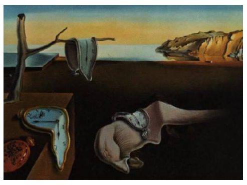
Journal of Statistical Software

*MMMMMM YYYY, Volume VV, Issue II.* *http://www.jstatsoft.org/*



# ggRandomForests: Exploring Random Forest Survival

**John Ehrlinger**
Microsoft

**Abstract**

Random forest (Leo Breiman 2001a) (RF) is a non-parametric statistical method requiring no distributional assumptions on covariate relation to the response. RF is a robust, nonlinear technique that optimizes predictive accuracy by fitting an ensemble of trees to stabilize model estimates. Random survival forests (RSF) (Ishwaran and Kogalur 2007; Ishwaran et al. 2008) are an extension of Breimans RF techniques allowing efficient non-parametric analysis of time to event data. The **randomForestSRC** package (Ishwaran and Kogalur 2014) is a unified treatment of Breimans random forest for survival, regression and classification problems. Predictive accuracy makes RF an attractive alternative to parametric models, though complexity and interpretability of the forest hinder wider application of the method. We introduce the **ggRandomForests** package, tools for visually understand random forest models grown in R (R Core Team 2014) with the **randomForest-SRC** package. The **ggRandomForests** package is structured to extract intermediate data objects from **randomForestSRC** objects and generate figures using the **ggplot2** (Wickham 2009) graphics package. This document is structured as a tutorial for building random forest for survival with the **randomForestSRC** package and using the **ggRandomForests** package for investigating how the forest is constructed. We analyse the Primary Biliary Cirrhosis of the liver data from a clinical trial at the Mayo Clinic (Fleming and Harrington 1991). We demonstrate random forest variable selection using Variable Importance (VIMP) (Leo Breiman 2001a) and Minimal Depth (Ishwaran et al. 2010), a property derived from the construction of each tree within the forest. We will also demonstrate the use of variable dependence and partial dependence plots (Friedman 2000) to aid in the interpretation of RSF results. We then examine variable interactions between covariates using conditional variable dependence plots. Our aim is to demonstrate the strength of using Random Forest methods for both prediction and information retrieval, specifically in time to event data settings.

*Keywords*: random forest, survival, vimp, minimal depth, R, **randomForestSRC**, **ggRandom-Forests**, **randomForest**.

## 1. Introduction

Random forest (Leo Breiman 2001a) (RF) is a non-parametric statistical method which requires no distributional assumptions on covariate relation to the response. RF is a robust, nonlinear technique that optimizes predictive accuracy by fitting an ensemble of trees to stabilize model estimates. Random Survival Forest (RSF) (Ishwaran and Kogalur 2007; Ishwaran et al. 2008) is an extension of Breiman's RF techniques to survival settings, allowing efficient non-parametric analysis of time to event data. The **randomForestSRC** package (`http://CRAN.R-project.org/package=randomForestSRC`) (Ishwaran and Kogalur 2014) is a unified treatment of Breiman's random forest for survival, regression and classification problems.

Predictive accuracy make RF an attractive alternative to parametric models, though complexity and interpretability of the forest hinder wider application of the method. We introduce the **ggRandomForests** package (`http://CRAN.R-project.org/package=ggRandomForests`) for visually exploring random forest models. The **ggRandomForests** package is structured to extract intermediate data objects from **randomForestSRC** objects and generate figures using the **ggplot2** graphics package (`http://CRAN.R-project.org/package=ggplot2`) (Wickham 2009).

Many of the figures created by the **ggRandomForests** package are also available directly from within the **randomForestSRC** package. However **ggRandomForests** offers the following advantages:

- Separation of data and figures: **ggRandomForests** contains functions that operate on either the `rfsrc` forest object directly, or on the output from **randomForestSRC** post processing functions (i.e., `plot.variable`, `var.select`) to generate intermediate **ggRandomForests** data objects. **ggRandomForests** functions are provide to further process these objects and plot results using the **ggplot2** graphics package. Alternatively, users can use these data objects for their own custom plotting or analysis operations.
- Each data object/figure is a single, self contained unit. This allows simple modification and manipulation of the data or `ggplot` objects to meet users specific needs and requirements.
- We chose to use the **ggplot2** package for our figures for flexibility in modifying the output. Each **ggRandomForests** plot function returns either a single `ggplot` object, or a `list` of `ggplot` objects, allowing the use of additional **ggplot2** functions to modify and customize the final figures.

This document is structured as a tutorial for using the **randomForestSRC** package for building and post-processing random survival forest models and using the **ggRandomForests** package for understanding how the forest is constructed. In this tutorial, we will build a random survival forest for the primary biliary cirrhosis (PBC) of the liver data set (Fleming and Harrington 1991), available in the **randomForestSRC** package.

In Section **??** we introduce the `pbc` data set and summarize the proportional hazards analysis of this data from Chapter 4 of (Fleming and Harrington 1991). In Section **??**, we describe how to grow a random survival forest with the **randomForestSRC** package. Random forest is not a parsimonious method, but uses all variables available in the data set to construct the response predictor. We demonstrate random forest variable selection techniques (Section **??**) using Variable Importance (VIMP) (Leo Breiman 2001a) in Section **??** and Minimal Depth (Ishwaran et al. 2010) in Section **??**. We then compare both methods with variables used in the(Fleming and Harrington 1991) model.

Once we have an idea of which variables we are most interested in, we use dependence plots(Friedman 2000) (Section **??**) to understand how these variables are related to the response. Variable dependence (Section **??**) plots give us an idea of the overall trend of a variable/response relation, while partial dependence plots (Section **??**) show us the risk adjusted relation by averaging out the effects of other variables. Dependence plots often show strongly non-linear variable/response relations that are not easily obtained through parametric modeling.

We then graphically examine forest variable interactions with the use of variable and partial dependence conditioning plots (coplots) (Chambers 1992; Cleveland 1993) (Section **??**) and close with concluding remarks in Section **??**.

## 2. Data summary: primary biliary cirrhosis (PBC) data set

The *primary biliary cirrhosis* of the liver (PBC) study consists of 424 PBC patients referred to Mayo Clinic between 1974 and 1984 who met eligibility criteria for a randomized placebo controlled trial of the drug D-penicillamine (DPCA). The data is described in (**?**, Chapter 0.2) and a partial likelihood model (Cox proportional hazards) is developed in Chapter 4.4. The `pbc` data set, included in the **randomForestSRC** package, contains 418 observations, of which 312 patients participated in the randomized trial (**?**, Appendix D).

```
R> data("pbc", package = "randomForestSRC")
```

For this analysis, we modify some of the data for better formatting of our results. Since the data contains about 12 years of follow up, we prefer using `years` instead of `days` to describe survival. We also convert the `age` variable to years, and the `treatment` variable to a factor containing levels of `c("DPCA", "placebo")`. The variable names, type and description are given in Table **??**.

### 2.1. Exploratory data analysis

It is good practice to view your data before beginning analysis. Exploratory Data Analysis (EDA) (Tukey 1977) will help you to understand the data, and find outliers, missing values and other data anomalies within each variable before getting deep into the analysis. To this end, we use **ggplot2** figures with the `facet_wrap` function to create two sets of panel plots, one of histograms for categorical variables (Figure 1), and another of scatter plots for continuous variables (Figure 2). Variables are plotted along a continuous variable on the X-axis to separate the individual observations.

In categorical EDA plots (Figure 1), we are looking for patterns of missing data (white portion of bars). We often use surgical date for our X-axis variable to look for possible periods of low enrollment. There is not a comparable variable available in the `pbc` data set, so instead we used follow up time (`years`). Another reasonable choice may have been to use the patient

Table 1: ‘pbc‘ data set variable dictionary.

| Variable name | Description | Type |
|---|---|---|
| years | Time (years) | numeric |
| status | Event (F = censor, T = death) | logical |
| treatment | Treament (DPCA, Placebo) | factor |
| age | Age (years) | numeric |
| sex | Female = T | logical |
| ascites | Presence of Asictes | logical |
| hepatom | Presence of Hepatomegaly | logical |
| spiders | Presence of Spiders | logical |
| edema | Edema (0, 0.5, 1) | factor |
| bili | Serum Bilirubin (mg/dl) | numeric |
| chol | Serum Cholesterol (mg/dl) | integer |
| albumin | Albumin (gm/dl) | numeric |
| copper | Urine Copper (ug/day) | integer |
| alk | Alkaline Phosphatase (U/liter) | numeric |
| sgot | SGOT (U/ml) | numeric |
| trig | Triglicerides (mg/dl) | integer |
| platelet | Platelets per cubic ml/1000 | integer |
| prothrombin | Prothrombin time (sec) | numeric |
| stage | Histologic Stage | factor |

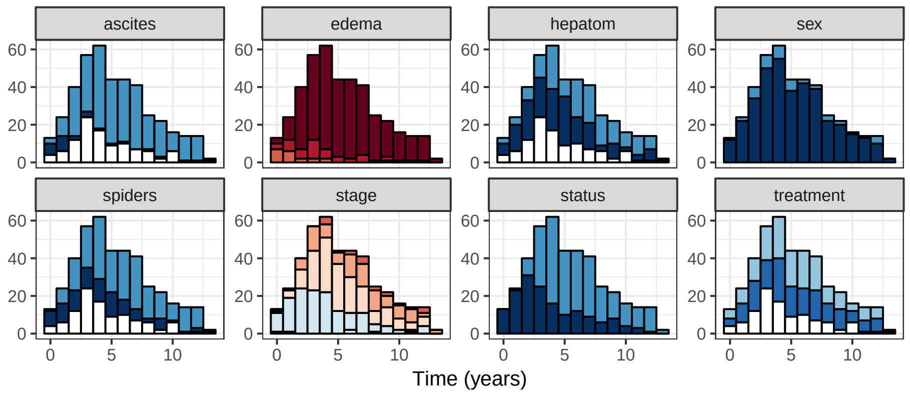


Figure 1: EDA plots for categorical variables (logicals and factors). Bars indicate number of patients within 1 year of followup interval for each categorical variable. Colors correspond to class membership within each variable. Missing values are included in white.

`age` variable for the X-axis. The important quality of the selected variable is to spread the observations out to aid in finding data anomalies.

In continuous data EDA plots (Figure 2), we are looking for missingness (rug marks) and extreme or non-physical values. For survival settings, we color and shape the points as red ‘x’s to indicate events, and blue circles to indicate censored observation.

