SVR.

A Figura 10 ilustra o mapeamento dos dados para um espaço de característica superior, por meio da função ϕ, sendo o kernel entre $\mathbf{x}_i$ e $\mathbf{x}_j$ dado por $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$.

Para a implementação do método SVCR é utilizado o pacote `survivalsvm`, o qual permite a escolha dos kernels linear, aditivo (Daemen e De Moor 2009), gaussiano e polinomial, por meio dos atributos `lin_kernel`, `add_kernel`, `rbf_kernel` e `poly_kernel`, respectivamente. Mais detalhes sobre as aplicações do SVCR para a linguagem R podem ser vistos em Fouodo et al. (2018).

4.4 Índice de Concordância

Muitas vezes tem-se o objetivo de ranquear elementos de acordo com a suscetibilidade a ocorrência do evento de interesse. Dados censurados podem ser traduzidos em pares ordenados e, em seguida, em uma função de classificação pode ser ajustada para ordenar as amostras. Se tivermos uma função de ordenação f perfeita, ela iria prever $f(\mathbf{x}_i) < f(\mathbf{x}_j)$ sempre que $u_i \leq l_j$.

O Índice de Concordância (C − index) é definido como $P(\text{concordância})$ para quaisquer duas observações escolhidas aleatoriamente. Neste caso, a concordância significa que a observação com o menor tempo de sobrevivência dos dois também tem a maior pontuação de risco (Brentnall e Cuzick 2018).

Esta medida estima o grau de concordância entre as previsões do modelo ou estimativas de risco com relação aos dados de sobrevivência observados. De forma comum, o preditor (ou pontuação de risco) é baseado em um modelo de Cox, porém é possível utilizar outros tipos de regressão.

O Índice de Concordância é definido como (J. Wang, Williams, e Karafili 2018),

$$\text{C-index(v)} = \frac{\sum_{i=1}^{n} \sum_{j \neq i} I\left[(v(\mathbf{x}_i) - v(\mathbf{x}_j)) \times ((y_i - y_j)) \geq 0\right]}{\sum_{i=1}^{n} \sum_{j \neq i} \text{comp}(i,j)},$$

com $0 \leq \text{C-index(v)} \leq 1$, sendo $v(\mathbf{x}_i)$ é uma função de predição para $\mathbf{x}_i$ - por exemplo, o risco estimado por um modelo de Cox - I função indicadora e $\text{comp}(i,j)$ é um indicador de comparabilidade para um par de observações $\{(\mathbf{x}_i, y_i, \delta_i), (\mathbf{x}_j, y_j, \delta_j)\}$ definido como,

$$\text{comp}(i,j) = \begin{cases} 1 & \text{se } \delta_i = 1 \text{ e } \delta_j = 1, \\ & \delta_i = 1 \text{ e } \delta_j = 0 \text{ e } y_i \leq y_j, \\ 0 & \text{caso contrário.} \end{cases}$$

Um C-index com valor 1 indica concordância perfeita. Por outro lado, 0,5 significa uma previsão aleatória.

Ao contrário de outras medidas de desempenho de sobrevivência, o C-index não depende de um único tempo fixo, sendo um índice global para validar a capacidade preditiva de um método de sobrevivência.

O C-index pode ser calculado pela função `survConcordance` do pacote `survival`.

5 Aplicações em dados reais

Esta seção apresenta diversas aplicações para dados reais dos métodos citados por meio da Linguagem R.

5.1 Internação de pacientes cardiopatas submetidos à cirurgia cardíaca

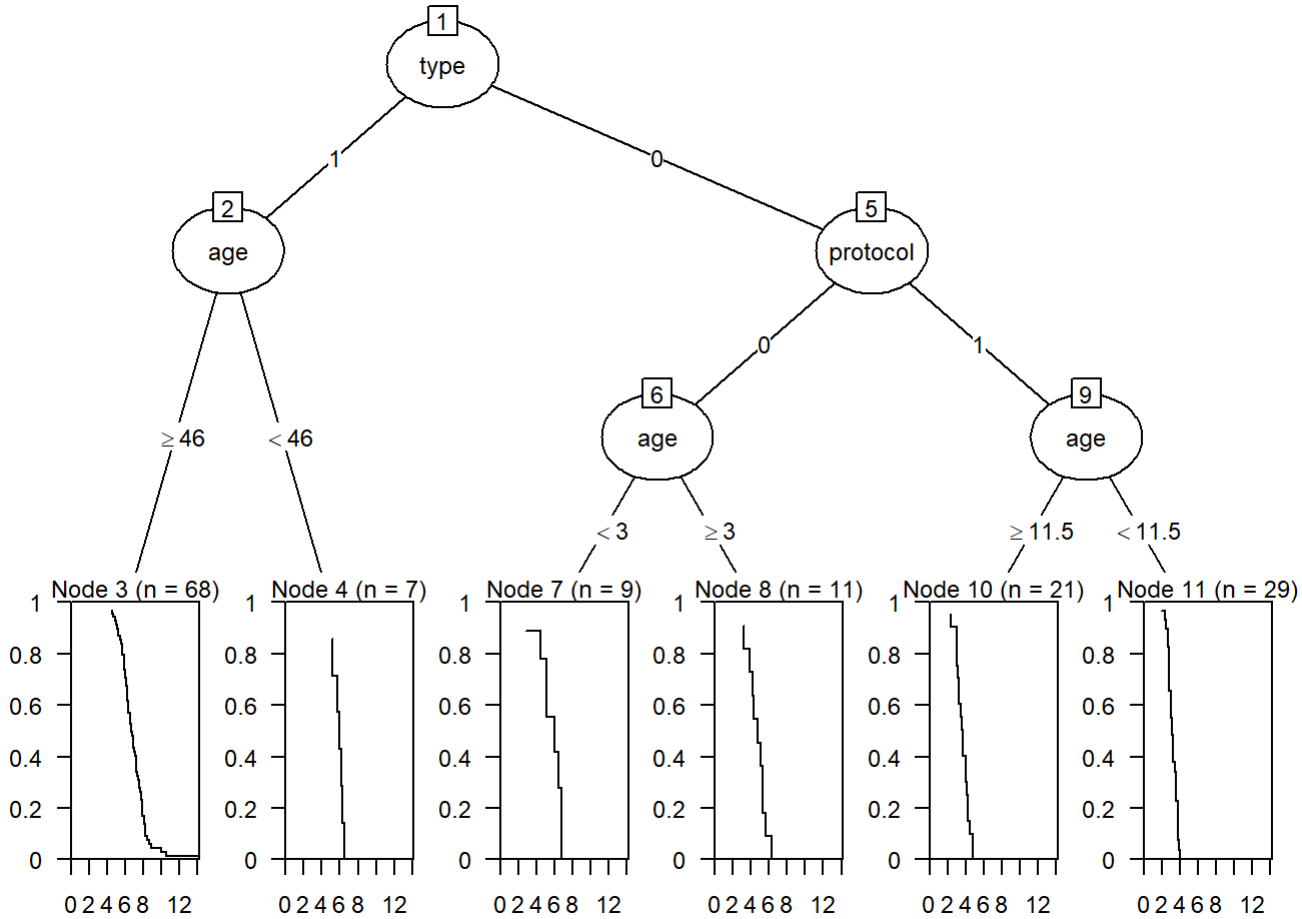
Este conjunto de dados é baseado em pesquisa conduzida pelo Instituto do Coração do Hospital das Clínicas da Faculdade de Medicina da Universidade de São Paulo, Brasil, com o objetivo de comparar o tempo de internação de pacientes cardiopatas submetidos à cirurgia cardíaca (Fernandes et al. 2004; Maia et al. 2023). O estudo considerou 145 pacientes e 7 variáveis, das quais cinco são as variáveis explicativas, bem como a variável resposta e o indicador de censura. A variável resposta, T, representa a quantidade de tempo (em horas) da admissão do paciente até sua alta da enfermaria cirúrgica e é acompanhada por uma variável indicadora, δ. Se $\delta = 0$, há censura, o que significa que o tempo exato de internação é desconhecido. Se $\delta = 1$, o tempo de internação é conhecido exatamente. Também foram levadas em consideração as seguintes variáveis explicativas: idade do paciente em anos (X_1), tipo de protocolo (X_2), que pode ser convencional (0) ou fast track (1), raça (X_3), que é dividida na classificação brasileira em branca (1), negra (2) e amarela (3), sexo (X_4), que é dividido entre feminino (0) e masculino (1), e tipo de paciente (X_5), separado em congênito (0) e coronário (1).

```
#=========== Packages =====================
require(readxl)
require(rpart)
require(partykit)
require(survival)
require(survivalsvm)
require(randomForestSRC)

#=========== Dataset =====================
```

```r
dados <- read_excel("codes//data//viarapida.xlsm",sheet=1, col_names=T)
colnames(dados)<-c("time", "status_c", "age", "protocol", "race", "gender", "type")

dados$protocol <- factor(dados$protocol)
dados$race <- factor(dados$race)
dados$gender <- factor(dados$gender)
dados$type <- factor(dados$type)

### NAIVE COX
mod.cox <- coxph(Surv(time, status_c) ~ .,data=dados)
surv_time_hat <- predict(mod.cox, newdata=dados)

survConcordance(Surv(time, status_c) ~ surv_time_hat,
                data=dados)$concordance
```

```
concordant
  0.758584
```

```r
### Árvore de Sobrevivência
mod.sdt <- rpart(Surv(time, status_c) ~ .,
                 data=dados)

plot(as.party(mod.sdt), gp = gpar(fontsize = 9))
```




```r
preds <- predict(mod.sdt, newdata=dados)
surv_time_hat <- as.numeric(preds)
```

```
survConcordance(Surv(time, status_c) ~ surv_time_hat,
                data=dados)$concordance
```

```
concordant
 0.7896283
```

```
## Regressão Censurada por Vetores de Suporte
mod.svcr <- survivalsvm(Surv(time, status_c) ~ .,
                    dados, type="regression",
                    kernel="rbf_kernel",
                    gamma.mu = 1)

preds <- predict(mod.svcr, newdata=dados)$predicted
surv_time_hat <- as.numeric(preds)

1-survConcordance(Surv(time, status_c) ~ surv_time_hat,
                data=dados)$concordance
```

```
concordant
 0.8837739
```

5.2 Tempo até a parada de amamentação

Esse é um conjunto de dados real do Departamento de Obstetrícia da Faculdade de Medicina da USP (FM-USP), que apresenta informações de recém-nascidos gemelares, cuja variável de interesse é o tempo até a parada total de amamentação, em que algumas observações apresentam censura. A base foi formada observando filhos recém-nascidos de mães grávidas de gêmeos, com os dados coletados via entrevistas. Vamos usar como variável resposta a variável `Tempo_vida_ate_parada_amament_ate180`, que representa o tempo, em dias, até a parada total da amamentação, e como variável indicadora de falha a variável `censura`, que recebe 1 caso a mãe tenha parado totalmente de amamentar e 0 caso essa informação seja censurada. A seguir temos uma tabela que descreve o que cada variável representa.

Descrição das variáveis do conjunto de dados de tempo até a parada total de amamentação

Variável	Descrição
Idade_menorIgual26	Idade da mãe
escolaridade_cat	Escolaridade da mãe
Horas_trab__menor_Igual7	Horas de trabalho fora
Mora_com_compan	Se mora com companheiro
Gravid_plan	Se a gravidez foi planejada
Gravid_desej	Se a gravidez foi desejada
Tipo_parto	Via de parto
IG_parto_36	Idade gestacional do parto
Complic_relac_parto	Se teve alguma complicação relacionada ao parto
UTI	Se a mãe foi para UTI

Variável	Descrição
Dificuldade_para_amamentar	Se relatou dificuldade para amamentar
Amament_prev_SN	Se já amamentou antes
media_amament_prev_12	Média do tempo médio de amamentação prévia
Amamen_Exclusiva_SN	Se a amamentação é exclusiva
censura	Indicadora de censura
Tempo_vida_ate_parada_amament_ate180	Tempo até a parada de amamentação em dias

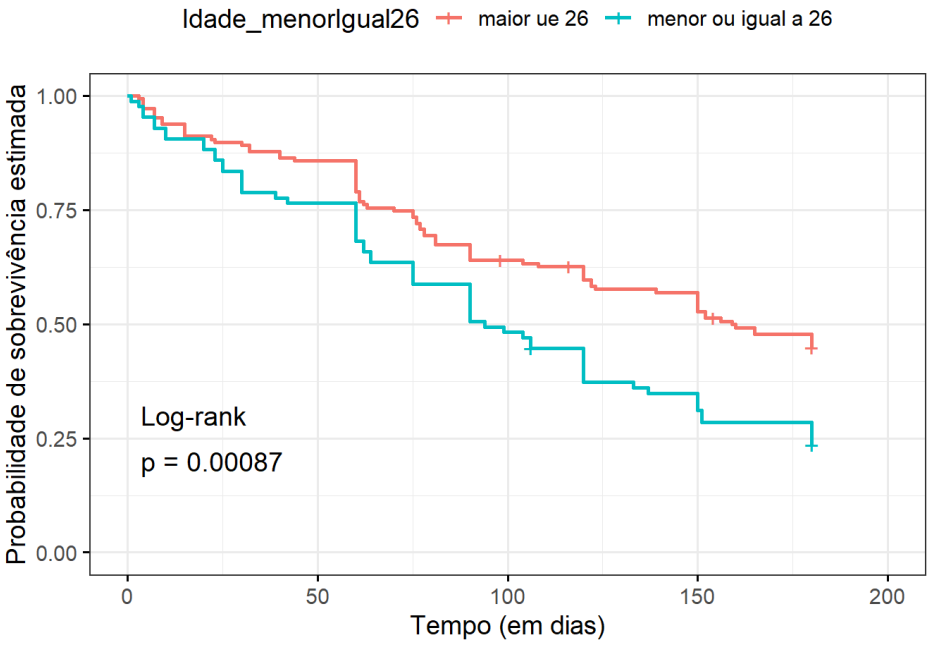
Primeiro é necessário carregar os pacotes a serem utilizados e importar a base de dados utilizada nessa aplicação.