Extreme value examples are evident in a few of the variables in Figure 2. We are typically

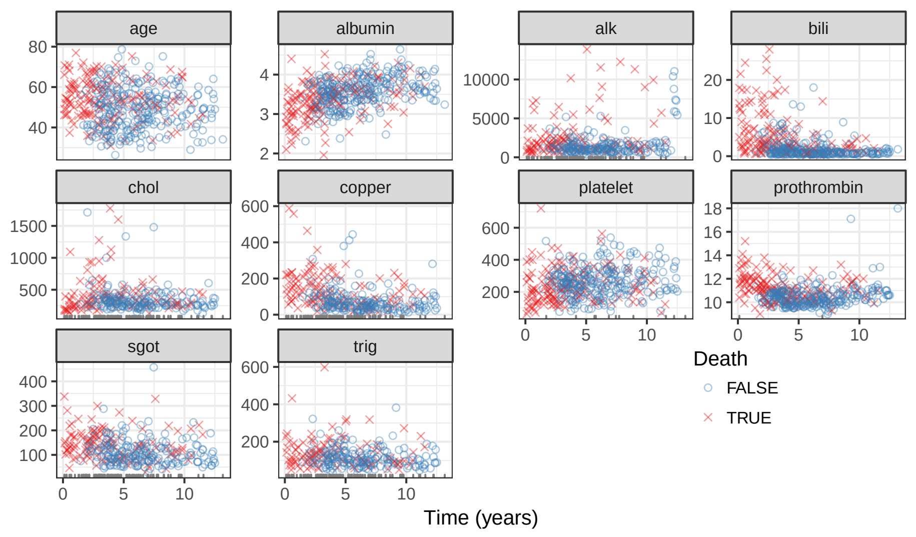


Figure 2: EDA plots for continuous variables. Symbols indicate observations with variable value on Y-axis against follow up time in years. Symbols are colored and shaped according to the death event (‘status‘ variable). Missing values are indicated by rug marks along the X-axis

looking for values that are outside of the biological range. This is often caused by measurements recorded in differing units, which can sometimes be corrected algorithmically. Since we can not ask the original investigator to clarify these values in this particular study, we will continue without modifying the data.

Both EDA figures indicate the `pbc` data set contains quite a bit of missing data. Table **??** shows the number of missing values in each variable of the `pbc` data set. Of the 19 variables in the data, 12 have missing values. The `pbc` column details variables with missing data in the full `pbc` data set, though there are patients that were not randomized into the trial. If we restrict the data to the trial only, most of the missing values are also removed, leaving only 4 variables with missing values. Therefore, we will focus on the 312 observations from the clinical trial for the remainder of this document. We will discuss how **randomForestSRC** handles missing values in Section **??**.

## 2.2. PBC Model Summary

We conclude the data set investigation with a summary of(Fleming and Harrington 1991) model results from Chapter 4.4. We start by generating Kaplan–Meier (KM) survival estimates comparing the treatment groups of DPCA and placebo. We use the **ggRandomForests** `gg_survival` function to generate these estimates from the data set as follows.

```
R> # Create the trial and test data sets.
R> pbc.trial <- pbc %>% filter(!is.na(treatment))
R> pbc.test <- pbc %>% filter(is.na(treatment))
R>
R> # Create the gg_survival object
R> gg_dta <- gg_survival(interval = "years",
```

Table 2: Missing value counts in ‘pbc‘ data set and pbc clinical trial observations (‘pbc.trial‘).

| | pbc | pbc.trial |
|---|---|---|
| treatment | 106 | 0 |
| ascites | 106 | 0 |
| hepatom | 106 | 0 |
| spiders | 106 | 0 |
| chol | 134 | 28 |
| copper | 108 | 2 |
| alk | 106 | 0 |
| sgot | 106 | 0 |
| trig | 136 | 30 |
| platelet | 11 | 4 |
| prothrombin | 2 | 0 |
| stage | 6 | 0 |

```
R+                          censor = "status",
R+                          by = "treatment",
R+                          data = pbc.trial,
R+                          conf.int = 0.95)
```

The code block reduces the `pbc` data set to the `pbc.trial` which only include observations from the clinical trial. The remaining observations are stored in the `pbc.test` data set for later use. The **ggRandomForests** package is designed to use a two step process in figure generation. The first step is data generation, where we store a `gg_survival` data object in the `gg_dta` object. The `gg_survival` function uses the `data` set, follow up `interval`, `censor` indicator and an optional grouping argument (`by`). By default `gg_survival` also calculates 95% confidence band, which we can control with the `conf.int` argument.

In the figure generation step, we use the **ggRandomForests** plot routine `plot.gg_survival` as shown in the following code block. The `plot.gg_survival` function uses the `gg_dta` data object to plot the survival estimate curves for each group and corresponding confidence interval ribbons. We have used additional **ggplot2** commands to modify the axis and legend labels (`labs`), the legend location (`theme`) and control the plot range of the y-axis (`coord_cartesian`) for this figure.

```
R> plot(gg_dta) +
R+   labs(y = "Survival Probability", x = "Observation Time (years)",
R+        color = "Treatment", fill = "Treatment") +
R+   theme(legend.position = c(0.2, 0.2)) +
R+   coord_cartesian(y = c(0, 1.01))
```

The `gg_survival` plot of Figure 3 is analogous to(Fleming and Harrington 1991) Figure 0.2.3 and Figure 4.4.1, showing there is little difference between the treatment and control groups.

The `gg_survival` function generates a variety of time-to-event estimates, including the cumulative hazard. The follow code block creates a cumulative hazard plot (**?**, Figure 0.2.1)

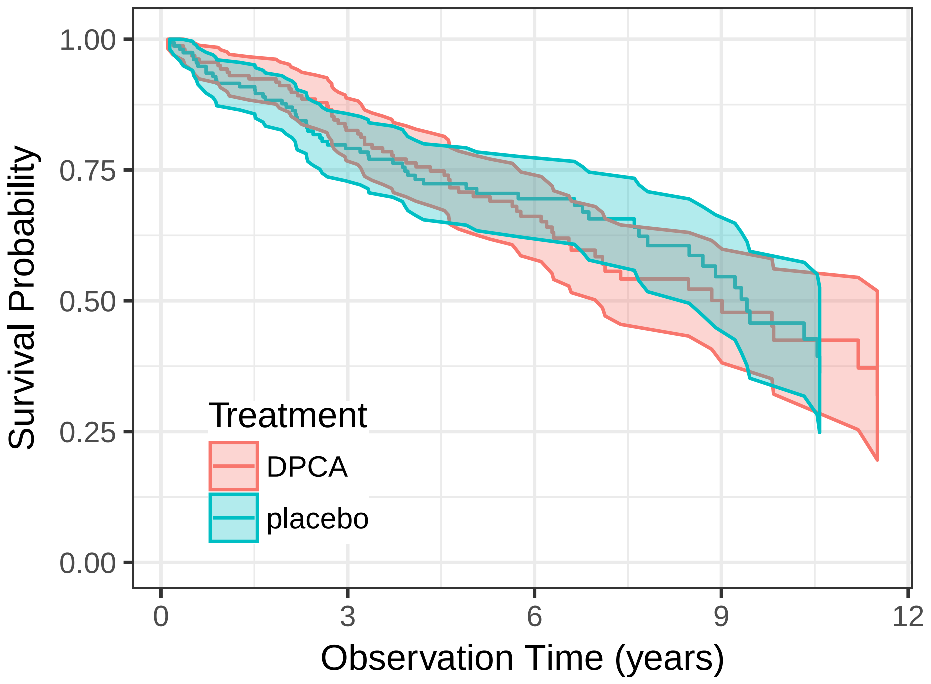


Figure 3: Kaplan–Meier survival estimates comparing the DPCA treatment (red) with placebo (blue) groups for the pbc.trail data set. Median survival with shaded 95% confidence band.

in Figure 4 using the same data object generated by the original `gg_survival` function call. The red `DPCA` line is directly comparable to Figure 0.2.1, we've add the cumulative hazard estimates for the `placebo` population in blue.

```
R> plot(gg_dta, type = "cum_haz") +
R+   labs(y = "Cumulative Hazard", x = "Observation Time (years)",
R+        color = "Treatment", fill = "Treatment") +
R+   theme(legend.position = c(0.2, 0.8)) +
R+   coord_cartesian(ylim = c(-0.02, 1.22))
```

In Figure 3, we demonstrated grouping on the categorical variable (`treatment`). To demonstrate plotting grouped survival on a continuous variable, we examine KM estimates of survival within stratified groups of bilirubin measures. The groupings are obtained directly from(Fleming and Harrington 1991) Figure 4.4.2, where they presented univariate model results of predicting survival on a function of bilirubin.

We set up the `bili` groups on a temporary data set (`pbc.bili`) using the `cut` function with intervals matching the reference figure. For this example we combine the data generation and plot steps into a single line of code. The `error` argument of the `plot.gg_survival` function is used to control display of the confidence bands. We suppress the intervals for this figure with `error = "none"` and again modify the plot display with **ggplot2** commands to generate Figure 5.

```
R> pbc.bili <- pbc.trial
```

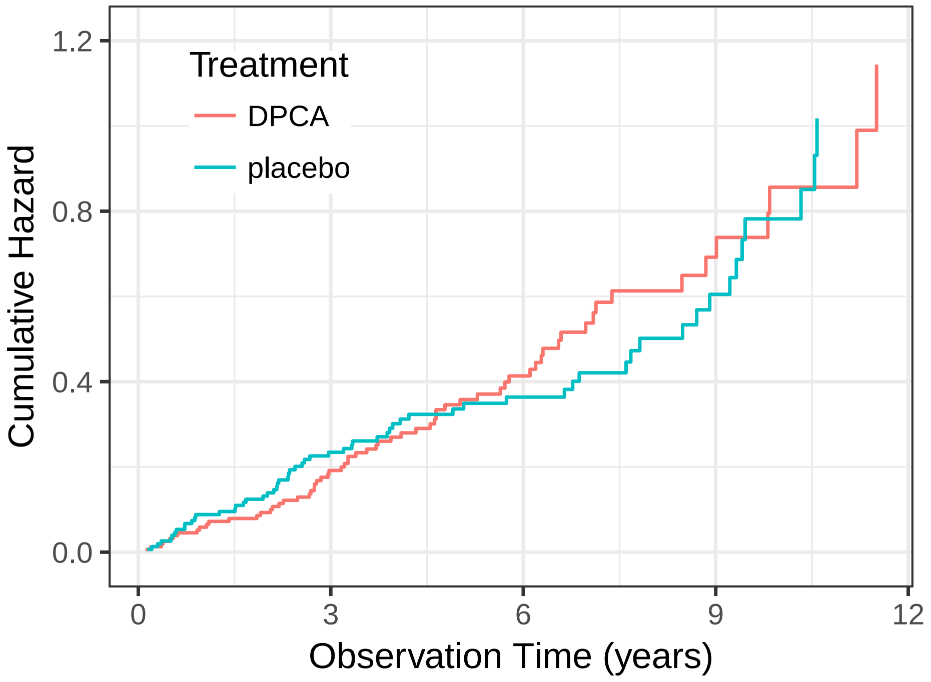


Figure 4: Kaplan–Meier cumulative hazard estimates comparing the DPCA treatment (red) with placebo (blue) groups for the pbc data set.

```
R> pbc.bili$bili_grp <- cut(pbc.bili$bili, breaks = c(0, 0.8, 1.3, 3.4, 29))
R>
R> plot(gg_survival(interval = "years", censor = "status", by = "bili_grp",
R+                  data = pbc.bili), error = "none") +
R+   labs(y = "Survival Probability", x = "Observation Time (years)",
R+        color = "Bilirubin")
```

In Chapter 4,(Fleming and Harrington 1991) use partial likelihood methods to build a linear model with log transformations on some variables. We summarize the final, biologically reasonable model in Table **??** for later comparison with our random forest results.