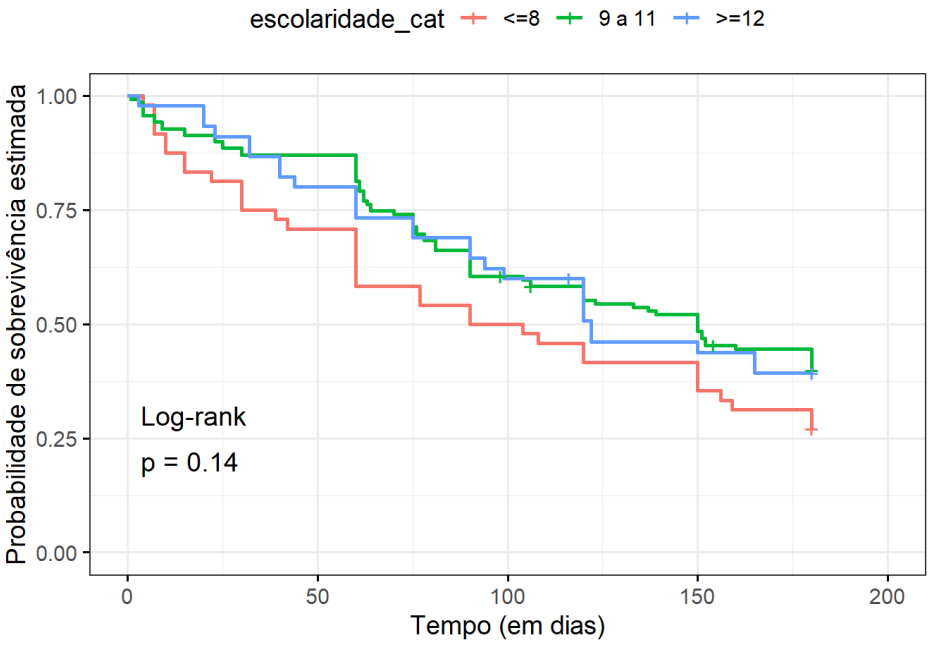
```
#========== Packages =====================
require(readxl)
require(rpart)
require(rpart.plot)
require(survival)
require(survivalsvm)
require(randomForestSRC)
require(dplyr)
require(survminer)
require(caret)

#========== Dataset =====================
data <- readRDS("codes//data//dados_amamenacao_usp.rds")
```

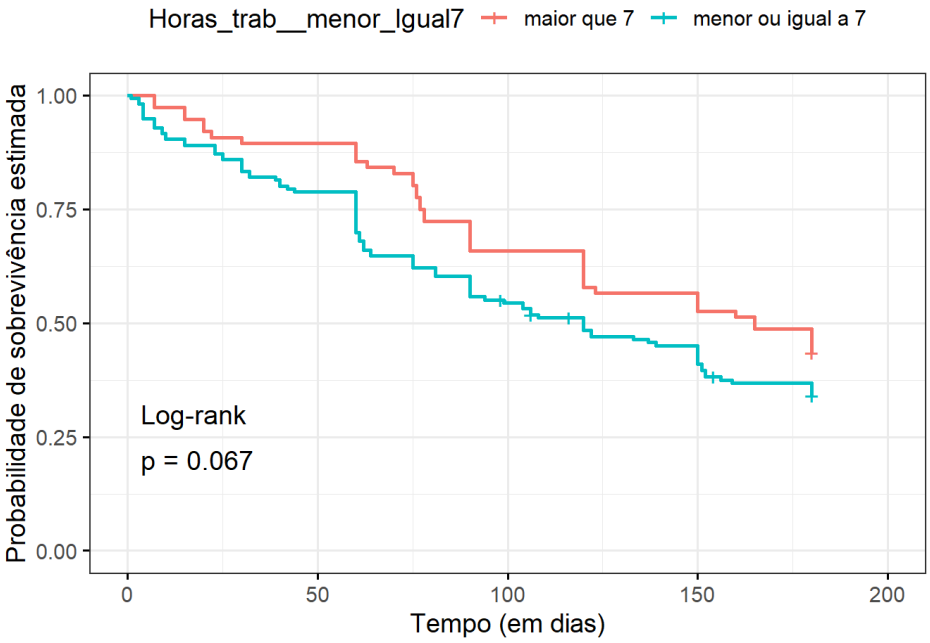
Abaixo, usamos as funções `survfit`, que ajusta o modelo de Kaplan-Meier para cada grupo definido pela variável x; `ggsurvplot`, que constrói as curvas de sobrevivência para cada grupo e `survdiff`, usado para realizar o teste de Log-rank para comparar as curvas de sobrevivência entre os grupos. Nesse caso, os gráficos de Kaplan Meier nos permitem visualizar a probabilidade de sobrevivência ao longo do tempo para diferentes níveis de uma variável, enquanto o teste de Log-rank verifica se as diferenças entre as curvas são significativas. Aqui podemos observar que algumas das variáveis que apresentaram diferença entre as curvas foram Dificuldade_para_amamentar, Amamen_Exclusiva_SN e IG_parto_36.

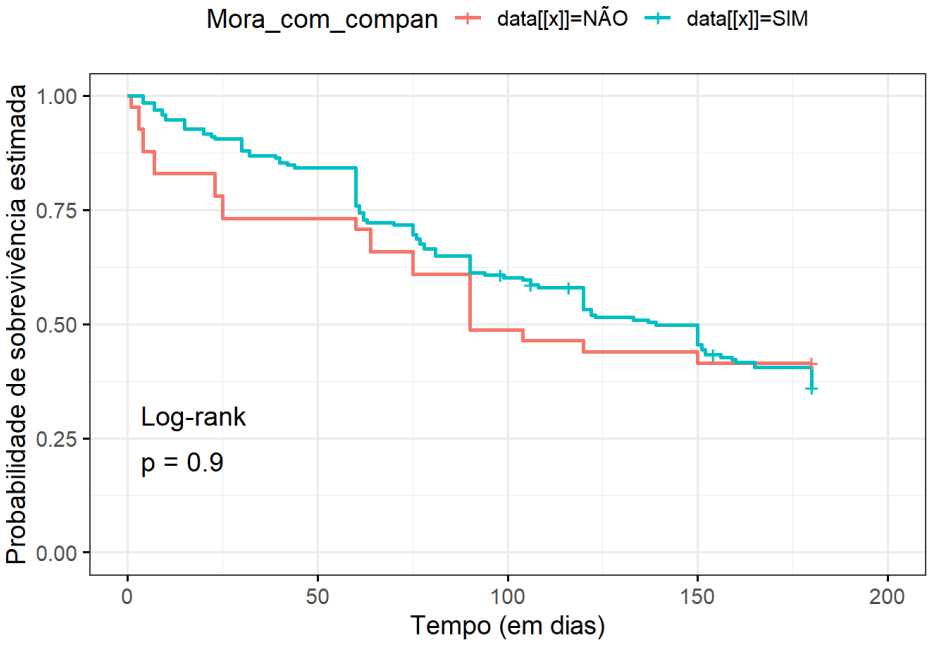
```
variaveis <- colnames(data)[1:14]

lista_de_plots <- list()
df_logrank <- data.frame()

for (i in 1:length(variaveis)) {
  x <- variaveis[i]

  ekm <- survfit(Surv(data$Tempo_vida_ate_parada_amament_ate180, data$censura) ~
                   data[[x]])

  p <- ggsurvplot(
    ekm,
    data = data,
    pval = TRUE,
    pval.method = TRUE,
    conf.int = FALSE,
```

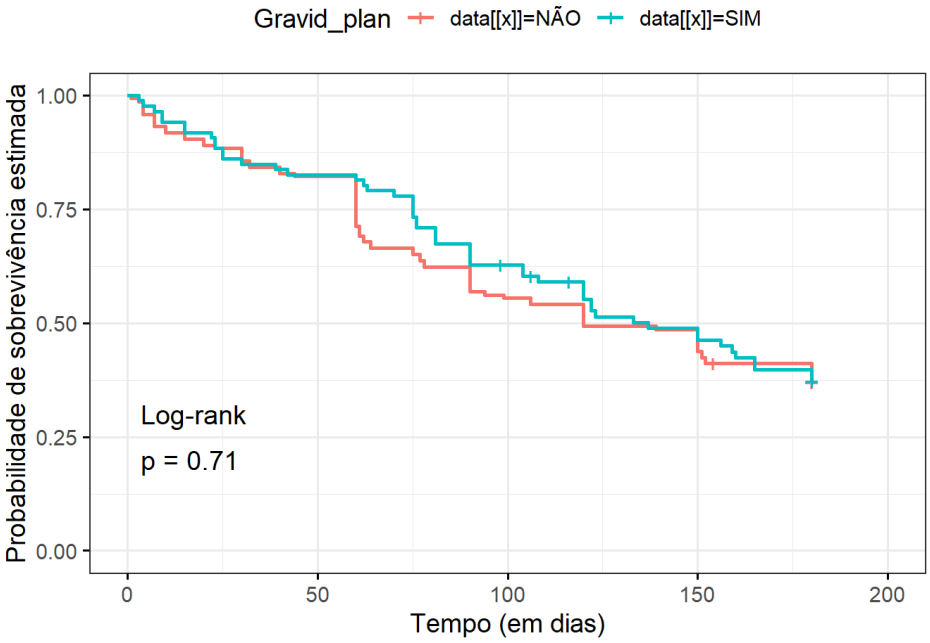
```
    legend.labs = levels(data[[x]]),
    legend.title = x,
    xlab = "Tempo (em dias)",
    ylab = "Probabilidade de sobrevivência estimada",
    ggtheme = theme_bw(base_size = 14)
  )

  logrank <- survdiff(Surv(data$Tempo_vida_ate_parada_amament_ate180, data$censura) ~
                        data[[x]])

  df_logrank_aux <- data.frame(
    variavel = x,
    logrank = round(logrank$chisq, 3),
    p_valor = round(logrank$pvalue, 3)
  )

  lista_de_plots[[i]] <- p

  df_logrank <- bind_rows(df_logrank, df_logrank_aux)
}

for(j in lista_de_plots){
  print(j)
}
```

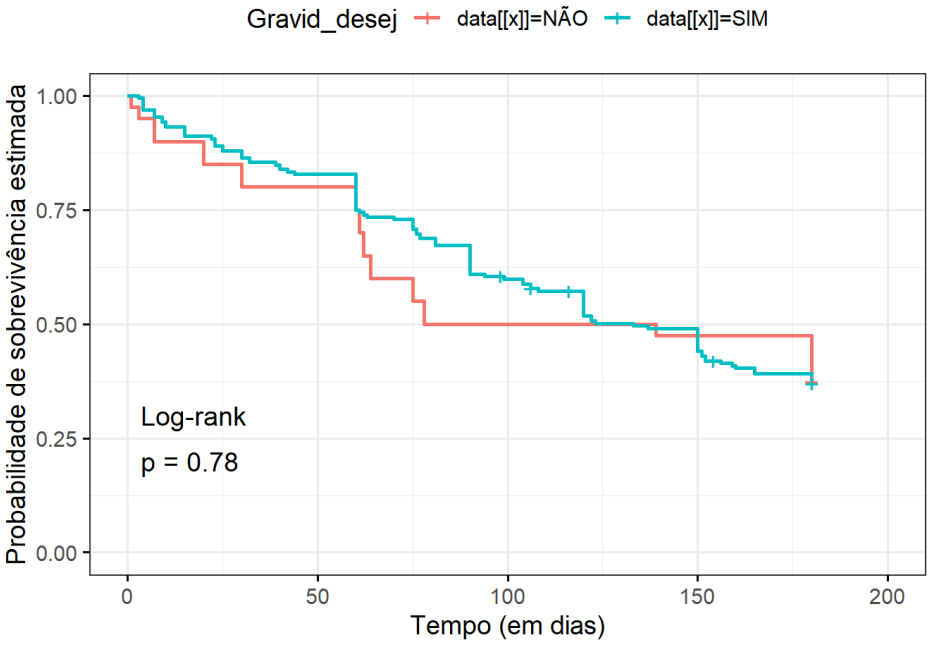


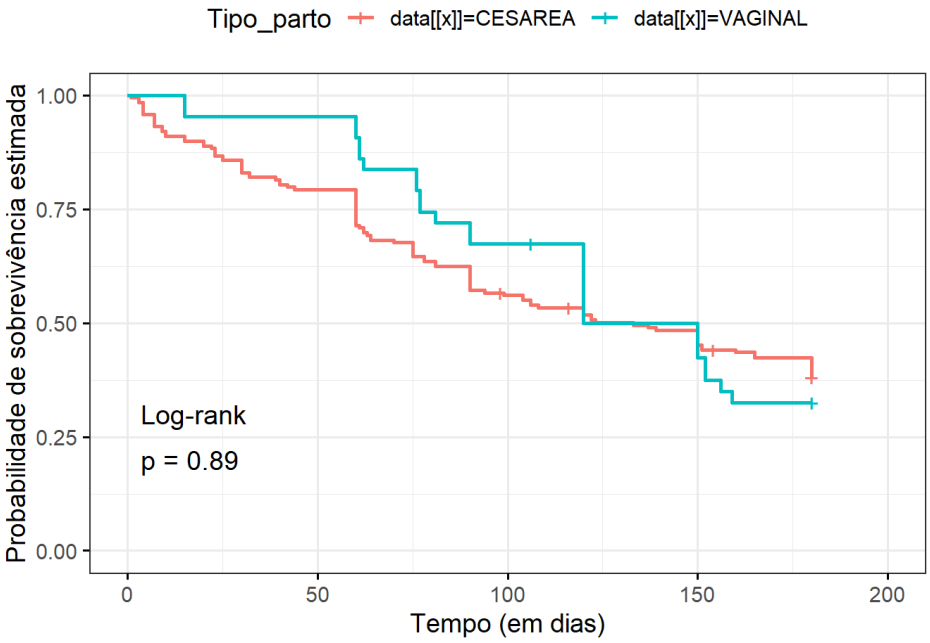

escolaridade_cat
<=8
9 a 11
>=12
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.14
0
50
100
150
200
Tempo (em dias)

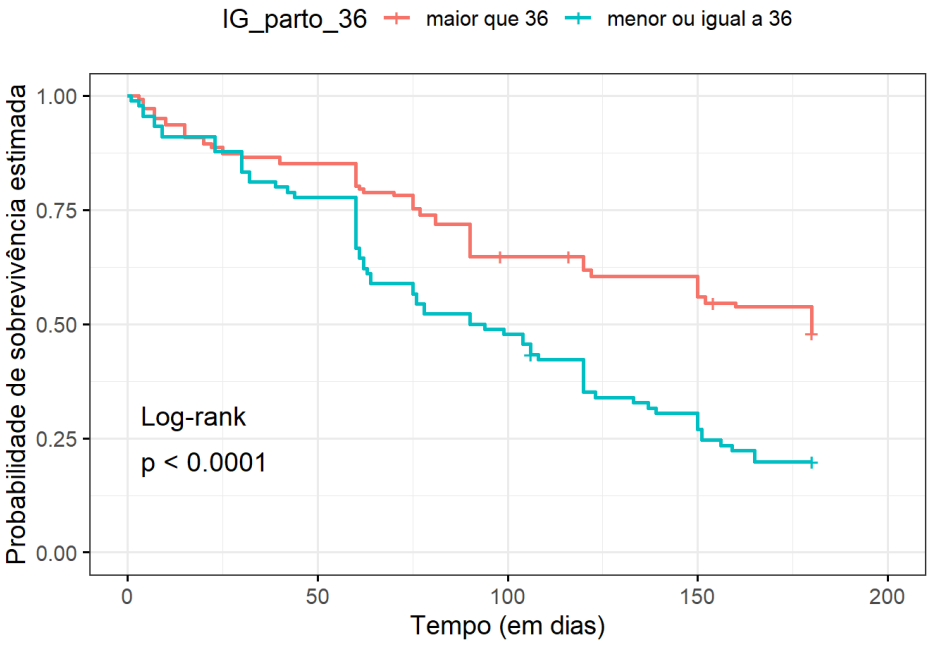

Horas_trab__menor_Igual7
maior que 7
menor ou igual a 7
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.067
0
50
100
150
200
Tempo (em dias)