Table 3: ‘pbc‘ proportional hazards model summary of 312 randomized cases in ‘pbc.trial‘ data set. (Table 4.4.3c [@fleming:1991])

| | Coef. | Std. Err. | Z stat. |
|---|---|---|---|
| Age | 0.033 | 0.009 | 3.84 |
| log(Albumin) | -3.055 | 0.724 | -4.22 |
| log(Bilirubin) | 0.879 | 0.099 | 8.90 |
| Edema | 0.785 | 0.299 | 2.62 |
| log(Prothrombin Time) | 3.016 | 1.024 | 2.95 |

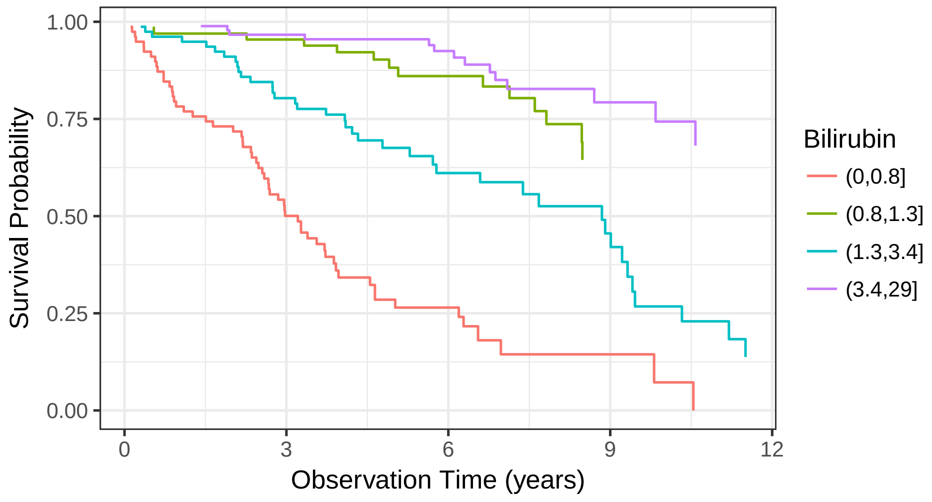


Figure 5: Kaplan–Meier survival estimates comparing different groups of Bilirubin measures (bili) for the pbc data set. Groups defined in Chapter 4 of [@fleming:1991].

## 3. Random survival forest

A Random Forest (Leo Breiman 2001a) is grown by *bagging* (L Breiman 1996a) a collection of *classification and regression trees* (CART) (Breiman et al. 1984). The method uses a set of *B bootstrap* (Efron and Tibshirani 1994) samples, growing an independent tree model on each sub-sample of the population. Each tree is grown by recursively partitioning the population based on optimization of a *split rule* over the $p$-dimensional covariate space. At each split, a subset of $m \leq p$ candidate variables are tested for the split rule optimization, dividing each node into two daughter nodes. Each daughter node is then split again until the process reaches the *stopping criteria* of either *node purity* or *node member size*, which defines the set of *terminal (unsplit) nodes* for the tree. In regression trees, node impurity is measured by mean squared error, whereas in classification problems, the Gini index is used(Friedman 2000) .

Random forest sorts each training set observation into one unique terminal node per tree. Tree estimates for each observation are constructed at each terminal node, among the terminal node members. The Random Forest estimate for each observation is then calculated by aggregating, averaging (regression) or votes (classification), the terminal node results across the collection of $B$ trees.

Random Survival Forest (Ishwaran 2007; Ishwaran et al. 2008) (RSF) are an extension of Random Forest to analyze right censored, time to event data. A forest of survival trees is grown using a log-rank splitting rule to select the optimal candidate variables. Survival estimate for each observation are constructed with a Kaplan–Meier (KM) estimator within each terminal node, at each event time.

Random Survival Forests adaptively discover nonlinear effects and interactions and are fully nonparametric. Averaging over many trees enables RSF to approximate complex survival functions, including non-proportional hazards, while maintaining low prediction error. (Ishwaran and Kogalur 2010) showed that RSF is uniformly consistent and that survival forests have a uniform approximating property in finite-sample settings, a property not possessed by individual survival trees.

The **randomForestSRC** `rfsrc` function call grows the forest, determining the type of forest by the response supplied in the `formula` argument. In the following code block, we grow a random forest for survival, by passing a survival (`Surv`) object to the forest. The forest uses all remaining variables in the `pbc.trial` data set to generate the RSF survival model.

```
R> rfsrc_pbc <- rfsrc(Surv(years, status) ~ ., data = pbc.trial,
R+                    nsplit = 10, na.action = "na.impute",
R+                    tree.err = TRUE)
```

The `print.rfsrc` function returns information on how the random forest was grown. Here the `family = "surv"` forest has `ntree = 1000` trees (the default `ntree` argument). The forest selected from `ceil`$(\sqrt{p = 17}) = 5$ randomly selected candidate variables for splitting at each node, stopping when a terminal node contained three or fewer observations. For continuous variables, we used a random logrank split rule, which randomly selects from `nsplit = 10` split point values, instead of optimizing over all possible values.

## 3.1. Generalization error

One advantage of random forest is a built in generalization error estimate. Each bootstrap sample selects approximately 63.2% of the population on average. The remaining 36.8% of observations, the Out-of-Bag (L Breiman 1996b) (OOB) sample, can be used as a hold out test set for each tree. An OOB prediction error estimate can be calculated for each observation by predicting the response over the set of trees which were not trained with that particular observation. Out-of-Bag prediction error estimates have been shown to be nearly identical to $n$–fold cross validation estimates (Hastie, Tibshirani, and Friedman 2009). This feature of random forest allows us to obtain both model fit and validation in one pass of the algorithm.

The `gg_error` function operates on the random forest (`rfsrc_pbc`) object to extract the error estimates as a function of the number of trees in the forest. The following code block first creates a `gg_error` data object, then uses the `plot.gg_error` function to create a `ggplot` object for display in a single line of code.

```
R> plot(gg_error(rfsrc_pbc))
```

The `gg_error` plot of Figure 6 demonstrates that it does not take a large number of trees to stabilize the forest prediction error estimate. However, to ensure that each variable has enough of a chance to be included in the forest prediction process, we do want to create a rather large random forest of trees.

## 3.2. Training Set Prediction

The `gg_rfsrc` function extracts the OOB prediction estimates from the random forest. This code block executes the data extraction and plotting in one line, since we are not interested in holding the prediction estimates for later reuse. Each of the **ggRandomForests** plot commands return `ggplot` objects, which we can also store for modification or reuse later in the analysis (`ggRFsrc` object). Note that we again use additional **ggplot2** commands to modify the display of the plot object.

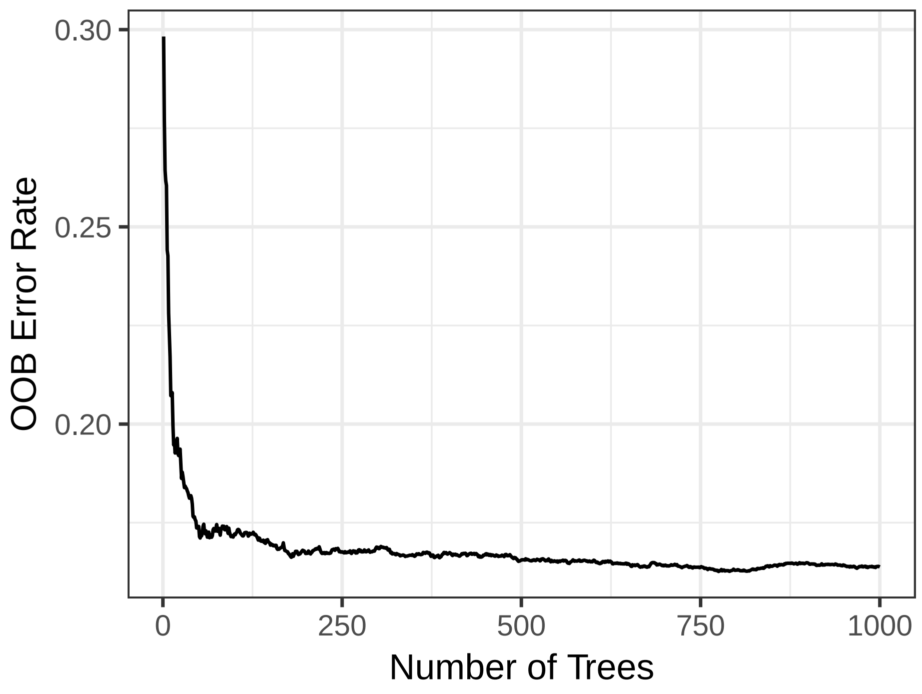


Figure 6: Random forest OOB prediction error estimates as a function of the number of trees in the forest.

```
R> ggRFsrc <- plot(gg_rfsrc(rfsrc_pbc), alpha = 0.2) +
R+   scale_color_manual(values = strCol) +
R+   theme(legend.position = "none") +
R+   labs(y = "Survival Probability", x = "Time (years)") +
R+   coord_cartesian(ylim = c(-0.01, 1.01))
R> show(ggRFsrc)
```

The `gg_rfsrc` plot of Figure 7 shows the predicted survival from our RSF model. Each line represents a single patient in the training data set, where censored patients are colored blue, and patients who have experienced the event (death) are colored in red. We extend all predicted survival curves to the longest follow up time (12 years), regardless of the actual length of a patient's follow up time.

Interpretation of general survival properties from Figure 7 is difficult because of the number of curves displayed. To get more interpretable results, it is preferable to plot a summary of the survival results. The following code block compares the predicted survival between treatment groups, as we did in Figure 3.

```
R> plot(gg_rfsrc(rfsrc_pbc, by = "treatment")) +
R+   theme(legend.position = c(0.2, 0.2)) +
R+   labs(y = "Survival Probability", x = "Time (years)") +
R+   coord_cartesian(ylim = c(-0.01, 1.01))
```

The `gg_rfsrc` plot of Figure 8 shows the median survival with a 95% shaded confidence band for the `DPCA` group in red, and the `placebo` group in blue. When calling `gg_rfsrc` with either

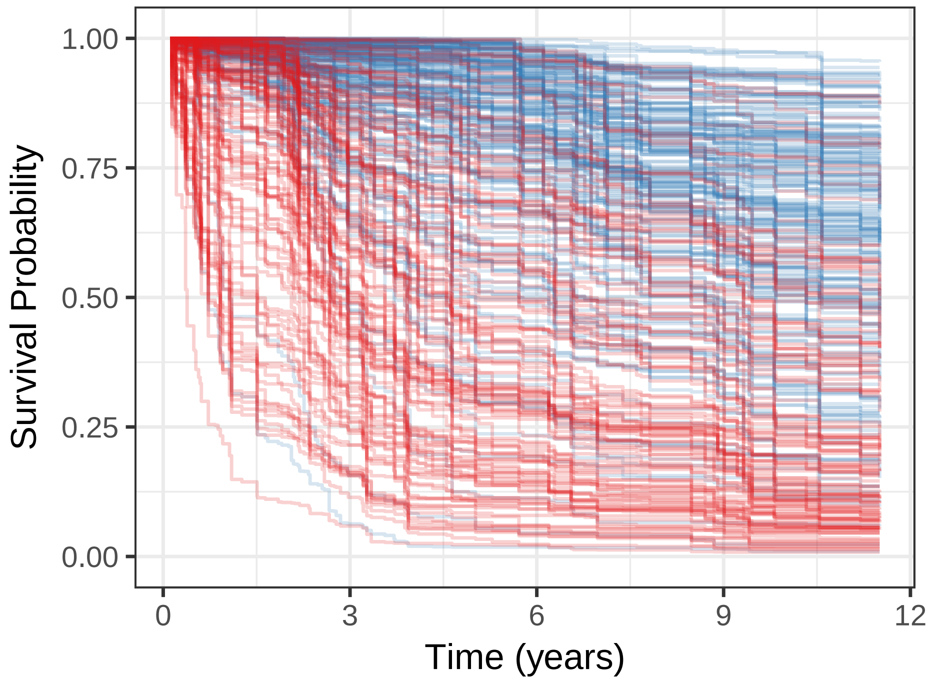


Figure 7: Random forest OOB predicted survival. Blue curves correspond to censored observations, red curves correspond to observations experiencing death events.

a `by` argument or a `conf.int` argument, the function calculates a bootstrap confidence interval around the median survival line. By default, the function will calculate the `conf.int=0.95` confidence interval, with the number of `bs.samples` equal to the number of observations.

## 3.3. Random forest imputation

There are two modeling issues when dealing with missing data values: `How does the algorithm build a model when values are missing from the training data?'', and`How does the algorithm predict a response when values are missing from the test data?". The standard procedure for linear models is to either remove or impute the missing data values before modelling. Removing the missingness is done by either removing the variable with missing values (column wise) or removing the observations (row wise). Removal is a simple solution, but may bias results when either observations or variables are scarce.

The **randomForestSRC** package imputes missing values using *adaptive tree imputation* (Ishwaran et al. 2008). Rather than impute missing values before growing the forest, the algorithm takes a "just–in–time" approach. At each node split, the set of `mtry` candidate variables is checked for missing values. Missing values are then imputed by randomly drawing values from non-missing data within the node. The split-statistic is then calculated on observations that were not missing values. The imputed values are used to sort observations into the subsequent daughter nodes and then discarded before the next split occurs. The process is repeated until the stopping criteria is reached and all observations are sorted into terminal nodes.

A final imputation step can be used to fill in missing values from within the terminal nodes. This step uses a process similar to the previous imputation but uses the OOB non-missing

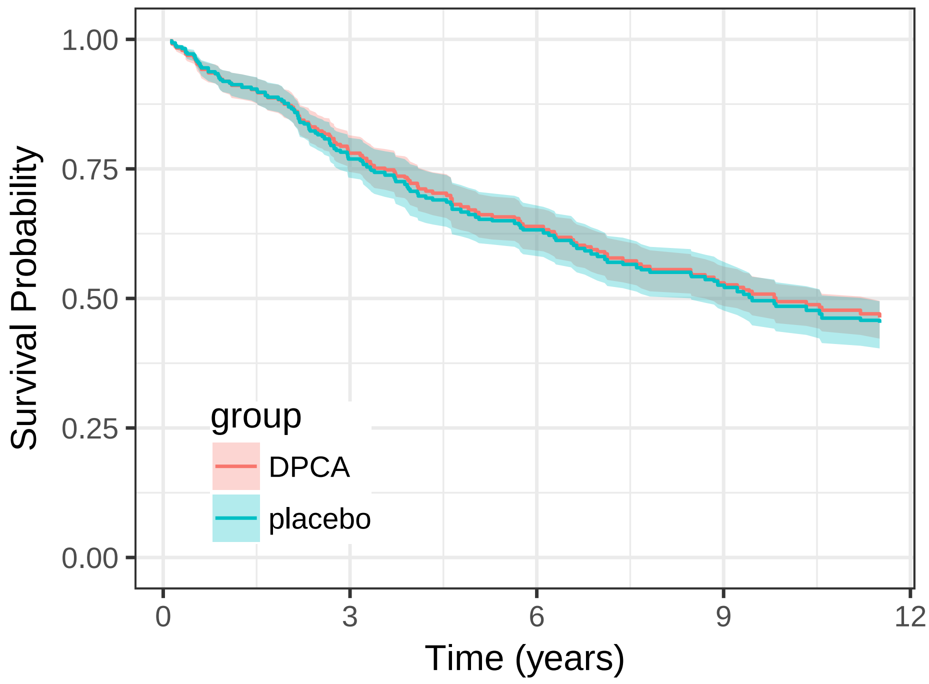


Figure 8: Random forest predicted survival stratified by treatment groups. DPCA group in red, placebo in blue with shaded 95% confidence bands.

terminal node data for the random draws. These values are aggregated (averaging for continuous variables, voting for categorical variables) over the `ntree` trees in the forest to estimate an imputed data set. By default, the missing values are not filled into the training data, but are available within the forest object for later use if desired.

Adaptive tree imputation still requires the missing at random assumptions (Rubin 1976). At each imputation step, the random forest assumes that similar observations are grouped together within each node. The random draws used to fill in missing data do not bias the split rule, but only sort observations similar in non-missing data into like nodes. An additional feature of this approach is the ability of predicting on test set observations with missing values.

## 3.4. Test set predictions

The strength of adaptive tree imputation becomes clear when doing prediction on test set observations. If we want to predict survival for patients that did not participate in the trial using the model we created in Section **??**, we need to somehow account for the missing values detailed in Table **??**.

The `predict.rfsrc` call takes the forest object (`rfsrc_pbc`), and the test data set (`pbc_test`) and returns a predicted survival using the same forest imputation method for missing values within the test data set (`na.action="na.impute"`).

```
R> rfsrc_pbc_test <- predict(rfsrc_pbc, newdata = pbc.test,
R+                            na.action = "na.impute")
```

The forest summary indicates there are 106 test set observations with 36 deaths and the predicted error rate is 19.1%. We plot the predicted survival just as we did the training set estimates.

```
R> plot(gg_rfsrc(rfsrc_pbc_test), alpha=.2) +
R+   scale_color_manual(values = strCol) +
R+   theme(legend.position = "none") +
R+   labs(y = "Survival Probability", x = "Time (years)") +
R+   coord_cartesian(ylim = c(-0.01, 1.01))
```

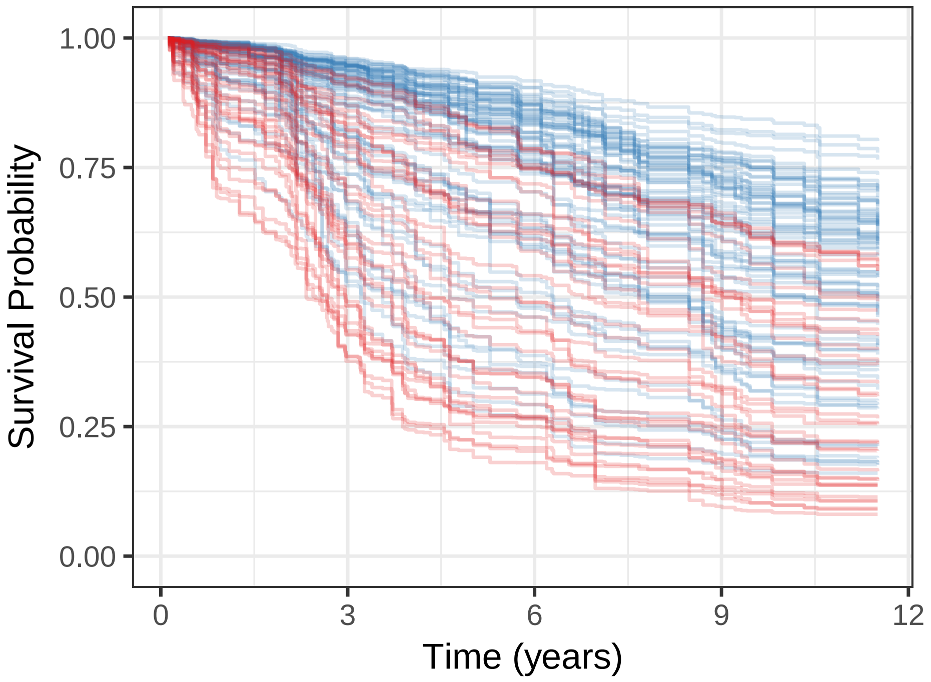


Figure 9: Random forest survival estimates for patients in the pbc.test data set. Blue curves correspond to censored patients, red curves correspond to patients experiencing a death event.

The `gg_rfsrc` plot of Figure 9 shows the test set predictions, similar to the training set predictions in Figure 7, though with fewer patients the survival curves do not cover the same area of the figure. It is important to note that because Figure 7 is constructed with OOB estimates, the survival results are comparable as estimates from unseen observations in Figure 9.

## 4. Variable selection

Random forest is not a parsimonious method, but uses all variables available in the data set to construct the response predictor. Also, unlike parametric models, random forest does not require the explicit specification of the functional form of covariates to the response. Therefore there is no explicit $p$-value/significance test for variable selection with a random forest model.

Instead, RF ascertains which variables contribute to the prediction through the split rule optimization, optimally choosing variables which separate observations.

The typical goal of a random forest analysis is to build a *prediction* model, in contrast to extracting *information* regarding the underlying process (Leo Breiman 2001b). There is not usually much care given in how variables are included into the training data set. Since the goal is prediction, investigators often include the "kitchen sink" if it can help.

In contrast, in survival settings we are typically also interested in how we can possibly improve the the outcome of interest. To achieve this, for understandable inference, it is important to avoid both duplication and transformations of variables whenever possible when building our data sets. Duplication of variables, including multiple measures of a similar covariate, can reduce or mask the importance of the covariate. Transformations can also mask importance as well as make interpretation of the inference results difficult to impossible.

In this Section, We explore two separate approaches to investigate the RF variable selection process. Variable Importance (Section **??**), a property related to variable misspecification, and Minimal Depth (Section **??**), a property derived from the construction of the trees within the forest.