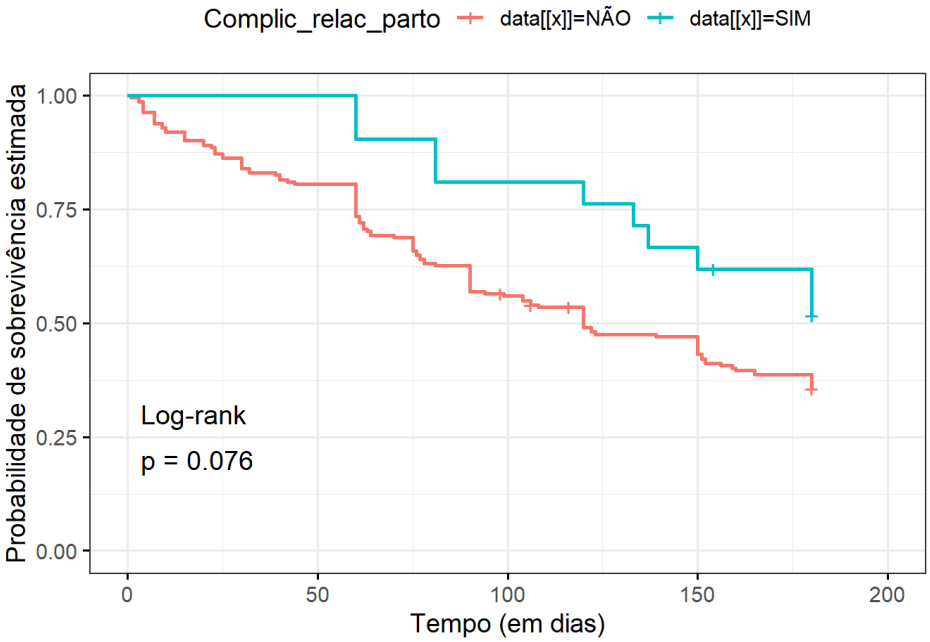

Mora_com_compan
data[[x]]=NÃO
data[[x]]=SIM
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.9
0
50
100
150
200
Tempo (em dias)

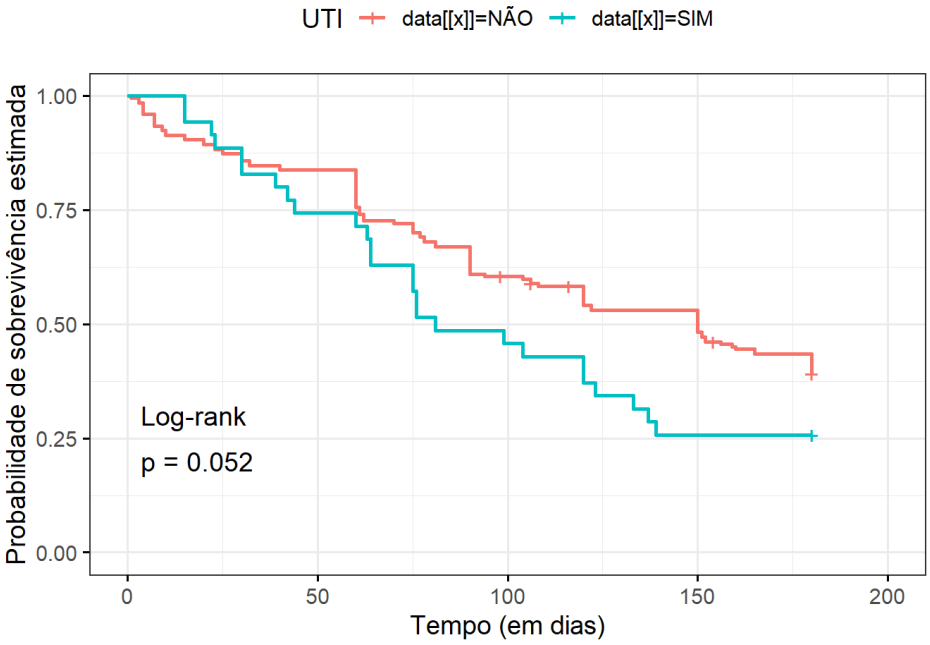

Gravid_plan
data[[x]]=NÃO
data[[x]]=SIM
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.71
0
50
100
150
200
Tempo (em dias)

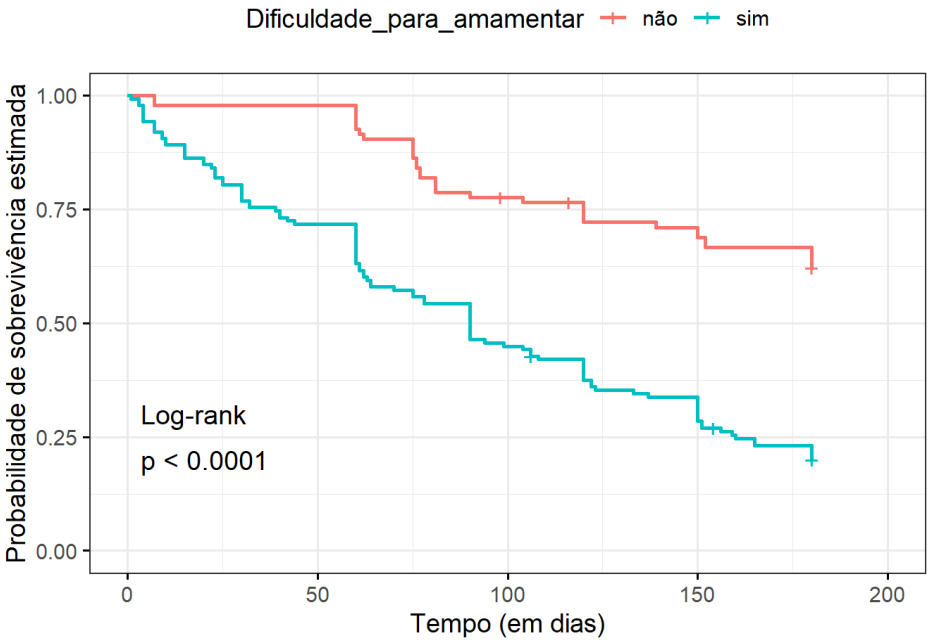

Gravid_desej
data[[x]]=NÃO
data[[x]]=SIM
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.78
0
50
100
150
200
Tempo (em dias)

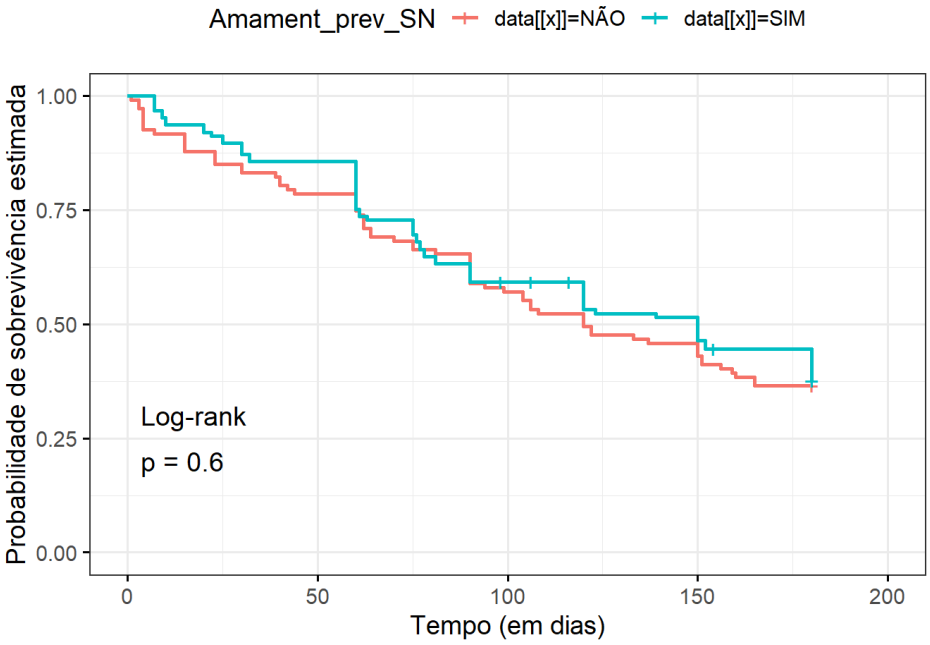

Tipo_parto
data[[x]]=CESAREA
data[[x]]=VAGINAL
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.89
0
50
100
150
200
Tempo (em dias)

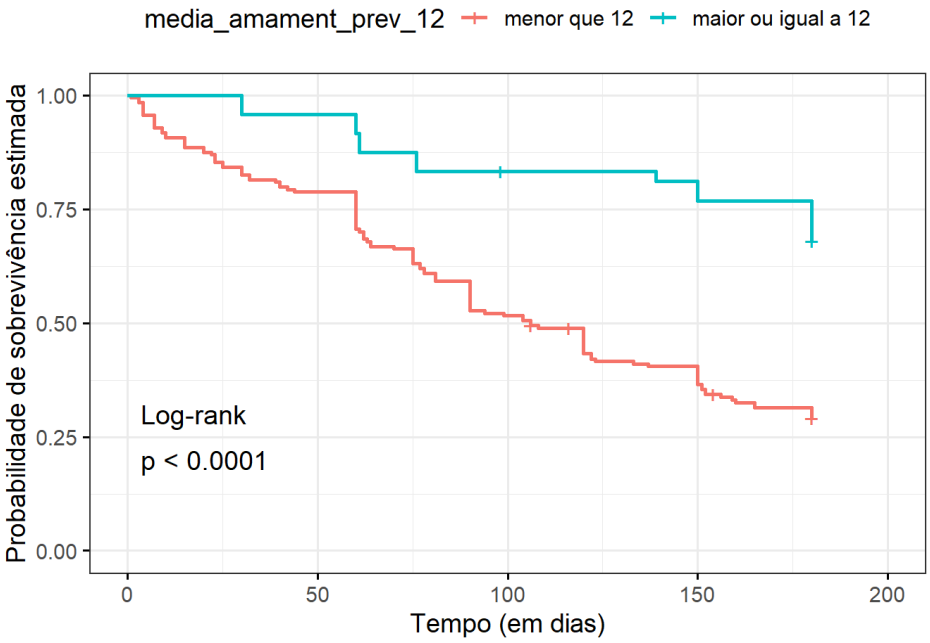

IG_parto_36
maior que 36
menor ou igual a 36
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p < 0.0001
0
50
100
150
200
Tempo (em dias)

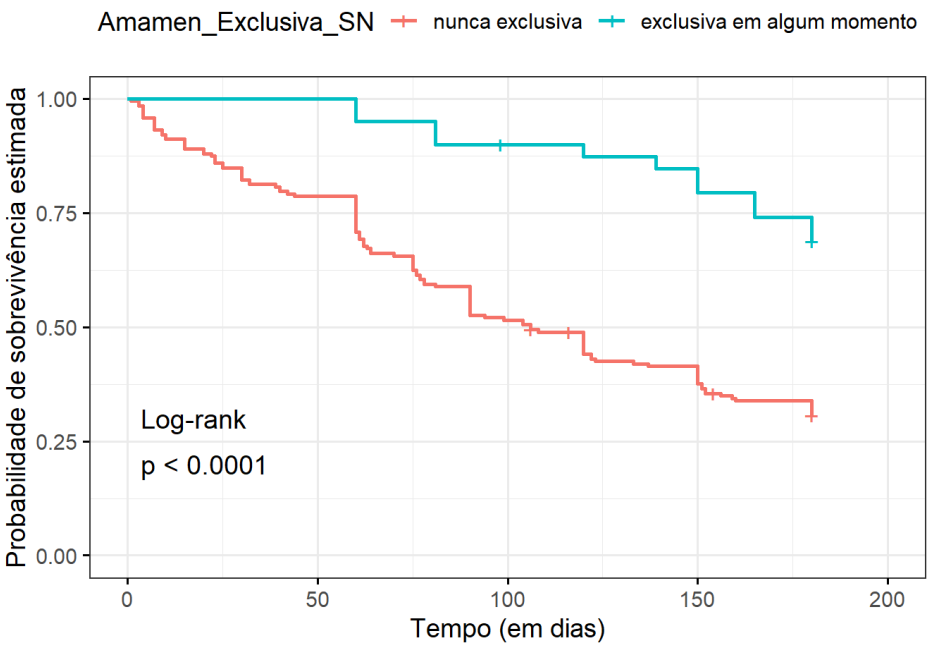

Complic_relac_parto
data[[x]]=NÃO
data[[x]]=SIM
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.076
0
50
100
150
200
Tempo (em dias)


UTI
data[[x]]=NÃO
data[[x]]=SIM
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.052
0
50
100
150
200
Tempo (em dias)


Dificuldade_para_amamentar
não
sim
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p < 0.0001
0
50
100
150
200
Tempo (em dias)


Amament_prev_SN
data[[x]]=NÃO
data[[x]]=SIM
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.6
0
50
100
150
200
Tempo (em dias)


media_amament_prev_12
menor que 12
maior ou igual a 12
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p < 0.0001
0
50
100
150
200
Tempo (em dias)



Em seguida, vamos usar a função `coxph` para ajustar um modelo de riscos proporcionais de Cox. O objeto `Surv` especifica a variável dependente de sobrevivência, com o tempo até o evento (Tempo_vida_ate_parada_amament_ate180) e a variável indicadora de censura (censura). Já `predict` faz predições com base no modelo ajustado, retornando as estimativas do tempo de sobrevivência para os novos dados. Usando a função `concordance`, é possível calcular a concordância (ou índice C) entre o tempo real de sobrevivência e as predições; o valor 1 indica predições perfeitas, enquanto valores próximos de 0.5 predições aleatórias. O valor calculado por `1 - concordance` indica a taxa de discordância das predições. Uma menor taxa indica um melhor ajuste do modelo, abaixo podemos observar o valor retornado nesse caso.