## 4.1. Variable Importance

*Variable importance* (VIMP) was originally defined in CART using a measure involving surrogate variables (see Chapter 5 of (Breiman et al. 1984)). The most popular VIMP method uses a prediction error approach involving "noising-up" each variable in turn. VIMP for a variable $x_v$ is the difference between prediction error when $x_v$ is randomly permuted, compared to prediction error under the observed values (Leo Breiman 2001a; Liaw and Wiener 2002; Ishwaran 2007; Ishwaran et al. 2008).

Since VIMP is the difference in OOB prediction error before and after permutation, a large VIMP value indicates that misspecification detracts from the predictive accuracy in the forest. VIMP close to zero indicates the variable contributes nothing to predictive accuracy, and negative values indicate the predictive accuracy *improves* when the variable is misspecified. In the later case, we assume noise is more informative than the true variable. As such, we ignore variables with negative and near zero values of VIMP, relying on large positive values to indicate that the predictive power of the forest is dependent on those variables.

The `gg_vimp` function extracts VIMP measures for each of the variables used to grow the forest. The `plot.gg_vimp` function shows the variables, in VIMP rank order, labeled with the named vector in the `lbls` argument.

```
R> plot(gg_vimp(rfsrc_pbc), lbls = st.labs) +
R+   theme(legend.position = c(0.8, 0.2)) +
R+   labs(fill = "VIMP > 0")
```

The `gg_vimp` plot of Figure 10 details VIMP ranking for the `pbc.trial` baseline variables, from the largest (Serum Bilirubin) at the top, to smallest (Triglicerides) at the bottom. VIMP measures are shown using bars to compare the scale of the error increase under permutation and colored by the sign of the measure (red for negative values). Note that four of the five highest ranking variables by VIMP match those selected by the(Fleming and Harrington 1991)

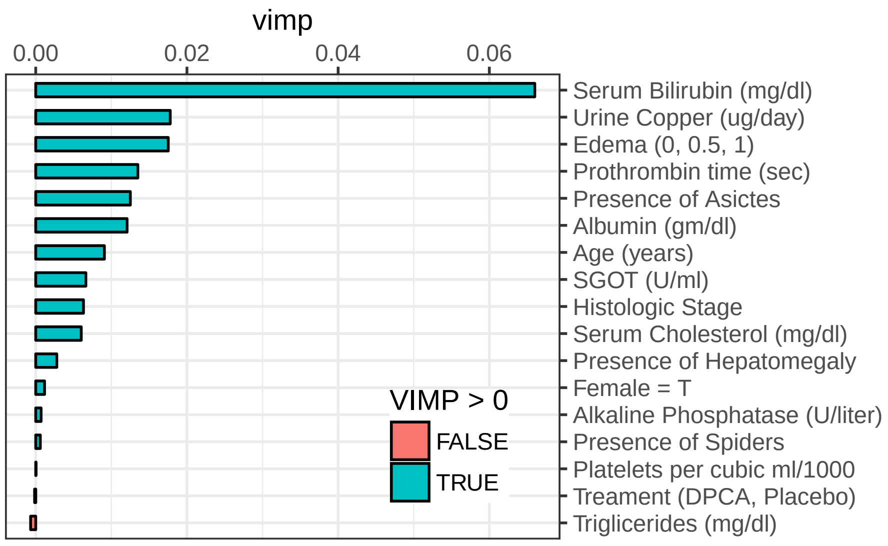


Figure 10: Random forest Variable Importance (VIMP). Blue bars indicates positive VIMP, red indicates negative VIMP. Importance is relative to positive length of bars.

model listed in Table **??**, with urine copper (2) ranking higher than age (8). We will return to this in Section **??**.

## 4.2. Minimal Depth

In VIMP, prognostic risk factors are determined by testing the forest prediction under alternative data settings, ranking the most important variables according to their impact on predictive ability of the forest. An alternative method uses inspection of the forest construction to rank variables. *Minimal depth* (Ishwaran et al. 2010; Ishwaran et al. 2011) assumes that variables with high impact on the prediction are those that most frequently split nodes nearest to the root node, where they partition the largest samples of the population.

Within each tree, node levels are numbered based on their relative distance to the root of the tree (with the root at 0). Minimal depth measures important risk factors by averaging the depth of the first split for each variable over all trees within the forest. The assumption in the metric is that smaller minimal depth values indicate the variable separates large groups of observations, and therefore has a large impact on the forest prediction.

In general, to select variables according to VIMP, we examine the VIMP values, looking for some point along the ranking where there is a large difference in VIMP measures. Given minimal depth is a quantitative property of the forest construction, **?** also derive an analytic threshold for evidence of variable impact. A simple optimistic threshold rule uses the mean of the minimal depth distribution, classifying variables with minimal depth lower than this threshold as important in forest prediction.

The **randomForestSRC** `var.select` function uses the minimal depth methodology for variable selection, returning an object with both minimal depth and vimp measures. The **ggRandomForests** `gg_minimal_depth` function is analogous to the `gg_vimp` function. Variables are ranked from most important at the top (minimal depth measure), to least at the bottom (maximal minimal depth).

```
R> varsel_pbc <- var.select(rfsrc_pbc)
```

```
minimal depth variable selection ...


-----------------------------------------------------------
family             : surv
var. selection     : Minimal Depth
conservativeness   : medium
x-weighting used?  : TRUE
dimension          : 17
sample size        : 312
ntree              : 1000
nsplit             : 10
mtry               : 5
nodesize           : 3
refitted forest    : FALSE
model size         : 13
depth threshold    : 6.7484
PE (true OOB)      : 16.378


Top variables:
            depth vimp
bili        1.744   NA
albumin     2.482   NA
copper      2.759   NA
prothrombin 2.883   NA
edema       3.246   NA
chol        3.256   NA
platelet    3.457   NA
age         3.688   NA
sgot        3.803   NA
alk         3.951   NA
trig        4.618   NA
ascites     5.246   NA
stage       5.306   NA
-----------------------------------------------------------
```

```
R> gg_md <- gg_minimal_depth(varsel_pbc, lbls = st.labs)
R> print(gg_md)
```

```
-----------------------------------------------------------
gg_minimal_depth
model size         : 13
```

```
depth threshold    : 6.7484

PE :[1] 16.378
------------------------------------------------------------

Top variables:
            depth vimp
bili         1.74   NA
albumin      2.48   NA
copper       2.76   NA
prothrombin  2.88   NA
edema        3.25   NA
chol         3.26   NA
platelet     3.46   NA
age          3.69   NA
sgot         3.80   NA
alk          3.95   NA
trig         4.62   NA
ascites      5.25   NA
stage        5.31   NA
------------------------------------------------------------
```

The `gg_minimal_depth` summary mostly reproduces the output from the `var.select` function from the **randomForestSRC** package. We report the minimal depth threshold (`threshold` 6.748) and the number of variables with depth below that threshold (`model size` 13). We also list a table of the top (13) selected variables, in minimal depth rank order with the associated VIMP measures. The minimal depth numbers indicate that `bili` tends to split between the first and second node level, and the next three variables (`albumin`, `copper`, `prothrombin`) split between the second and third levels on average.

```
R> plot(gg_md, lbls = st.labs)
```

The `gg_minimal_depth` plot of Figure 11 is similar to the `gg_vimp` plot in Figure 10, ranking variables from most important at the top (minimal depth measure), to least at the bottom (maximal minimal depth). The vertical dashed line indicates the minimal depth threshold where smaller minimal depth values indicate higher importance and larger values indicate lower importance.

## 4.3. Variable selection comparison

Since the VIMP and Minimal Depth measures use different criteria, we expect the variable ranking to be somewhat different. We use `gg_minimal_vimp` function to compare rankings between minimal depth and VIMP in Figure 12.

```
R> plot(gg_minimal_vimp(gg_md), lbls = st.labs) +
R+   theme(legend.position=c(0.8, 0.2))
```

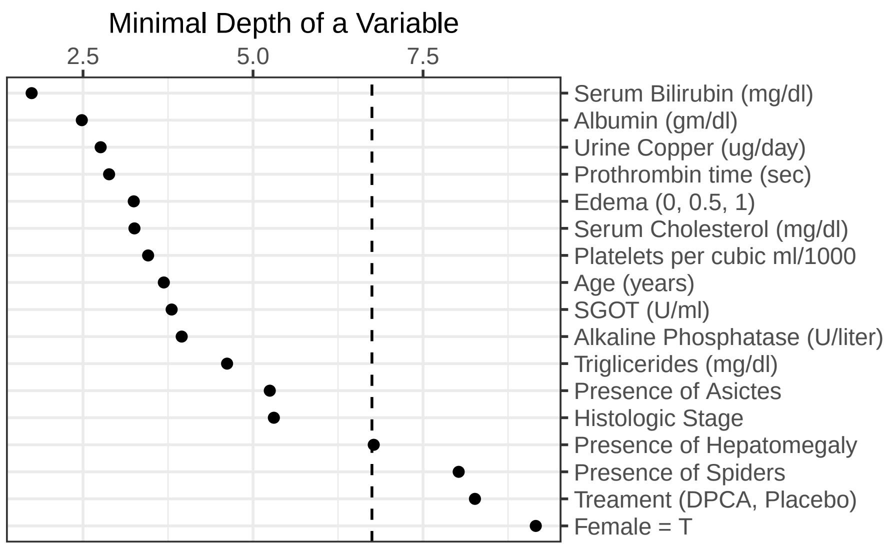


Figure 11: Minimal Depth variable selection. Low minimal depth indicates important variables. The dashed line is the threshold of maximum value for variable selection.

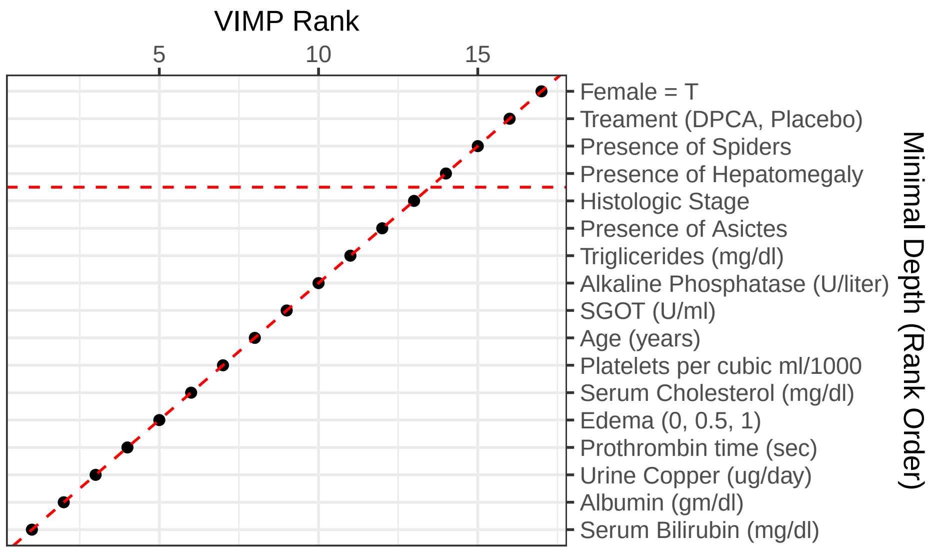


Figure 12: Comparing Minimal Depth and Vimp rankings. Points on the red dashed line are ranked equivalently, points above have higher VIMP ranking, those below have higher minimal depth ranking.

The points along the red dashed line indicate where the measures are in agreement. Points above the red dashed line are ranked higher by VIMP than by minimal depth, indicating the variables are more sensitive to misspecification. Those below the line have a higher minimal depth ranking, indicating they are better at dividing large portions of the population. The further the points are from the line, the more the discrepancy between measures.

We examine the ranking of the different variable selection methods further in Table **??**. We

Table 4: Comparison of variable selection criteria. Minimal depth ranking, VIMP ranking and [@fleming:1991] (FH) proportional hazards model ranked according to ‘abs(Z stat)‘ from Table **??**.

| Variable | FH | Min depth | VIMP |
|---|---|---|---|
| bili | 1 | 1 | 1 |
| albumin | 2 | 2 | 6 |
| copper | NA | 3 | 2 |
| prothrombin | 4 | 4 | 4 |
| edema | 5 | 5 | 3 |
| chol | NA | 6 | 10 |
| platelet | NA | 7 | 15 |
| age | 3 | 8 | 7 |
| sgot | NA | 9 | 8 |
| alk | NA | 10 | 13 |
| trig | NA | 11 | 17 |
| ascites | NA | 12 | 5 |
| stage | NA | 13 | 9 |

can use the Z statistic from Table **??** to rank variables selected in the(Fleming and Harrington 1991) model to compare with variables selected by minimal depth and VIMP. The table is constructed by taking the top ranked minimal depth variables (below the selection threshold) and matching the VIMP ranking and(Fleming and Harrington 1991) model transforms. We see all three methods indicate a strong relation of serum bilirubin to survival, and overall, the minimal depth and VIMP rankings agree reasonably well with the(Fleming and Harrington 1991) model.

The minimal depth selection process reduced the number of variables of interest from˜17 to 13, which is still a rather large subset of interest. An obvious selection set is to examine the five variables selected by(Fleming and Harrington 1991). Combining the Minimal Depth and(Fleming and Harrington 1991) model, there may be evidence to keep the top 7 variables. Though minimal depth does not indicate the `edema` variable is very interesting, VIMP ranking does agree with the proportional hazards model, indicating we might not want to remove the `edema` variable. Both minimal depth and VIMP suggest including `copper`, a measure associated with liver disease.

Regarding the `chol` variable, recall missing data summary of Table **??**. In in the trial data set, there were 28 observations missing `chol` values. The forest imputation randomly sorts observations with missing values into daughter nodes when using the `chol` variable, which is also how **randomForestSRC** calculates VIMP. We therefore expect low values for VIMP when a variable has a reasonable number of missing values.

Restricting our remaining analysis to the five(Fleming and Harrington 1991) variables, plus the `copper` retains the biological sense of these analysis. We will now examine how these six variables are related to survival using variable dependence methods to determine the direction of the effect and verify that the log transforms used by(Fleming and Harrington 1991) are appropriate.

## 5. Variable dependence

As random forest is not parsimonious, we have used minimal depth and VIMP to reduce the number of variables to a manageable subset. Once we have an idea of which variables contribute most to the predictive accuracy of the forest, we would like to know how the response depends on these variables.

Although often characterized as a *black box* method, the forest predictor is a function of the predictor variables $\hat{f}_{RF} = f(x)$. We use graphical methods to examine the forest predicted response dependency on covariates. We again have two options, variable dependence plots (Section **??**) are quick and easy to generate, and partial dependence plots (Section **??**) are more computationally intensive but give us a risk adjusted look at variable dependence.

Variable Dependence (`gg_variable`)

*Variable dependence* plots show the predicted response relative to a covariate of interest, with each training set observation represented by a point on the plot. Interpretation of variable dependence plots can only be in general terms, as point predictions are a function of all covariates in that particular observation.

Variable dependence is straight forward to calculate, involving only the getting the predicted response for each observation. In survival settings, we must account for the additional dimension of time. We plot the response at specific time points of interest, for example survival at 1 or 3 years.

```
R> ggRFsrc + geom_vline(aes(xintercept = 1), linetype = "dashed") +
R+    geom_vline(aes(xintercept = 3), linetype = "dashed") +
R+   coord_cartesian(xlim = c(0, 5))
```

The `gg_rfsrc` of Figure 13 identical to Figure 7 (stored in the `ggRFsrc` variable) with the addition of a vertical dashed line at the 1 and 3 year survival time. A variable dependence plot is generated from the predicted response value of each survival curve at the intersecting time line plotted against covariate value for that observation. This can be visualized as taking a slice of the predicted response at each time line, and spreading the resulting points out along the variable of interest.

The `gg_variable` function extracts the training set variables and the predicted OOB response from `rfsrc` and `predict` objects. In the following code block, we store the `gg_variable` data object for later use (`gg_v`), as all remaining variable dependence plots can be constructed from this object.

```
R> gg_v <- gg_variable(rfsrc_pbc, time = c(1, 3),
R+                      time.labels = c("1 Year", "3 Years"))
R>
R> plot(gg_v, xvar = "bili", alpha = 0.4) + #, se=FALSE
R+   labs(y = "Survival", x = st.labs["bili"]) +
R+   theme(legend.position = "none") +
R+   scale_color_manual(values = strCol, labels = event.labels) +
R+   scale_shape_manual(values = event.marks, labels = event.labels) +
R+   coord_cartesian(ylim = c(-0.01, 1.01))
```

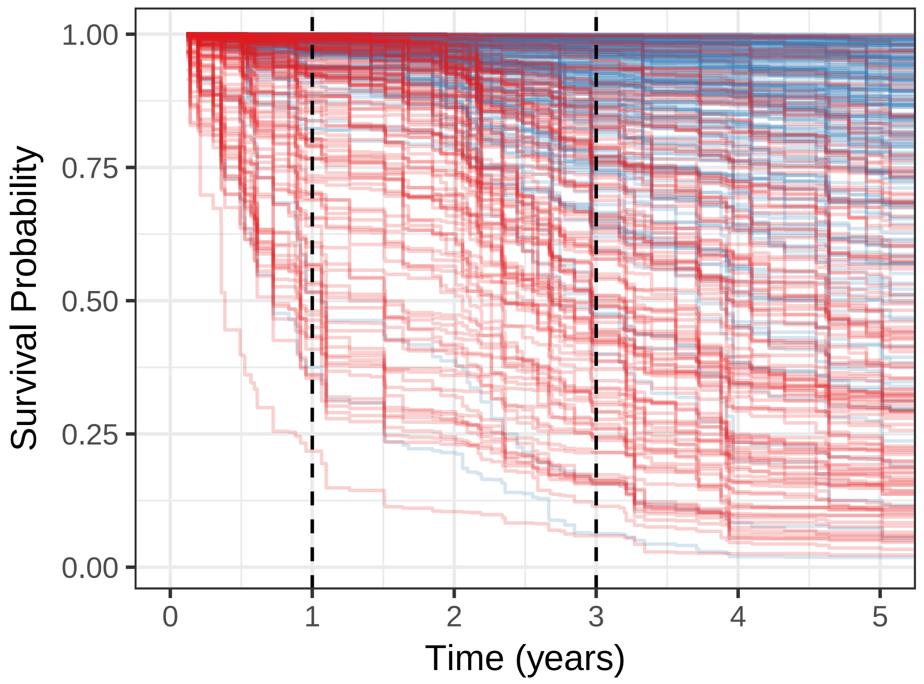


Figure 13: Random forest predicted survival (Figure 7) with vertical dashed lines indicate the 1 and 3 year survival estimates.

The `gg_variable` plot of Figure 14 shows variable dependence for the Serum Bilirubin (`bili`) variable. Again censored cases are shown as blue circles, events are indicated by the red 'x' symbols. Each predicted point is dependent on the full combination of all other covariates, not only on the covariate displayed in the dependence plot. The smooth loess line (Cleveland 1981; Cleveland and Devlin 1988) indicates the trend of the prediction over the change in the variable.

Examination of Figure 14 indicates most of the cases are grouped in the lower end of `bili` values. We also see that most of the higher values experienced an event. The "normal" range of Bilirubin is from 0.3 to 1.9 mg/dL, indicating the distribution from our population is well outside the normal range. These values make biological sense considering Bilirubin is a pigment created in the liver, the organ effected by the PBC disease. The figure also shows that the risk of death increases as time progresses. The risk at 3 years is much greater than that at 1 year for patients with high Bilirubin values compared to those with values closer to the normal range.

The `plot.gg_variable` function call operates on the `gg_variable` object controlled by the list of variables of interest in the `xvar` argument. By default, the `plot.gg_variable` function returns a list of `ggplot` objects, one figure for each variable named in `xvar`. The remaining arguments are passed to internal **ggplot2** functions controlling the display of the figure. The `se` argument is passed to the internal call to `geom_smooth` for fitting smooth lines to the data. The `alpha` argument lightens the coloring points in the `geom_point` call, making it easier to see point over plotting. We also demonstrate modification of the plot labels using the `labs` function and point attributes with the `scale_` functions.