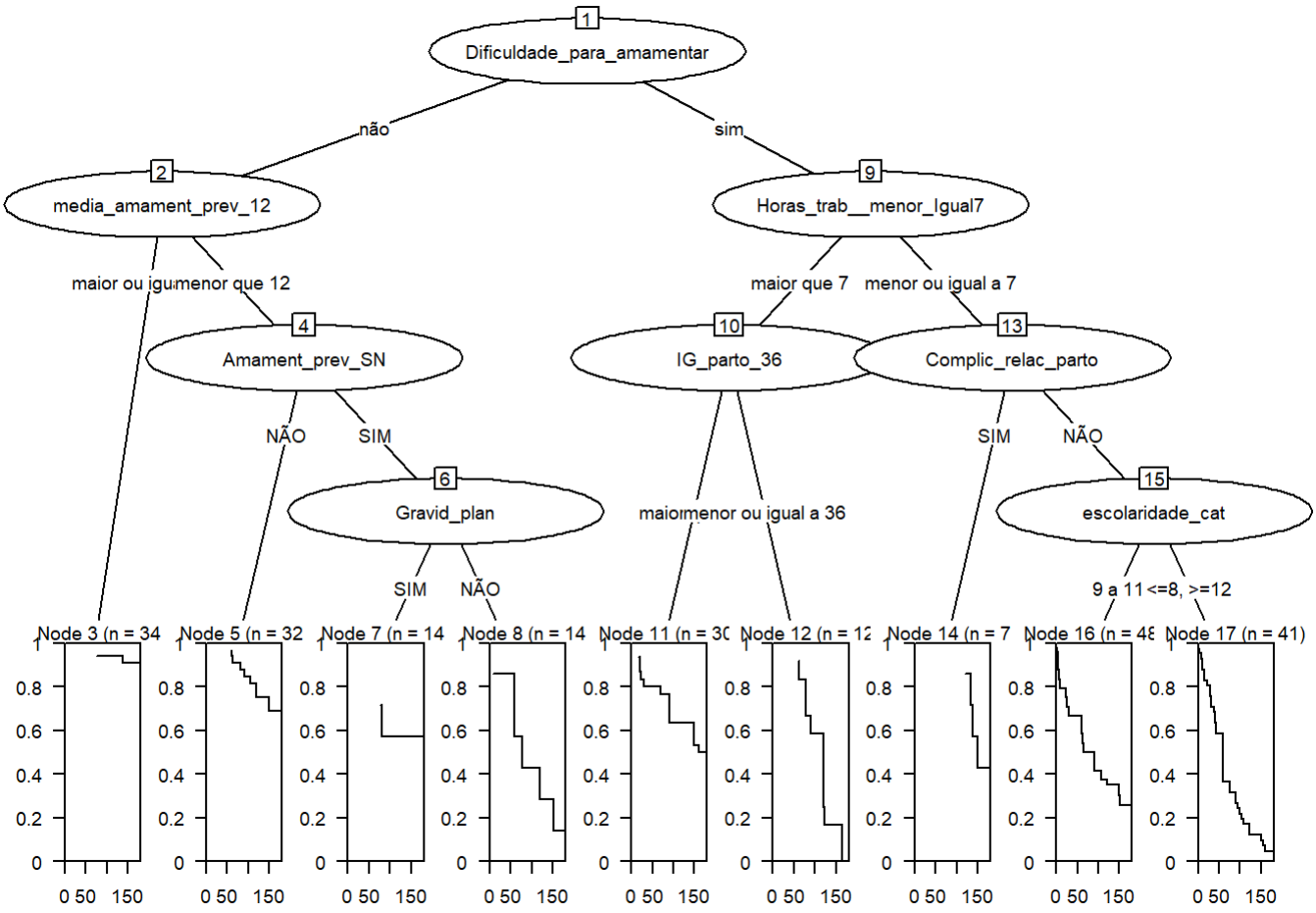
```
### NAIVE COX
mod.cox <- coxph(Surv(Tempo_vida_ate_parada_amament_ate180, censura) ~ .,data=data)
surv_time_hat <- predict(mod.cox, newdata=data)

survConcordance(Surv(Tempo_vida_ate_parada_amament_ate180, censura) ~ surv_time_hat, data=data
```

```
concordant
 0.7267457
```

No trecho a seguir, construiremos uma árvore de decisão, que divide os dados em grupos com base em suas características, modelando a sobrevivência dentro de cada grupo. A função de risco é calculada separadamente para cada nó terminal da árvore, como podemos ver abaixo. Aqui, `rpart` é usada para ajustar uma árvore de decisão para dados de sobrevivência, enquanto `rpart.plot` plota a árvore de decisão ajustada. Mais uma vez, Usando a função `concordance`, é possível calcular a concordância (ou índice C) entre o tempo real de sobrevivência e as predições.

```
### Árvore de sobrevivência
mod.sdt <- rpart(Surv(Tempo_vida_ate_parada_amament_ate180, censura) ~ .,
                 data=data)

surv_time_hat2 <- predict(mod.sdt, newdata=data)
```

```
plot(as.party(mod.sdt), gp = gpar(fontsize = 7))
```




```
survConcordance(Surv(Tempo_vida_ate_parada_amament_ate180, censura) ~ surv_time_hat2, data=dat
```

```
concordant
 0.7318263
```

A seguir, para empregar o modelo RSF, vamos dividir os dados em dois subconjuntos: treinamento (80%) e teste (20%). Isso permite utilizar uma parte dos dados para construir o modelo e avaliar sua performance utilizando a outra parte. Para dividir os dados, foi usada a dunção `createDataPartition`, que realiza a partição dos dados de forma a manter proporção de censuras em ambos os conjuntos.

```
# Criando amostras de treinamento e teste
set.seed(2402)
linhas_treino_rsf <- createDataPartition(
  data$censura,
  p = 0.8,
  list = FALSE,
  times = 1
)

dados_treino_rsf <- data[linhas_treino_rsf, ]
dados_teste_rsf <- data[-linhas_treino_rsf, ]
```

No próximo trecho, vamos usar a função `tune`, do pacote `randomForestSRC` para realizar a busca pelos melhores hiperparâmetros para o modelo de Random Survival Forest. O resultado `mtry` indica o número de variáveis a serem consideradas em cada divisão da árvore e `nodesize` o tamanho mínimo dos nós terminais.

```
# Transformar todas as colunas em factor, exceto as duas últimas
data[ , 1:(ncol(data)-2)] <- lapply(data[ , 1:(ncol(data)-2)], as.factor)

data <- as.data.frame(data)

# Tunando os hiperparâmetros do modelo
set.seed(2402)
fit_tune_rsf <- randomForestSRC::tune(
  Surv(Tempo_vida_ate_parada_amament_ate180, censura) ~ .,
  data = data,
  splitrule = "logrank",
  nsplit = 1
)

fit_tune_rsf$optimal
```

```
nodesize     mtry
       1       14
```

Agora, iremos ajustar o modelo de Random Survival Forest com os hiperparâmetros selecionados na etapa anterior, usando a função `rfsrc`, em que o argumento `splitrule = "logrank"` indica que a divisão das árvores será feita com base na estatística Log-rank.

```
# Ajustando o modelo com os hiperparâmetros selecionados
set.seed(2402)
fit_rsf <- rfsrc(
  Surv(Tempo_vida_ate_parada_amament_ate180, censura) ~ .,
  data = data,
  splitrule = "logrank",
  nsplit = 1,
  mtry = fit_tune_rsf$optimal[2],
  nodesize = fit_tune_rsf$optimal[1]
)

fit_rsf
```

```
                         Sample size: 232
                    Number of deaths: 144
                     Number of trees: 500
          Forest terminal node size: 1
       Average no. of terminal nodes: 79.696
No. of variables tried at each split: 14
              Total no. of variables: 14
       Resampling used to grow trees: swor
    Resample size used to grow trees: 147
                            Analysis: RSF
                              Family: surv
```

```
                     Splitting rule: logrank *random*
      Number of random split points: 1
                        (OOB) CRPS: 10.20172791
                  (OOB) stand. CRPS: 0.05667627
   (OOB) Requested performance error: 0.12529145
```

Acima observamos, que para a amostra de treinamento, o erro de predição OOB foi de 12,5%.

Para avaliar o desempenho preditivo do modelo ajustado vamos usar a função `predict`, que realiza a predição para os dados de teste com base no modelo RSF ajustado. Também podemos obervar o índice C, apresentado logo abaixo.

```r
# Verificando o desempenho preditivo do modelo
set.seed(2402)
pred_fit_rsf <- predict(fit_rsf, dados_teste_rsf, importance = "permute")

pred_fit_rsf
```

```
  Sample size of test (predict) data: 46
               Number of grow trees: 500
  Average no. of grow terminal nodes: 79.696
         Total no. of grow variables: 14
      Resampling used to grow trees: swor
    Resample size used to grow trees: 147
                            Analysis: RSF
                              Family: surv
                                CRPS: 33.18605657
                          stand. CRPS: 0.18436698
         Requested performance error: 0.48866213
```

```r
surv_time_hat3 <- as.numeric(pred_fit_rsf$predicted)

survConcordance(Surv(Tempo_vida_ate_parada_amament_ate180, censura) ~ surv_time_hat3, data=dad
```

```
concordant
       0.5
```

Com base nesse resultado, podemos verificar que o erro de predição obtido foi de 48,8%.

A seguir, vamos utilizar Support Vector Machines (SVM) para realizar uma regressão de sobrevivência, e depois avaliaremos a performance do modelo usando o índice de C. A função `survivalsvm` aplica SVM a dados de sobrevivência, enquanto `predict` faz previsões a partir do modelo ajustado e `concordance` é utilizada para calcular o índice de concordância, mostrado abaixo.

```r
mod.svcr <- survivalsvm(Surv(Tempo_vida_ate_parada_amament_ate180, censura) ~ .,
                        data, type="regression",
                        kernel="rbf_kernel",
                        gamma.mu = 1)

preds <- predict(mod.svcr, newdata=data)
surv_time_hat <- as.numeric(preds$predicted)
```

```
1-survConcordance(Surv(Tempo_vida_ate_parada_amament_ate180, censura) ~ surv_time_hat, data=da
```

```
concordant
 0.8381155
```

5.3 Previsão da próxima compra de clientes de uma indústria

Esse conjunto de dados é composto por informações de clientes de uma indústria em João Pessoa, Paraíba, referente ao seu histórico de compras, que está dividido em duas janelas, chamadas de janela comportamental e janela de predição. Na janela comportamental temos as informações das compras realizadas pelos clientes em determinado período de tempo (datas, frequência, valor, quantidade, dentre outras) e na janela de predição temos da data da última compra fora da janela comportamental para os clientes que realizaram uma compra. A janela está ilustrada na figura a seguir.

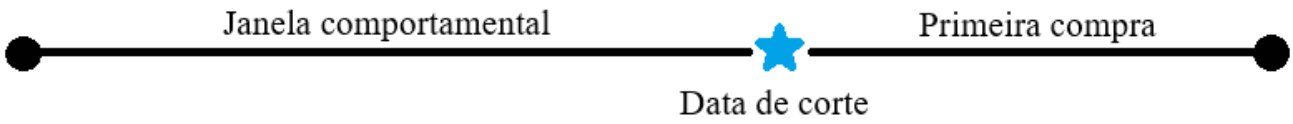


Ilustração da Janela Comportamental

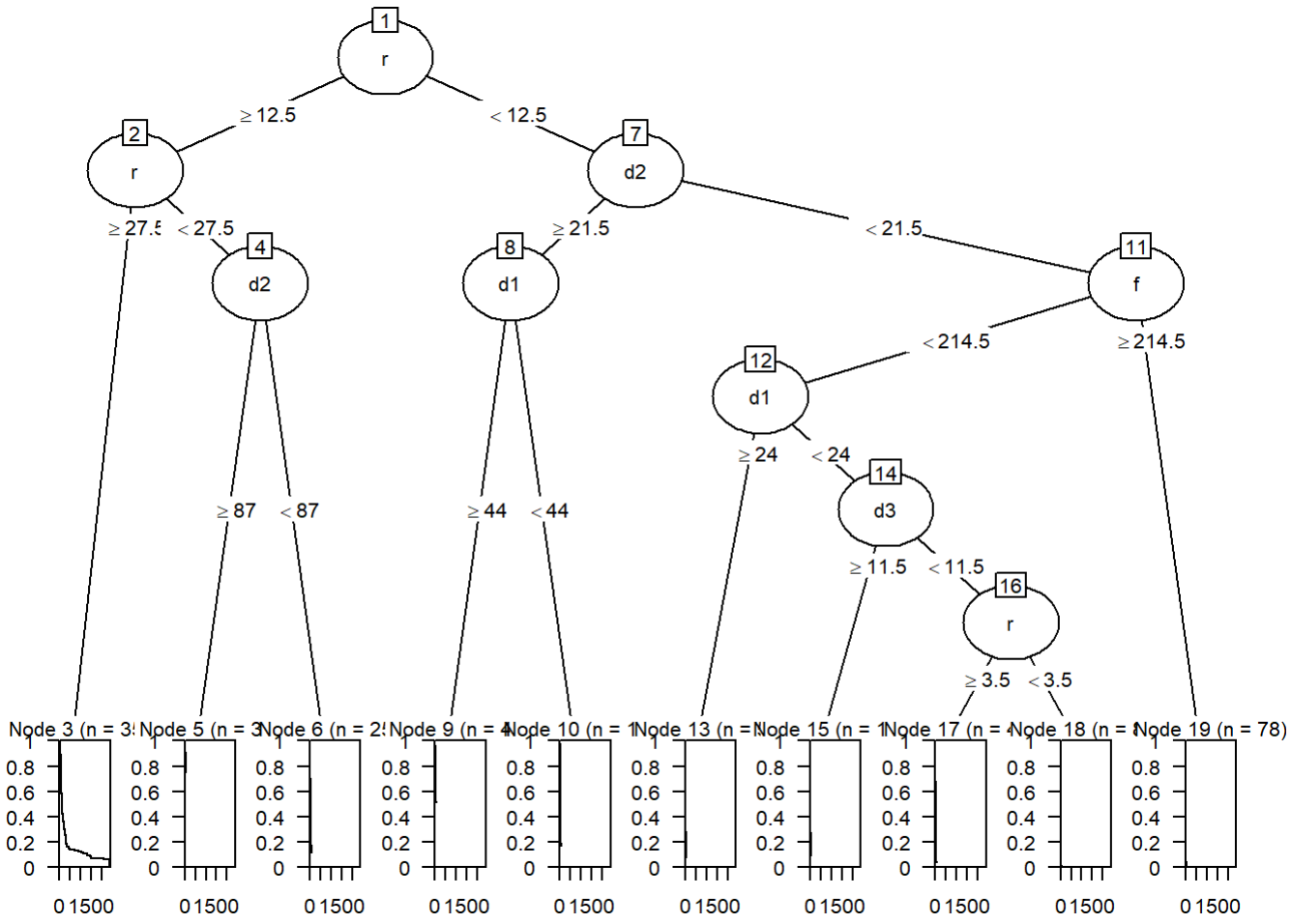
O interesse é estimar em quantos dias desde a última compra o cliente voltará a comprar, ou seja, o número de dias entre a última compra na janela comportamental e a primeira compra na janela de predição. O número de dias desde a última compra para clientes que não efetuaram compras fora da janela comportamental é definido como a data mais recente da janela de predição menos a data da última compra e esse tempo é considerado como um dado censurado. As variáveis disponíveis são: `idade_cliente` (a idade do cliente, em dias), `d1`, `d2`, `d3` (diferenças, em dias, entre as quatro últimas compras dentro da janela comportamental), `r` (Recência: índice que mede o nível de atividade dos clientes), `f` (Frequência: índice que quantifica o nível de fidelização dos clientes), `v` (Valor: índice que mede o valor monetário dos clientes para a empresa), `Y` (Tempo desde a última compra dentro da janela comportamental) e `censura` (Se os tempos são censurados ou não). Há um total de 1171 observações com 23% de censura.

```
#========== Packages =====================
require(readr)
require(rpart)
require(partykit)
require(survival)
require(survivalsvm)
require(randomForestSRC)

#========== Dataset =====================
dados <- read_csv("codes/data/next_buy_prediction.csv")

### NAIVE COX
mod.cox <- coxph(Surv(Y, censura) ~ ., data = dados)
surv_time_hat <- predict(mod.cox, newdata = dados)
```

```
survConcordance(Surv(Y, censura) ~ surv_time_hat,
                data=dados)$concordance
```

```
concordant
 0.8266966
```

```
### Árvores de Sobrevivência
mod.sdt <- rpart(Surv(Y, censura) ~ ., data = dados)

plot(as.party(mod.sdt), gp = gpar(fontsize = 8))
```




```
preds <- predict(mod.sdt, newdata = dados)
surv_time_hat <- as.numeric(preds)

survConcordance(Surv(Y, censura) ~ surv_time_hat,
                data=dados)$concordance
```

```
concordant
 0.8190154
```

```
### Floresta Aleatória de Sobrevivência - RSF

mod.rsf <- rfsrc(Surv(Y, censura) ~ .,
                 data = dados)
```

```r
preds <- predict(mod.rsf, newdata = dados)
surv_time_hat <- as.numeric(preds$predicted)

survConcordance(Surv(Y, censura) ~ surv_time_hat,
                data=dados)$concordance
```

```
concordant
 0.8743249
```

```r
## Regressão censurada por vetores de suporte - SVCR
mod.svcr <- survivalsvm(Surv(Y, censura) ~ .,
                        dados, type = "regression",
                        kernel = "rbf_kernel",
                        gamma.mu = 1)

preds <- predict(mod.svcr, newdata = dados)$predicted
surv_time_hat <- as.numeric(preds)

1-survConcordance(Surv(Y, censura) ~ surv_time_hat,
                  data=dados)$concordance
```

```
concordant
  0.799914
```

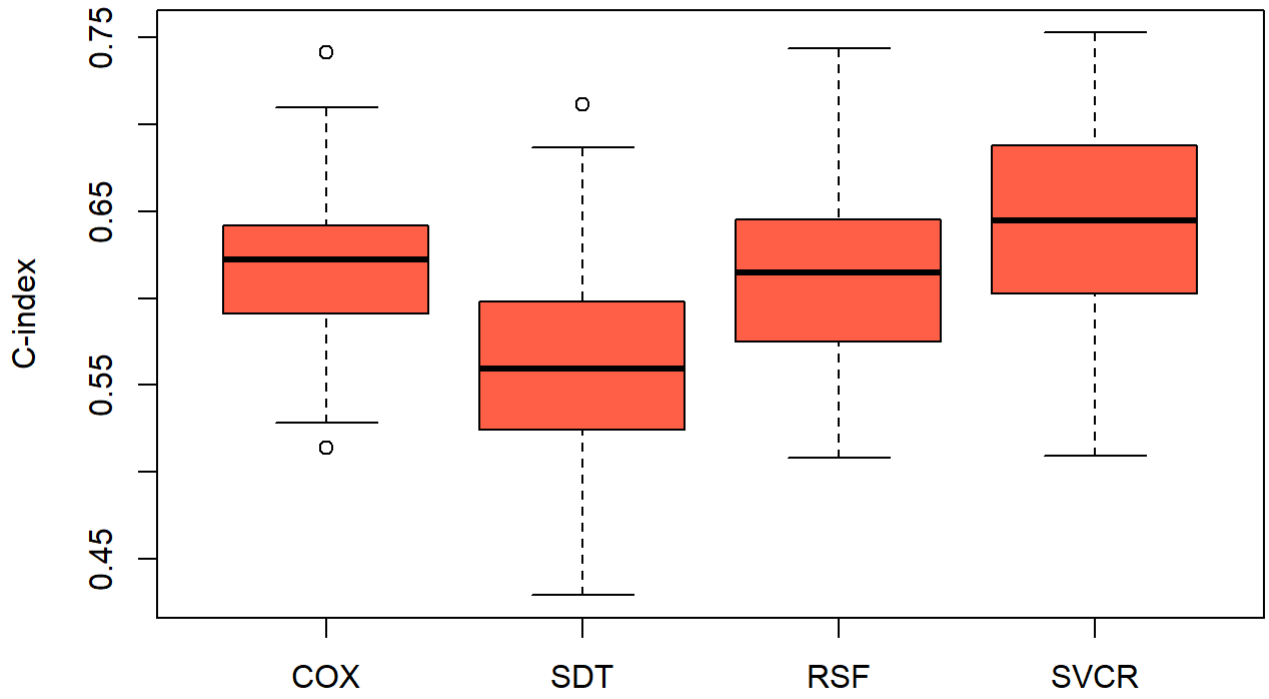
5.4 Dados do câncer de pulmão

Estes dados são referêntes a sobrevivência em pacientes com câncer de pulmão avançado do North Central Cancer Treatment Group (Loprinzi et al. 1994). Conjunto de dados `lung`, disponível no pacote `survival`. As variáveis consideradas englobam time: Tempo de sobrevivência em dias; status: status de censura 1=censurado, 2=morte; age: Idade em anos; sex: Masculino=1 Feminino=2; ph.ecog: Pontuação de desempenho ECOG conforme classificação do médico. 0=assintomático, 1= sintomático, mas completamente ambulatorial, 2= na cama <50% do dia, 3= na cama > 50% do dia, mas não acamado, 4 = acamado; ph.karno: Pontuação de desempenho de Karnofsky (ruim=0-bom=100) classificada pelo médico; pat.karno: Pontuação de desempenho de Karnofsky conforme classificação do paciente; meal.cal: Calorias consumidas nas refeições; wt.loss: Perda de peso nos últimos seis meses.

```r
#========== Packages =====================
require(readxl)
require(rpart)
require(survival)
require(survivalsvm)
require(randomForestSRC)

#========== Dataset =====================

data(lung)

lung$status <- lung$status -1
lung <- na.omit(lung)
```

```r
n <- nrow(lung)

dados=lung[,-1]


#=========== Holdout  Repetido ============

COX <- SDT <- RSF <- SVCR <- numeric(0)


for (i in 1:50) {
  set.seed(i)

  trein.index <- sample(1:n, 0.75 * n)
  teste.index <- setdiff(1:n, trein.index)

  dados.trein <- dados[trein.index,]
  dados.teste <- dados[teste.index,]

  ### NAIVE COX
  mod.cox <- coxph(Surv(time, status) ~ .,data=dados.trein)
  surv_time_hat <- predict(mod.cox, newdata=dados.teste)

  COX[i] <- survConcordance(Surv(time, status) ~ surv_time_hat,
                            data=dados.teste)$concordance


  ### SDT
  mod.sdt <- rpart(Surv(time, status) ~ .,
                   data=dados.trein)

  preds <- predict(mod.sdt, newdata=dados.teste)
  surv_time_hat <- as.numeric(preds)

  SDT[i] <- survConcordance(Surv(time, status) ~ surv_time_hat,
                            data=dados.teste)$concordance

  ### RSF

  mod.rsf <- rfsrc(Surv(time, status) ~ .,
                   data=dados.trein)

  preds <- predict(mod.rsf, newdata=dados.teste)
  surv_time_hat <- as.numeric(preds$predicted)

  RSF[i] <- survConcordance(Surv(time, status) ~ surv_time_hat,
                            data=dados.teste)$concordance

  ## SVCR
  mod.svcr <- survivalsvm(Surv(time, status) ~ .,
                          dados.trein, type="regression",
                          kernel="add_kernel",
                          gamma.mu = 1)
```

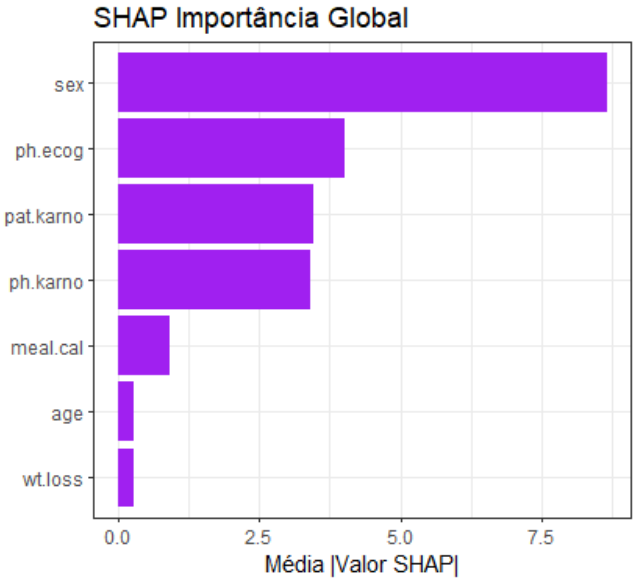
```
  preds <- predict(mod.svcr, newdata=dados.teste)$predicted
  surv_time_hat <- as.numeric(preds)

  SVCR[i] <- 1- survConcordance(Surv(time, status) ~ surv_time_hat,
                                data=dados.teste)$concordance
}

res <- data.frame(COX,SDT,RSF,SVCR)
boxplot(res,ylab="C-index",col="tomato")
```

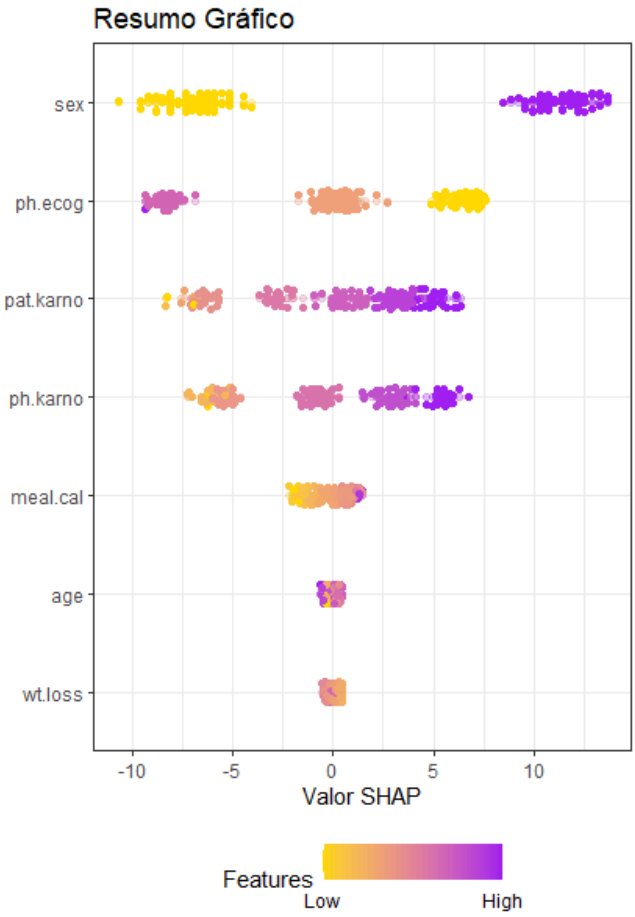



```
apply(res,2,mean)
```

```
      COX       SDT       RSF      SVCR
0.6168506 0.5659721 0.6140366 0.6395954
```

Uma vez que o método SVCR é um método de regressão censurada e, para este caso, foi com maior capacidade preditiva, é possível interpretar a influência das variáveis através de métodos de inteligência artificial explicatável como o SHAP Values (Lundberg e Lee 2017), por meio do pacote `shapviz` (Mayer 2023).





5.5 Pacientes com câncer do colo do útero

Para esta aplicação, o conjunto de dados utilizado é proveniente da base de dados do Registro Hospitalar de Câncer (RHC) da Fundação Oncocentro de São Paulo, a qual reúne informações de casos de câncer diagnosticados desde janeiro de 2000 no estado de São Paulo. Mais especificamente, para este exemplo, ajustaremos uma floresta aleatória de sobrevivência para analisar o tempo de sobrevivência de 13.214 pacientes com câncer do colo do útero registradas no período de 2000 a 2019. Utilizaremos como variável resposta a variável `tempo_dias`, que representa o tempo, em dias, entre o diagnóstico e o último follow-up

da paciente, e como variável indicadora de falha a variável `status`, que recebe 1 caso a paciente tenha morrido em decorrência do câncer do colo do útero e 0 caso essa informação seja censurada (seja porque a paciente estava viva até o último follow-up ou porque o acompanhamento parou de ser realizado por algum motivo que não seja a morte pelo câncer do colo do útero).

Ao longo deste exemplo, faremos uma breve análise exploratória dos dados, realizaremos o processo de tunagem dos hiperparâmetros do modelo, verificaremos o desempenho preditivo do modelo ajustado e utilizaremos um método de interpretabilidade para entendermos o impacto de cada variável nas predições do modelo obtido. Uma breve descrição das variáveis que serão utilizadas nesta aplicação podem ser vistas na tabela abaixo.