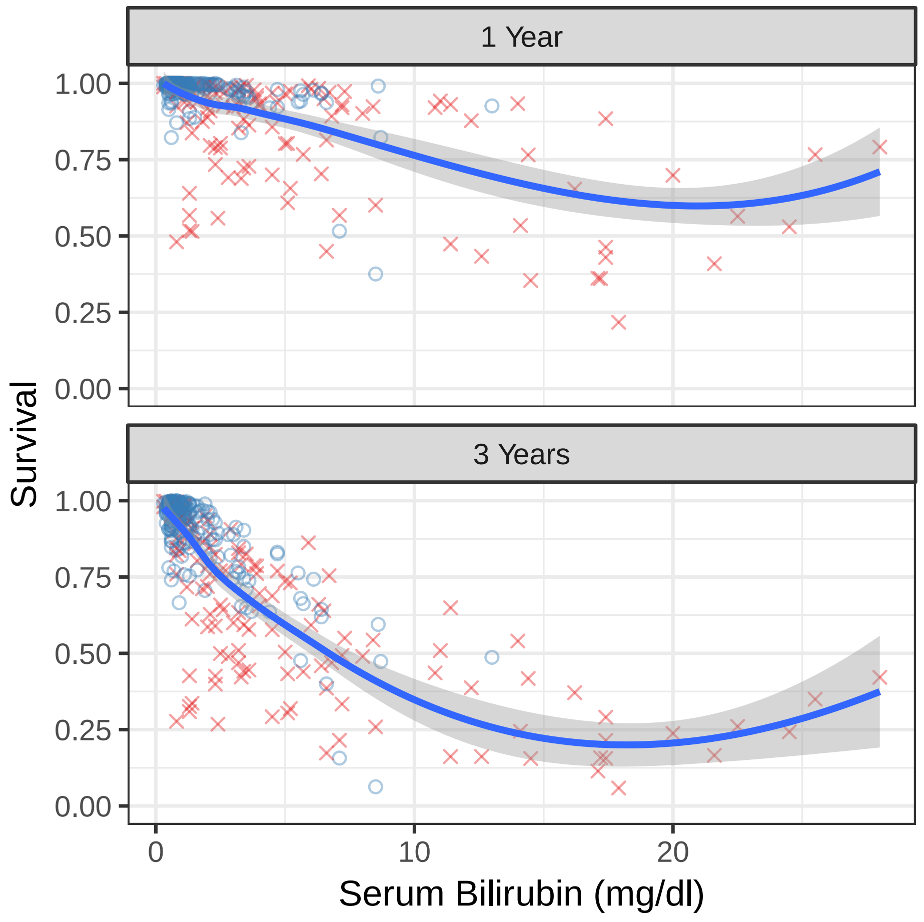


Figure 14: Variable dependence of survival at 1 and 3 years on bili variable. Individual cases are marked with blue circles (alive or censored) and red x (dead). Loess smooth curve with shaded 95% confidence band indicates decreasing survival with increasing bilirubin.

An additional `plot.gg_variable` argument (`panel = TRUE`) can be used to combine multiple variable dependence plots into a single figure. In the following code block, we plot the remaining continuous variables of interest found in Section **??**.

```
R> xvar <- c("bili", "albumin", "copper", "prothrombin", "age")
R> xvar.cat <- c("edema")
R>
R> plot(gg_v, xvar = xvar[-1], panel = TRUE, alpha = 0.4) + #se = FALSE, span=1
R+   labs(y = "Survival") +
R+   theme(legend.position = "none") +
R+   scale_color_manual(values = strCol, labels = event.labels) +
R+   scale_shape_manual(values = event.marks, labels = event.labels) +
R+   coord_cartesian(ylim = c(-0.05, 1.05))
```

The `gg_variable` plot in Figure 15 displays a panel of the remaining continuous variable dependence plots. The panels are sorted in the order of variables in the `xvar` argument and

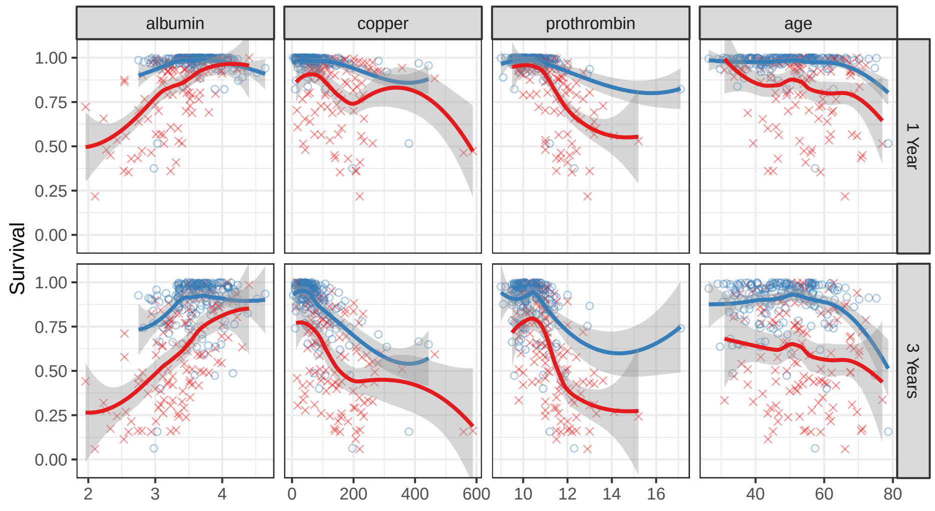


Figure 15: Variable dependence of predicted survival at 1 and 3 years on continuous variables of interest. Individual cases are marked with blue circles for censored cases and red x for death events. Loess smooth curve indicates the survival trend with increasing values.

include a smooth loess line (Cleveland 1981; Cleveland and Devlin 1988) to indicate the trend of the prediction dependence over the covariate values. The `se=FALSE` argument turns off the loess confidence band, and the `span=1` argument controls the degree of smoothing.

The figures indicate that survival increases with `albumin` level, and decreases with `bili`, `copper`, `prothrombin` and `age`. Note the extreme value of `prothrombin` ($> 16$) influences the loess curve more than other points, which would make it a candidate for further investigation.

We expect survival at 3 years to be lower than at 1 year. However, comparing the two time plots for each variable does indicate a difference in response relation for `bili`, `copper` and `prothrombine`. The added risk for high levels of these variables at 3 years indicates a non-proportional hazards response. The similarity between the time curves for `albumin` and `age` indicates the effect of these variables is constant over the disease progression.

There is not a convenient method to panel scatter plots and boxplots together, so we recommend creating panel plots for each variable type separately. We plot the categorical variable (`edema`) in Figure 16 separately from the continuous variables in Figure 15.

```
R> plot(gg_v, xvar = xvar.cat, alpha = 0.4) + labs(y = "Survival") +
R+   theme(legend.position = "none") +
R+   scale_color_manual(values = strCol, labels = event.labels) +
R+   scale_shape_manual(values = event.marks, labels = event.labels) +
R+   coord_cartesian(ylim = c(-0.01, 1.02))
```

The `gg_variable` plot of Figure 16 for categorical variable dependence displays boxplots to examine the distribution of predicted values within each level of the variable. The points are plotted with a jitter to see the censored and event markers more clearly. The boxes are shown with horizontal bars indicating the median, 75th (top) and 25th (bottom) percentiles.

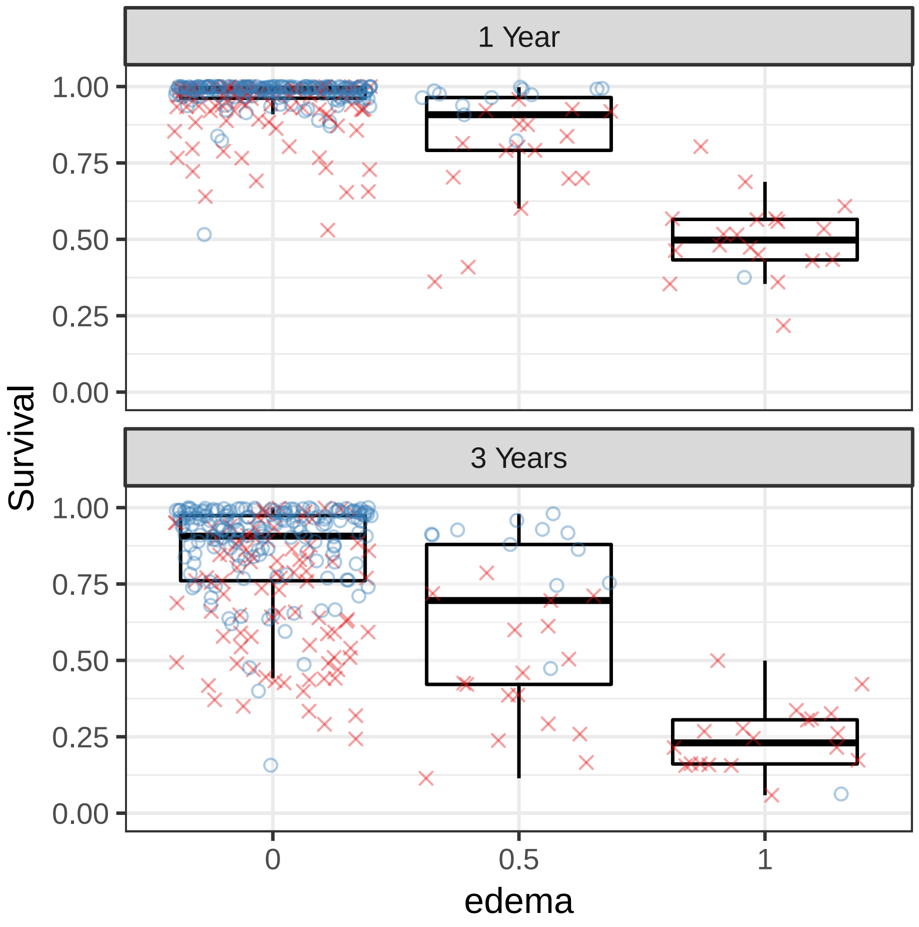


Figure 16: Variable dependence of survival 1 and 3 years on edema categorical variable. Symbols with blue circles indicate censored cases and red x indicate death events. Boxplots indicate distribution of predicted survival for all observations within each edema group.

Whiskers extend to 1.5 times the interquartile range. Points plotted beyond the whiskers are considered outliers.

When using categorical variables with linear models, we use boolean dummy variables to indicate class membership. In the case of `edema`, we would probably create two logical variables for `edema = 0.5` (complex Edema presence indicator) and `edema = 1.0` (Edema with diuretics) contrasted with the `edema = 0` variable (no Edema). Random Forest can use factor variables directly, separating the populations into homogeneous groups of `edema` at nodes that split on that variable. Figure 16 indicates similar survival response distribution between 1 and 3 year when `edema = 1.0`. The distribution of predicted survival does seem to spread out more than for the other values, again indicating a possible non-proportional hazards response.

## 5.1. Partial Dependence

*Partial dependence* plots are a risk adjusted alternative to variable dependence. Partial plots are generated by integrating out the effects of variables beside the covariate of interest. The figures are constructed by selecting points evenly spaced along the distribution of the variable of interest. For each of these points ($X = x$), we calculate the average RF prediction over all remaining covariates in the training set by

$$\tilde{f}(x) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}(x, x_{i,o}),$$

where $\hat{f}$ is the predicted response from the random forest and $x_{i,o}$ is the value for all other covariates other than $X = x$ for observation $i$(Friedman 2000).

Generating partial dependence data is effectively averaging the response for a series of nomograms constructed for each observation by varying the variable of interest. The operation is computationally intensive, especially when there are a large number of observations. The default parameters for the `plot.variable` function generate partial dependence estimates at `npts = 25` points along the variable of interest. For each point of interest, the `plot.variable` function averages the `n` response predictions. This process is repeated for each of the variables of interest.