Descrição das variáveis do conjunto de dados de câncer do colo do útero utilizadas na aplicação.

Variável	Descrição
tempo_dias	Tempo, em dias, entre o diagnóstico e o último follow-up
status	Variável indicadora de falha (1 = falha; 0 = censura)
escolari	Código para escolaridade do paciente (1 = Analfabeto; 2 = Ens. Fund. incompleto; 3 = Ens. Fund. completo; 4 = Ensino médio; 5 = Superior, 9 = Sem informação)
idade_cat	Idade do paciente (0 a 49 anos; 50 a 74 anos; ≥ 75 anos)
cateatend	Categoria de atendimento ao diagnóstico (1 or 3 = Consórcio ou particular; 2 = SUS; 9 = Sem informação)
diagprev	Diagnóstico e tratamento anterior (1 = Sem diagnóstico / Sem tratamento; 2 = Com diagnóstico / Sem tratamento)
ecgrup	Grupo de estadiamento clínico (I, II, III, IV)
cirurgia	Tratamento recebido no hospital = cirurgia (0 = Não; 1 = Sim)
radio	Tratamento recebido no hospital = radioterapia (0 = Não; 1 = Sim)
quimio	Tratamento recebido no hospital = quimioterapia (0 = Não; 1 = Sim)
hormonio	Tratamento recebido no hospital = hormonioterapia (0 = Não; 1 = Sim)
outros	Tratamento recebido no hospital = outros (0 = Não; 1 = Sim)
tratcons_cat	Diferença, em dias, entre as datas de consulta e tratamento (≤ 60 dias e > 60 dias)
diagtrat_cat	Diferença, em dias, entra as datas de tratamento e diagnóstico
anodiag_cat	Ano de diagnóstico
recnenhum	Sem recidiva (0 = Não; 1 = Sim)

Antes de qualquer análise, alguns tratamentos no conjunto de dados considerado são necessários. No código abaixo, transformamos todas as variáveis categóricas em fatores - atribuindo os nomes das categorias para que a análise exploratória fique mais clara - e categorizamos duas das variáveis quantitativas do conjunto de dados: `idade`, transformando-a em uma variável categórica com as categorias "0 a 49 anos", "50 a 74 anos" e "75 anos ou mais"; e `tratcons`, que passará a conter as categorias "<= 60 dias" e "> 60 dias". Além disso, por meio da função `drop_na()`, excluímos do conjunto de dados 387 observações que possuíam alguma informação faltante.

```
#========== Packages =====================
library(dplyr)
library(janitor)
library(tidyr)
```

```r
library(survival)
library(survminer)
library(gridExtra)
library(caret)
library(randomForestSRC)

#========== Dataset ======================
dados_cancer_utero <- readRDS("codes/data/FOSP_cancer_colo_utero_recrutamento2019.rds") |>
  clean_names() |>
  mutate(
    cateatend = factor(case_when(
      cateatend == 1 | cateatend == 3 ~ "Convênio ou particular",
      cateatend == 2 ~ "SUS",
      cateatend == 9 ~ "Sem informação"
    ), levels = c("Convênio ou particular", "SUS", "Sem informação")),
    diagprev = factor(case_when(
      diagprev == 1 ~ "Sem diagnóstico e sem tratamento",
      diagprev == 2 ~ "Com diagnóstico e sem tratamento"
    ), levels = c("Sem diagnóstico e sem tratamento", "Com diagnóstico e sem tratamento")),
    ecgrup = factor(ecgrup),
    cirurgia = factor(case_when(
      cirurgia == 0 ~ "Não",
      cirurgia == 1 ~ "Sim"
    ), levels = c("Não", "Sim")),
    hormonio = factor(case_when(
      hormonio == 0 ~ "Não",
      hormonio == 1 ~ "Sim"
    ), levels = c("Não", "Sim")),
    quimio = factor(case_when(
      quimio == 0 ~ "Não",
      quimio == 1 ~ "Sim"
    ), levels = c("Não", "Sim")),
    radio = factor(case_when(
      radio == 0 ~ "Não",
      radio == 1 ~ "Sim"
    ), levels = c("Não", "Sim")),
    outros = factor(case_when(
      outros == 0 ~ "Não",
      outros == 1 ~ "Sim"
    ), levels = c("Não", "Sim")),
    recnenhum = factor(case_when(
      recnenhum == 0 ~ "Não",
      recnenhum == 1 ~ "Sim"
    ), levels = c("Não", "Sim")),
    escolari = factor(case_when(
      escolari == 1 ~ "Analfabeto",
      escolari == 2 ~ "Ens. fund. incompleto",
      escolari == 3 ~ "Ens. fund. completo",
      escolari == 4 ~ "Ensino médio",
      escolari == 5 ~ "Superior",
      escolari == 9 ~ "Ignorada",
    ), levels = c("Analfabeto", "Ens. fund. incompleto", "Ens. fund. completo",
                  "Ensino médio", "Superior", "Ignorada")),
    idade_cat = factor(case_when(
```

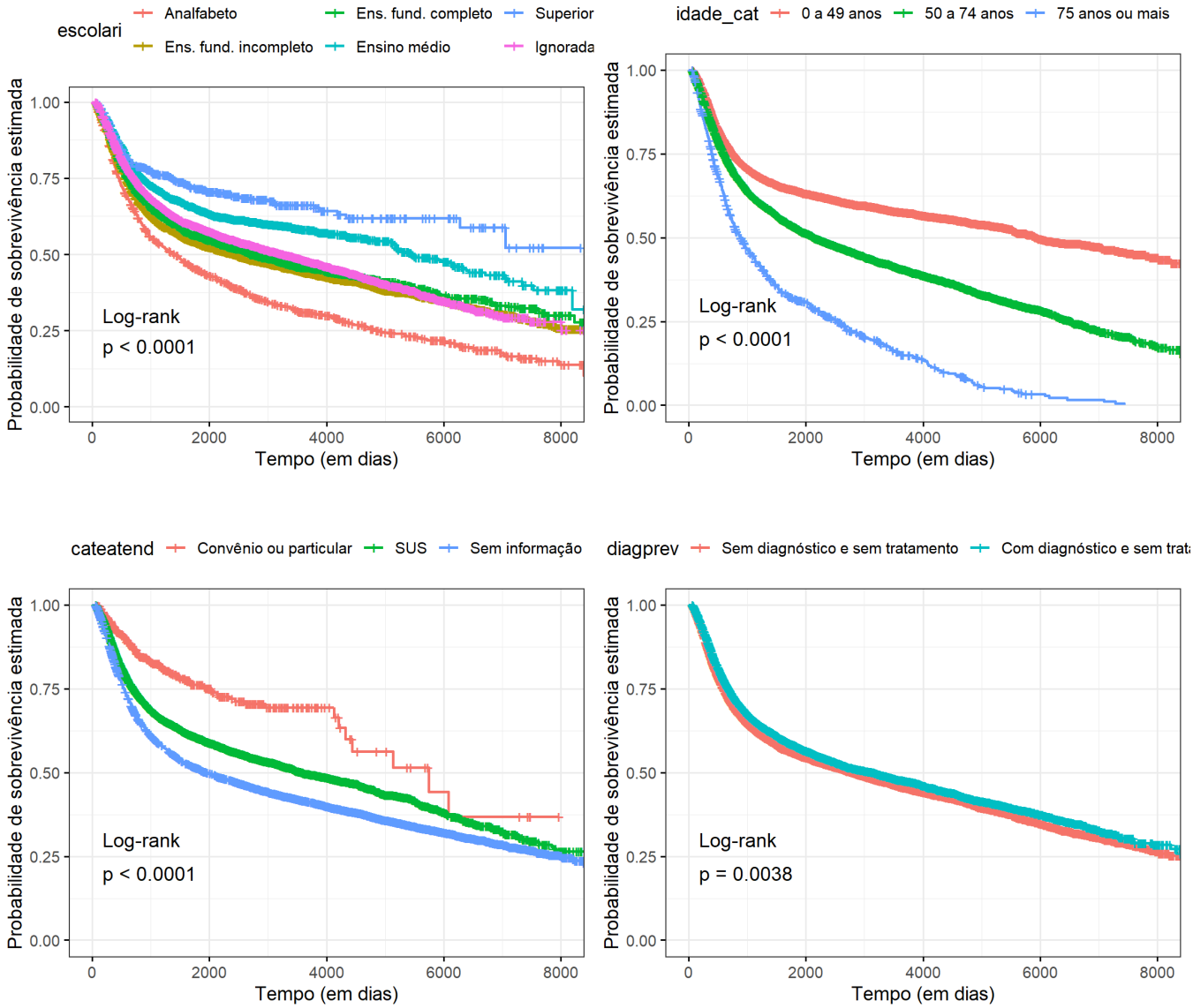
```
      idade <= 49 ~ "0 a 49 anos",
      idade >= 50 & idade <= 74 ~ "50 a 74 anos",
      idade >= 75 ~ "75 anos ou mais"
    ), levels = c("0 a 49 anos", "50 a 74 anos", "75 anos ou mais")),
    tratcons_cat = factor(case_when(
      tratcons <= 60 ~ "<= 60 dias",
      tratcons > 60 ~ "> 60 dias"
    ), levels = c("<= 60 dias", "> 60 dias"))
  ) |>
  select(
    tempo_dias, status, anodiag, cateatend, cirurgia,
    diagprev, diagtrat, ecgrup, escolari, hormonio, idade_cat, outros,
    quimio, radio, recnenhum, tratcons_cat
  ) |>
  drop_na()
```

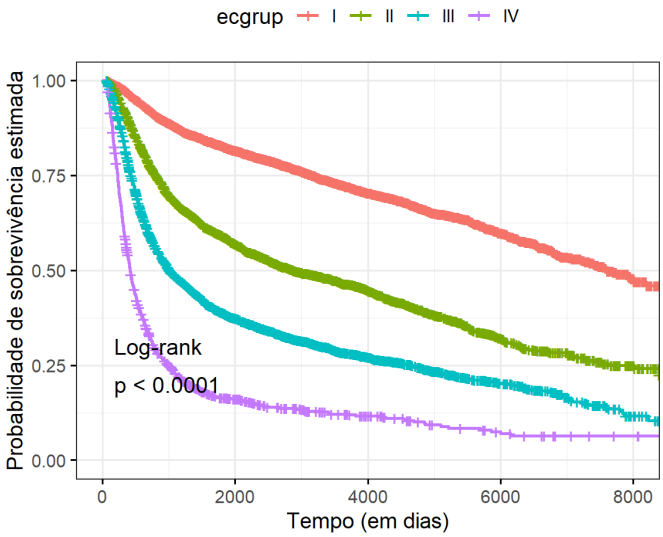
Quanto à análise exploratória, apresentamos, nas figuras abaixo, as curvas de sobrevivência estimadas via Kaplan-Meier e os resultados dos testes de logrank para cada covariável categórica do conjunto de dados. Para a obtenção dos gráficos e dos resultados dos testes, foram utilizadas, respectivamente, as funções `ggsurvplot()` e `survdiff()`, dos pacotes `survminer` e `survival`.

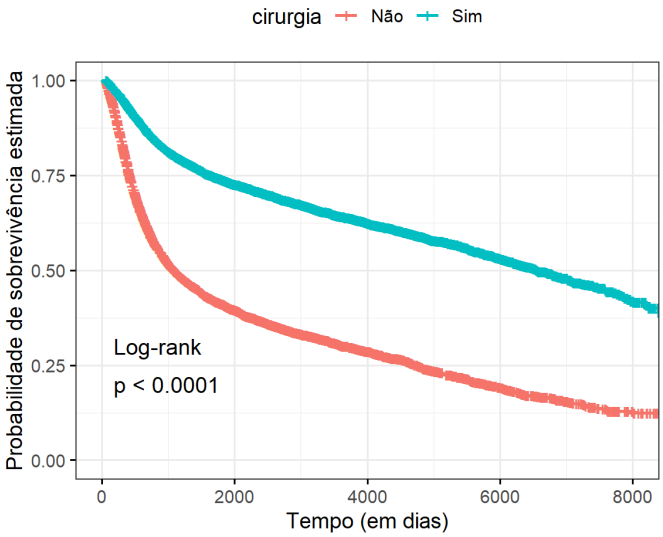
Como podemos notar, para todas as variáveis e adotando um nível de significância de 0,05, os resultados dos testes de logrank nos dão evidências para acreditar que ao menos uma das curvas estimadas difere de alguma das demais; ou seja, que indivíduos de pelo menos dois grupos distintos em cada variável possuem comportamentos de sobrevivência diferentes. São notáveis, especialmente, as diferenças entre as curvas de sobrevivência estimadas de pacientes nos diferentes grupos de idade, de pacientes nos diferentes estadiamentos clínicos e de pacientes que realizaram ou não cirurgia.

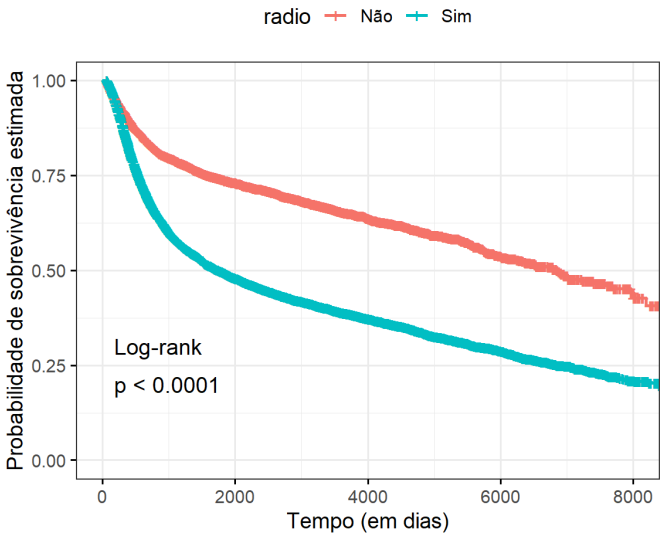
```
variaveis_categoricas <- c(
  "escolari", "idade_cat", "cateatend", "diagprev", "ecgrup", "cirurgia",
  "radio", "quimio", "hormonio", "outros", "tratcons_cat", "recnenhum"
)

lista_de_plots <- list()
df_logrank <- data.frame()

for (i in 1:length(variaveis_categoricas)) {
  variavel <- variaveis_categoricas[i]

  ekm <- survfit(Surv(dados_cancer_utero$tempo_dias, dados_cancer_utero$status) ~
                   dados_cancer_utero[[variavel]])

  p <- ggsurvplot(
    ekm,
    data = dados_cancer_utero,
    pval = TRUE,
    pval.method = TRUE,
    conf.int = FALSE,
    legend.labs = levels(dados_cancer_utero[[variavel]]),
    legend.title = variavel,
    xlab = "Tempo (em dias)",
    ylab = "Probabilidade de sobrevivência estimada",
    ggtheme = theme_bw(base_size = 14)
```

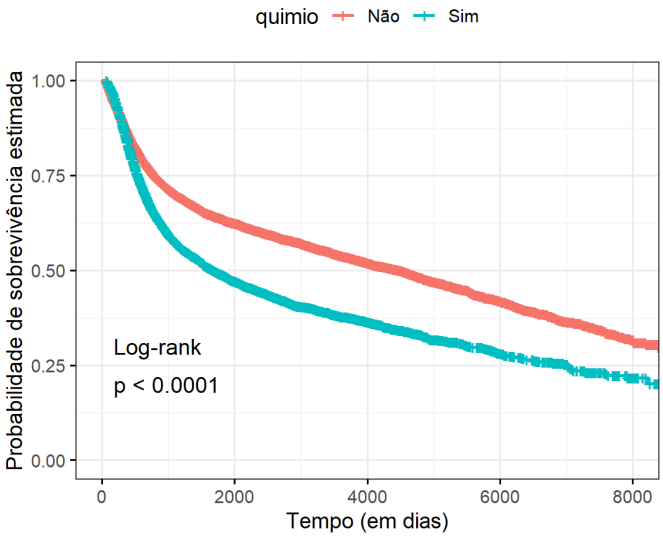
```r
  )

  logrank <- survdiff(Surv(dados_cancer_utero$tempo_dias, dados_cancer_utero$status) ~
                        dados_cancer_utero[[variavel]])

  df_logrank_aux <- data.frame(
    variavel = variavel,
    logrank = round(logrank$chisq, 3),
    p_valor = round(logrank$pvalue, 3)
  )

  lista_de_plots[[i]] <- p

  df_logrank <- bind_rows(df_logrank, df_logrank_aux)
}

arrange_ggsurvplots(lista_de_plots, ncol = 2)
```

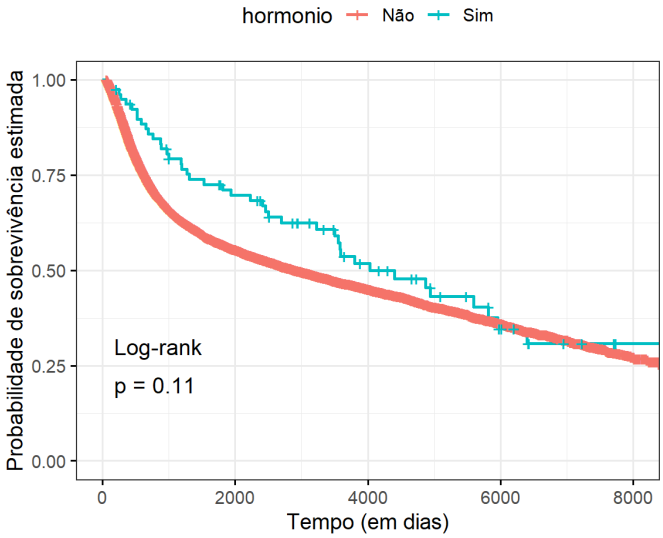


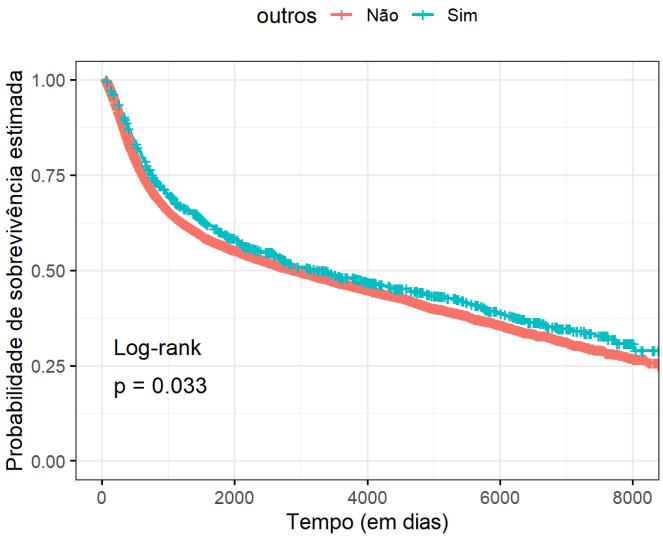

ecgrup I II III IV
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p < 0.0001
0
2000
4000
6000
8000
Tempo (em dias)

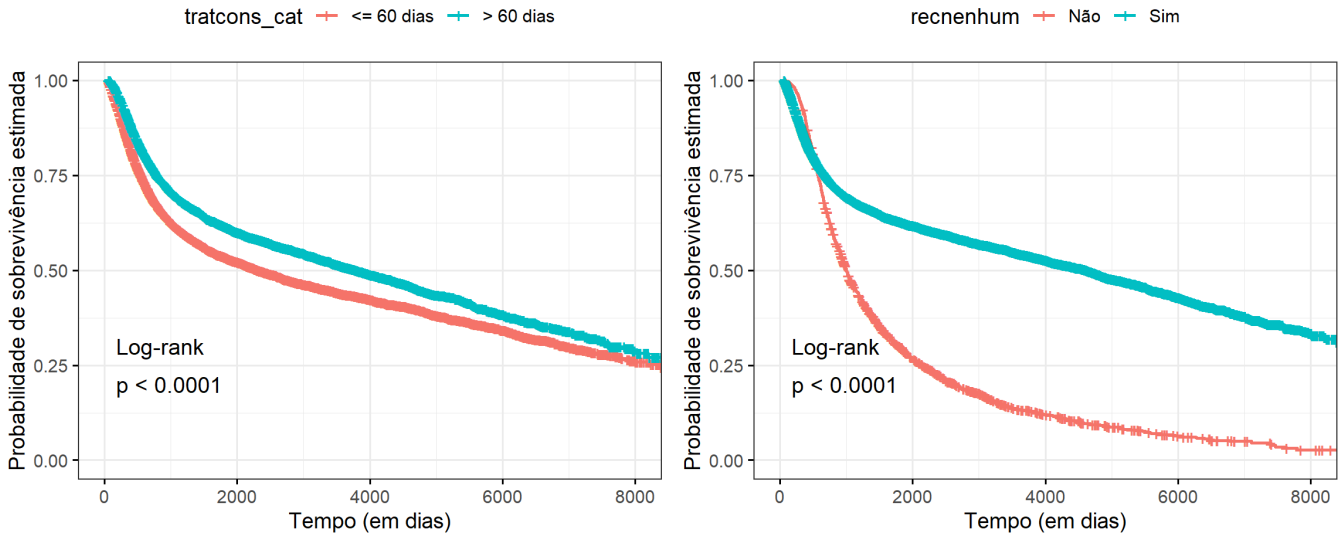

cirurgia Não Sim
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p < 0.0001
0
2000
4000
6000
8000
Tempo (em dias)


radio Não Sim
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p < 0.0001
0
2000
4000
6000
8000
Tempo (em dias)


quimio Não Sim
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p < 0.0001
0
2000
4000
6000
8000
Tempo (em dias)


hormonio Não Sim
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.11
0
2000
4000
6000
8000
Tempo (em dias)


outros Não Sim
Probabilidade de sobrevivência estimada
1.00
0.75
0.50
0.25
0.00
Log-rank
p = 0.033
0
2000
4000
6000
8000
Tempo (em dias)



Após a breve análise exploratória realizada, podemos iniciar os processos para o ajusto do modelo. Para evitar problemas relacionados à forma como as funções de ajuste tratam variáveis categóricas, iniciaremos transformando todas as variáveis categóricas do conjunto de dados em variáveis dummy. Assim, se uma variável categórica possuía k categorias, serão criadas $k - 1$ variáveis dummy, uma para cada categoria que não seja a categoria de referência da variável original. Para simplificar este exemplo, não definiremos categorias de referência especiais para cada variável, optando por utilizar as categorias que aparecem primeiro no argumento `levels`, da função `factor()`, que definimos quando estávamos transformando as variáveis do conjunto de dados em fatores.

```
# Substituindo as variáveis categóricas do conjunto de dados por variáveis dummy
dados_dummies_rsf <- cbind(
  dados_cancer_utero |> select(tempo_dias, status),
  model.matrix(~., data = dados_cancer_utero |> select(!c(tempo_dias, status))) |>
    as.data.frame() |>
    select(!`(Intercept)`)
)
```

A partir do novo conjunto de dados, criaremos um conjunto de treino, formado por 70% das observações, e um conjunto de teste, formado pelas 30% restantes. Utilizaremos o conjunto de treino para tunar e ajustar o modelo e o conjunto de teste para verificar seu desempenho preditivo. Para a realização dessa separação, podemos utilizar a função `createDataPartition()`, do pacote `caret`, que seleciona de maneira aleatória uma proporção das linhas do conjunto de dados original para compor o conjunto de treino.

```
# Criando amostras de treinamento e teste
set.seed(428)
linhas_treino_rsf <- createDataPartition(
  dados_dummies_rsf$status,
  p = 0.7,
  list = FALSE,
  times = 1
)

dados_treino_rsf <- dados_dummies_rsf[linhas_treino_rsf, ]
dados_teste_rsf <- dados_dummies_rsf[-linhas_treino_rsf, ]
```

Com o conjunto de treino em mãos, podemos partir para o processo de tunagem dos hiperparâmetros do modelo. No R, ajustes de florestas aleatórias de sobrevivência estão disponíveis dentro dos pacotes `randomForestSRC` e `ranger`. Apesar de ambos serem baseados na proposição original de Hemant Ishwaran et al. (2008), os dois pacotes diferem em algumas funcionalidades suportadas. Para este exemplo, utilizaremos o pacote `randomForestSRC`, construído especificamente com as florestas aleatórias de sobrevivência em mente e que, por consequência, possui funcionalidades mais completas para lidar com esse tipo de modelo. Uma dessas funcionalidades é, por exemplo, a existência da função `tune()`, que permite que realizemos a tunagem dos hiperparâmetros do modelo.

Por padrão, a função `tune()` realiza a tunagem dos hiperparâmetros "nodesize", que representa o número mínimo de observações nos nós terminais de cada árvore, e "mtry", que representa o número de covariáveis selecionadas aleatoriamente como candidatas para cada divisão de cada árvore. As florestas possuem um terceiro hiperparâmetro, "ntree" (referente ao número de árvores que irão compor a floresta), que por padrão não é tunado pela função `tune()`. Esse hiperparâmetro é fixado em 100 para o processo de tunagem e em 500 para o ajuste do modelo final.

Como podemos observar pela saída abaixo, os valores de nodesize e mtry que minimizam o erro de predição do modelo são, respectivamente, 8 e 18.

```r
# Tunando os hiperparâmetros do modelo
set.seed(428)
fit_tune_rsf <- randomForestSRC::tune(
  Surv(tempo_dias, status) ~ .,
  data = dados_treino_rsf,
  splitrule = "logrank",
  nsplit = 1
)
fit_tune_rsf$optimal
```

```
nodesize     mtry
       8       18

nodesize     mtry
       8       18
```

Algo interessante a se notar é que, tanto na função `tune()` quanto na função `rfsrc()` - que utilizaremos para fazer o ajuste do modelo com os hiperparâmetros selecionados -, aparecem outros dois argumentos interessantes: `splitrule` e `nsplit`. O argumento `splitrule` define a *splitting rule* a ser adotada, a qual, como comentamos quando definimos o algoritmo das florestas aleatórias de sobrevivência, controla o critério a ser utilizado para a divisão de cada nó da árvore. Por padrão, o pacote `randomForestSRC` utiliza a *splitting rule* logrank, por meio da qual é selecionado o valor da variável candidata que maximize a estatística de logrank entre os dois grupos formados pela divisão. O argumento `nsplit`, por outro lado, controla o número de pontos aleatórios a serem testados nas variáveis candidatas à divisão de cada nó. Quando esse argumento é 1, ajustamos o que são chamadas de *extremely random forests*, nas quais apenas um valor de cada variável é selecionado para ser testado durante a divisão dos nós.

Com esses pontos em mente, o código abaixo realiza o ajuste do modelo com os hiperparâmetros previamente selecionados. Como podemos observar, para a amostra de treinamento, o erro de predição OOB, dado por $1 - \text{C-Index}$, foi de 26%.

```r
# Ajustando o modelo com os hiperparâmetros selecionados
set.seed(428)
fit_rsf <- rfsrc(
  Surv(tempo_dias, status) ~ .,
  data = dados_treino_rsf,
  splitrule = "logrank",
  nsplit = 1,
  mtry = fit_tune_rsf$optimal[2],
  nodesize = fit_tune_rsf$optimal[1]
)

fit_rsf
```

```
                         Sample size: 9250
                    Number of deaths: 4839
                     Number of trees: 500
          Forest terminal node size: 8
       Average no. of terminal nodes: 715.052
No. of variables tried at each split: 18
              Total no. of variables: 22
       Resampling used to grow trees: swor
    Resample size used to grow trees: 5846
                            Analysis: RSF
                              Family: surv
                      Splitting rule: logrank *random*
       Number of random split points: 1
                           (OOB) CRPS: 1450.6248861
                  (OOB) stand. CRPS: 0.1729405
   (OOB) Requested performance error: 0.25987446
```

Ajustado o modelo, podemos verificar seu desempenho preditivo na amostra teste a partir da função `predict()`. Como podemos notar, o erro de predição obtido foi de 25,4%, indicando que o modelo consegue ordenar moderadamente bem os indivíduos da amostra teste com base em seus riscos estimados.

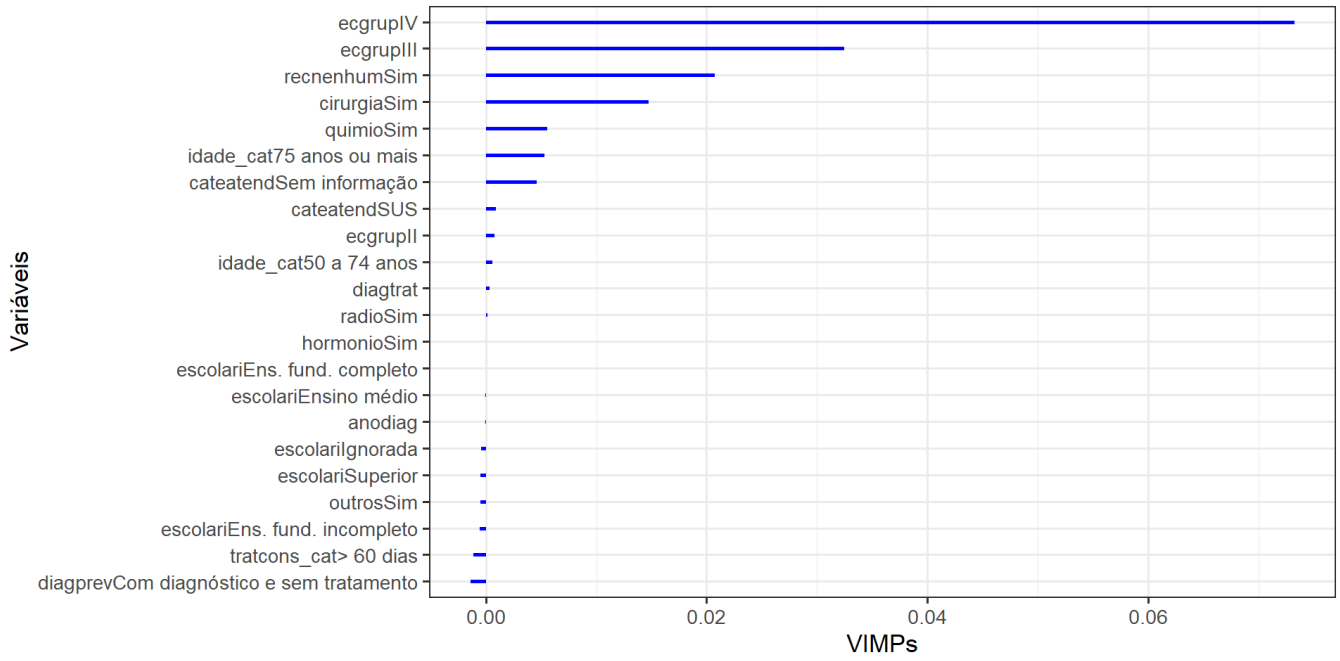
```r
# Verificando o desempenho preditivo do modelo
set.seed(428)
pred_fit_rsf <- predict(fit_rsf, dados_teste_rsf, importance = "permute")

pred_fit_rsf
```

```
  Sample size of test (predict) data: 3964
               Number of grow trees: 500
  Average no. of grow terminal nodes: 715.052
         Total no. of grow variables: 22
       Resampling used to grow trees: swor
    Resample size used to grow trees: 5846
                            Analysis: RSF
                              Family: surv
                                CRPS: 3525.39898531
                          stand. CRPS: 0.42029077
         Requested performance error: 0.25415572
```

Para finalizar, o pacote `randomForestSRC` possui a implementação de uma medida não paramétrica da importância das variáveis: o VIMP (*variable importance*). Dentro do pacote `randomForestSRC`, todas as funções capazes de calcular o VIMP - que incluem as funções `rfsrc()`, `predict()` e `vimp()` - possuem diferentes implementações dessa medida de importância. Para este exemplo, calculamos no código acima, por meio da função `predict()`, a importância via permutação, na qual o erro de predição atribuível a uma variável é estimado permutando aleatoriamente seus valores dentro da amostra OOB de uma árvore (para mais detalhes, ver H. Ishwaran et al. (2021)). Altos valores positivos de VIMP indicam variáveis com alta capacidade preditiva, enquanto valores negativos ou próximos de zero indicam variáveis que não importam ou atrapalham nas predições do modelo.

Como podemos notar através da figura abaixo, as informações que mais afetam positivamente as predições do modelo são estar no estadiamento clínico IV ou III, não ter apresentado nenhuma recidiva e ter feito cirurgia. Do outro lado do espectro, as variáveis de escolaridade, ano de diagnóstico e tempo entre o tratamento e a primeira consulta aparentam não importar ou até atrapalhar nas predições do modelo ajustado.

```r
# Criando um data.frame com os nomes das variáveis e seus VIMPs
df_vimps <- data.frame(
  var = names(sort(pred_fit_rsf$importance, decreasing = FALSE)),
  vimp = sort(as.numeric(pred_fit_rsf$importance), decreasing = FALSE)
)
df_vimps$var <- factor(df_vimps$var, levels = df_vimps$var)

# Plotando os VIMPs do modelo
ggplot(df_vimps, aes(y = var)) +
  geom_segment(aes(x = 0, xend = vimp, yend = var), color = "blue", linewidth = 1) +
  labs(x = "VIMPs", y = "Variáveis") +
  theme_bw(base_size = 13)
```



Referências

Ambrogi, Federico, e Thomas H Scheike. 2016. «Penalized estimation for competing risks regression with applications to high-dimensional covariates». *Biostatistics* 17 (4): 708–21.

Benda, Brent B. 2003. «Survival analysis of criminal recidivism of boot camp graduates using elements from general and developmental explanatory models». *International Journal of Offender Therapy and Comparative Criminology* 47 (1): 89–110.

Bishop, Christopher M. 2007. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1.ª ed. Springer.

Bou-Hamad, Imad, Denis Larocque, e Hatem Ben-Ameur. 2011. «A review of survival trees».

Box-Steffensmeier, Janet M, Raphael C Cunha, Roumen A Varbanov, Yee Shwen Hoh, Margaret L Knisley, e Mary Alice Holmes. 2015. «Survival analysis of faculty retention and promotion in the social sciences by gender». *PloS one* 10 (11): e0143093.

Breiman, Leo. 2001a. «Random Forests». *Machine Learning* 45 (1): 5–32. https://doi.org/10.1023/A:1010933404324.

———. 2001b. «Random forests». *Machine learning* 45: 5–32.

Breiman, L., Jerome H. Friedman, Richard A. Olshen, e C. J. Stone. 1984. «Classification and Regression Trees». Em.

Brentnall, Adam R, e Jack Cuzick. 2018. «Use of the concordance index for predictors of censored survival data». *Statistical methods in medical research* 27 (8): 2359–73.

Cabitza, Federico, Andrea Campagner, Felipe Soares, Luis García de Guadiana-Romualdo, Feyissa Challa, Adela Sulejmani, Michela Seghezzi, e Anna Carobene. 2021. «The importance of being external. methodological insights for the external validation of machine learning models in medicine». *Computer Methods and Programs in Biomedicine* 208: 106288.

Cavalcante, Thalytta, Raydonal Ospina, Víctor Leiva, Xavier Cabezas, e Carlos Martin-Barreiro. 2023. «Weibull regression and machine learning survival models: Methodology, comparison, and application to biomedical data related to cardiac surgery». *Biology* 12 (3): 442.

Collett, David. 2023. *Modelling survival data in medical research*. Chapman; Hall/CRC.

Cox, David R. 1972. «Regression models and life-tables. Journal of the Royal Statistical Society». *Series B (Methodological)* 34 (2): 187–220.

Daemen, Anneleen, e Bart De Moor. 2009. «Development of a kernel function for clinical data». Em *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 5913–17. IEEE.

Fernandes, A. M. da S., A. J. Mansur, L. F. Canêo, D. D. Lourenço, M. A. Piccioni, S. M. Franchi, C. M. C. Afiune, J. W. Gadioli, S. de A. Oliveira, e J. A. F. Ramires. 2004. «Redução do período de internação e de despesas no atendimento de portadores de cardiopatias congênitas submetidos à intervenção cirúrgica cardíaca no protocolo da via rápida». *Arquivos Brasileiros de Cardiologia* 83: 18–26.

Fouodo, Césaire JK, Inke R König, Claus Weihs, Andreas Ziegler, e Marvin N Wright. 2018. «Support Vector Machines for Survival Analysis with R.» *R Journal* 10 (1).

Gordon, Louis, e Richard A Olshen. 1985. «Tree-structured survival analysis.» *Cancer treatment reports* 69 (10): 1065–69.

Guan, Yuanfang, Hongyang Li, Daiyao Yi, Dongdong Zhang, Changchang Yin, Keyu Li, e Ping Zhang. 2021. «A survival model generalized to regression learning algorithms». *Nature computational science* 1 (6): 433–40.

Hastie, Trevor, Robert Tibshirani, e Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer Series em Statistics. New York, NY, USA: Springer New York Inc.

Hothorn, Torsten, Kurt Hornik, e Achim Zeileis. 2006. «Unbiased recursive partitioning: A conditional inference framework». *Journal of Computational and Graphical statistics* 15 (3): 651–74.

Hu, Chen, e Jon Arni Steingrimsson. 2018. «Personalized risk prediction in clinical oncology research: applications and practical issues using survival trees and random forests». *Journal of biopharmaceutical statistics* 28 (2): 333–49.

Huang, Yinan, Jieni Li, Mai Li, e Rajender Aparasu. 2023. «Application of machine learning in predicting survival outcomes involving real-world data: a scoping review». *BMC Medical Research Methodology* 23 (novembro). https://doi.org/10.1186/s12874-023-02078-1.

Ishwaran, Hemant, B. Kogalur Udaya, H. Blackstone Eugene, e S. Lauer Michael. 2008. «Random Survival Forests». *The Annals of Applied Statistics* 2: 841–60. https://doi.org/10.2307/30245111.

Ishwaran, H., M. S. Lauer, E. H. Blackstone, M. Lu, e U. B. Kogalur. 2021. «randomForestSRC: random survival forests vignette.» 2021. http://randomforestsrc.org/articles/survival.html.

Klein, John P, e Melvin L Moeschberger. 2006. *Survival analysis: techniques for censored and truncated data*. Springer Science & Business Media.

Kvamme, Håvard, Ørnulf Borgan, e Ida Scheel. 2019. «Time-to-Event Prediction with Neural Networks and Cox Regression». *ArXiv* abs/1907.00825. https://api.semanticscholar.org/CorpusID:195767074.

Langbein, Sophie Hanna, Mateusz Krzyziński, Mikołaj Spytek, Hubert Baniecki, Przemysław Biecek, e Marvin N Wright. 2024. «Interpretable machine learning for survival analysis». *arXiv preprint arXiv:2403.10250*.

LeBlanc, Michael, e John Crowley. 1992. «Relative risk trees for censored survival data». *Biometrics*, 411–25.

Loprinzi, Charles Lawrence, John A Laurie, H Sam Wieand, James E Krook, Paul J Novotny, John W Kugler, Joan Bartel, Marlys Law, Marilyn Bateman, e Nancy E Klatt. 1994. «Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group.» *Journal of Clinical Oncology* 12 (3): 601–7.

Lundberg, Scott M, e Su-In Lee. 2017. «A Unified Approach to Interpreting Model Predictions». Em *Advances in Neural Information Processing Systems 30*, editado por I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, e R. Garnett, 4765–74. Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Maia, Mateus, Jonatha Sousa Pimentel, Raydonal Ospina, e Anderson Ara. 2023. «Wavelet Support Vector Censored Regression». *Analytics* 2 (2): 410–25.

Mayer, M. 2023. «shapviz: SHAP Visualizations». *R package version 0.9. 0*.

Mingers, John. 1989. «An Empirical Comparison of Pruning Methods for Decision Tree Induction». *Machine Learning* 4 (janeiro): 227–43. https://doi.org/10.1023/A:1022604100933.

Pfisterer, Florian, Chris Harbron, Gunther Jansen, e Tao Xu. 2022. «Evaluating domain generalization for survival analysis in clinical studies». Em *Conference on Health, Inference, and Learning*, 32–47. PMLR.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rahman, Md Mahmudur, e Sanjay Purushotham. 2022. «Fair and interpretable models for survival analysis». Em *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1452–62.

Sambasivan, Rajiv, Sourish Das, e Sujit K Sahu. 2020. «A Bayesian perspective of statistical machine learning for big data». *Computational Statistics* 35 (3): 893–930.

Segal, Mark Robert. 1988. «Regression trees for censored data». *Biometrics*, 35–47.

Shivaswamy, Pannagadatta K, Wei Chu, e Martin Jansche. 2007. «A support vector approach to censored targets». Em *Seventh IEEE international conference on data mining (ICDM 2007)*, 655–60. IEEE.

Smola, Alex J, e Bernhard Schölkopf. 2004. «A tutorial on support vector regression». *Statistics and computing* 14: 199–222.

Spooner, Annette, Emily Chen, Arcot Sowmya, Perminder Sachdev, Nicole Kochan, Julian Trollor, e Henry Brodaty. 2020. «A comparison of machine learning methods for survival analysis of high-dimensional clinical data for dementia prediction». *Scientific Reports* 10 (novembro). https://doi.org/10.1038/s41598-020-77220-w.

Štěpánek, Lubomír, Filip Habarta, Ivana Malá, Luboš Marek, e Filip Pazdírek. 2020. «A Machine-learning Approach to Survival Time-event Predicting: Initial Analyses using Stomach Cancer Data». Em *2020 International Conference on e-Health and Bioengineering (EHB)*, 1–4. https://doi.org/10.1109/EHB50910.2020.9280301.

Therneau, Terry, e Beth Atkinson. 2022. *rpart: Recursive Partitioning and Regression Trees*. https://CRAN.R-project.org/package=rpart.

Van Belle, Vanya, Kristiaan Pelckmans, Sabine Van Huffel, e Johan AK Suykens. 2011. «Support vector methods for survival analysis: a comparison between ranking and regression approaches». *Artificial intelligence in medicine* 53 (2): 107–18.

Vapnik, Vladimir. 1998. «Statistical learning theory». *John Wiley & Sons google schola* 2: 831–42.

Vapnik, Vladimir Naumovich. 2000. «The nature of statistical learning theory, ser. Statistics for engineering and information science». *New York: Springer* 21 (1003-1008): 182.

Wang, Jingya, Matt Williams, e Erisa Karafili. 2018. «Apply Machine Learning Approaches to Survival Data». *Imperial College London*.

Wang, Ping, Yan Li, e Chandan K Reddy. 2019. «Machine learning for survival analysis: A survey». *ACM Computing Surveys (CSUR)* 51 (6): 1–36.

Wiegrebe, Simon, Philipp Kopper, Raphael Sonabend, e Andreas Bender. 2023. «Deep Learning for Survival Analysis: A Review.» *CoRR* abs/2305.14961. http://dblp.uni-trier.de/db/journals/corr/corr2305.html#abs-2305-14961.

Yang, Zhen, Juho Kanniainen, Tomi Krogerus, e Frank Emmert-Streib. 2022. «Prognostic modeling of predictive maintenance with survival analysis for mobile work equipment». *Scientific Reports* 12 (1): 8529.

Zeileis, Achim, Torsten Hothorn, e Kurt Hornik. 2008. «Model-Based Recursive Partitioning». *Journal of Computational and Graphical Statistics* 17 (2): 492–514. https://doi.org/10.1198/106186008X319331.

Zhou, Yan, e John J McArdle. 2015. «Rationale and applications of survival tree and survival ensemble methods». *Psychometrika* 80: 811–33.