For time to event data, we also have to deal with the additional time dimension, as with variable dependence. The following code block uses the `mclapply` function from the **parallel** package to run the `plot.variable` function for three time points (`time`=1, 3 and 5 years) in parallel. For RSF models, we calculate a risk adjusted survival estimates (`surv.type="surv"`), suppressing the internal base graphs (`show.plots = FALSE`) and store the point estimates in the `partial_pbc list`.

```
R> xvar <- c(xvar, xvar.cat)
R> partial_pbc <- mclapply(c(1,3,5), function(tm){
R+   plot.variable(rfsrc_pbc, surv.type = "surv", time = tm, xvar.names = xvar,
R+                 partial = TRUE, show.plots = FALSE)
R+   })
```

Because partial dependence data is collapsed onto the risk adjusted response, we can show multiple time curves on a single panel. The following code block converts the `plot.variable` output into a list of `gg_partial` objects, and then combines these data objects, with descriptive labels, along each variable of interest using the `combine.gg_partial` function.

```
R> gg_dta <- mclapply(partial_pbc, gg_partial)
R> pbc_ggpart <- combine.gg_partial(gg_dta[[1]], gg_dta[[2]],
R+                                  lbls = c("1 Year", "3 Years"))
```

We then segregate the continuous and categorical variables, and generate a panel plot of all continuous variables in the `gg_partial` plot of Figure 17. The panels are ordered by minimal depth ranking. Since all variables are plotted on the same Y-axis scale, those that are strongly related to survival make other variables look flatter. The figures also confirm the strong non-linear contribution of these variables. Non-proportional hazard response is also evident in at least the `bili` and `copper` variables by noting the divergence of curves as time progresses.

\begin{Schunk} \begin{Sinput} R> ggpart <- pbc_ggpart R> ggpart$edema <- NULL R> R> plot(ggpart, panel = TRUE) + #, se = FALSE R+ labs(x = “”, y = “Survival”, color =

"Time", shape = "Time") + R+ theme(legend.position = c(0.8, 0.2)) + R+ coord_cartesian(ylim = c(25, 101)) \end{Sinput}

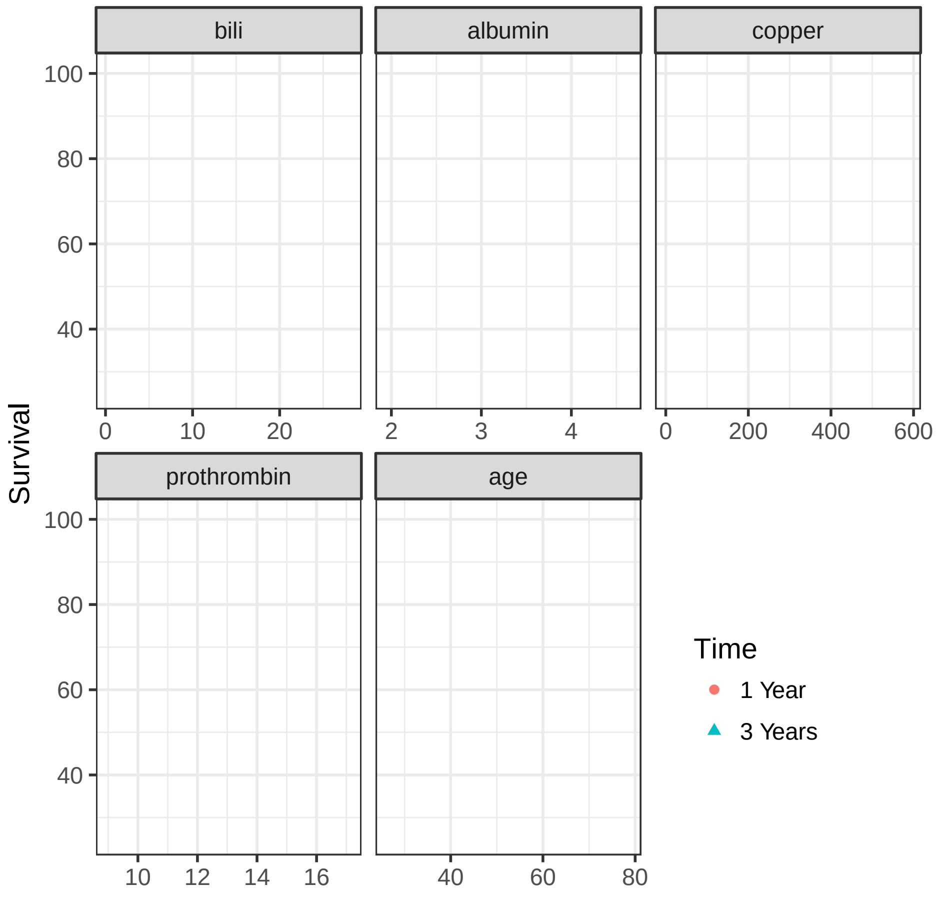


Figure 17: Partial dependence of predicted survival at 1 year (red circle) and 3 years (blue triangle) as a function continuous variables of interest. Symbols are partial dependence point estimates with loess smooth line to indicate trends.

\end{Schunk}

Categorical partial dependence is displayed as boxplots, similar to categorical variable dependence. Risk adjustment greatly reduces the spread of the response as expected, and may also move the mean response compared to the unadjusted results. The categorical `gg_partial` plot of Figure 18 indicates that, adjusting for other variables, survival decreases with rising `edema` values. We also note that the risk adjusted distribution does spread out as we move further out in time.

```
R> ggplot(pbc_ggpart[["edema"]], aes(y=yhat, x=edema, col=group))+
R+   geom_boxplot(notch = TRUE,
R+                outlier.shape = NA) + # panel=TRUE,
R+   labs(x = "Edema", y = "Survival (%)", color="Time", shape="Time") +
R+   theme(legend.position = c(0.2, 0.2)) +
R+    coord_cartesian(ylim = c(25, 101))
```

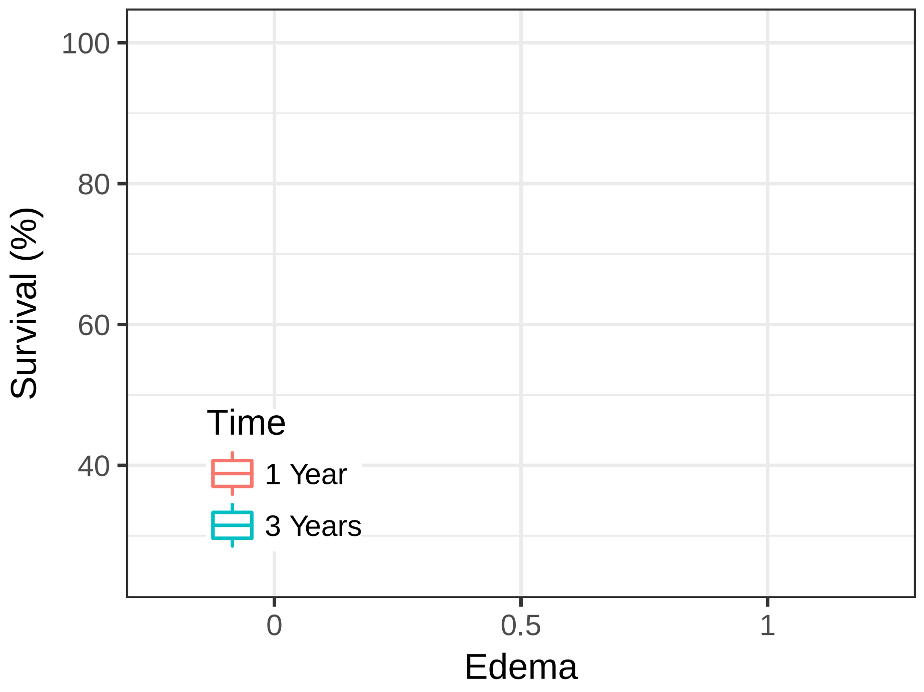


Figure 18: Partial dependence plot of predicted survival at 1 year (red) and 3 years (blue) as a function of edema groups (categorical variable). Boxplots indicate distribution within each group.

Partial dependence is an extrapolation operation. By averaging over a series of nomograms, the algorithm constructs observations for all values of the variable of interest, regardless of the relation with other variables. In contrast, variable dependence only uses observations from within the training set. A simple example would be for a model including BMI, weight and height. When examining partial dependence of BMI, the algorithm only manipulates BMI values, height or weight values. The averaging operation is then confounded in two directions. First, dependence on height and weight is shared with BMI, making it difficult to see the true response dependence. Second, partial dependence is calculated over nomograms that can not physically occur. For simple variable combinations, like BMI, it is not difficult to recognize this and modify the independent variable list to avoid these issues. However, care must be taken when interpreting more complex biological variables.

### 5.2. Partial dependence as a function of time

In the previous section, we calculated risk adjusted (partial) dependence at two time points (1 and 3 years). The selection of these points can be driven by biological times of interest (i.e., 1 year and 5 year survival in cancer studies) or by investigating time points of interest from a `gg_rfsrc` prediction plot. We typically restrict generating `gg_partial` plots to the variables of interest at two or three time points of interest due to computational constraints.

It is instructive to see a more detailed map of the risk adjusted response to get a feel for interpreting partial and variable dependence plots. In Figure 17, we can visualize the two curves as extending into the plane of the page along a time axis. Filling in more partial dependence curves, it is possible to create a partial dependence surface.

For this exercise, we will generate a series of 50 `gg_partial` plot curves for the `bili` variable. To fill the surface in, we also increased the number of points along the distribution of `bili` to `npts=50` to create a grid of $50 \times 50$ risk adjusted estimates of survival along time in one dimension and the `bili` variable in the second.

## References

R Core Team. 2014. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. `http://www.R-project.org/`.

Breiman, L. 1996a. "Bagging Predictors." *Machine Learning* 26: 123–40.

———. 1996b. "Out–Of–Bag Estimation." Statistics Department, University of California,Berkeley, CA. 94708. `https://www.stat.berkeley.edu/~breiman/OOBestimation.pdf`.

Breiman, L, Jerome H Friedman, R Olshen, and C Stone. 1984. *Classification and Regression Trees.* Monterey, CA: Wadsworth; Brooks.

Breiman, Leo. 2001a. "Random Forests." *Machine Learning* 45 (1). Kluwer Academic Publishers, Boston: 5–32.

———. 2001b. "Statistical Modeling: The Two Cultures." *Statistical Science* 16 (3): 199–231.

Chambers, J. M. 1992. *Statistical Models in S.* Wadsworth & Brooks/Cole.

Cleveland, William S. 1981. "LOWESS: A Program for Smoothing Scatterplots by Robust Locally Weighted Regression." *The American Statistician* 35 (1): 54.

———. 1993. *Visualizing Data.* Summit Press.

Cleveland, William S., and Susan J. Devlin. 1988. "Locally-Weighted Regression: An Approach to Regression Analysis by Local Fitting." *Journal of the American Statistical Association* 83 (403): 596–610.

Efron, Bradley, and Robert Tibshirani. 1994. *An Introduction to the Bootstrap.* Chapman & Hall/CRC.

Fleming, Thomas R., and David P. Harrington. 1991. *Counting Processes and Survival Analysis.* John Wiley & Sons, New York.

Friedman, Jerome H. 2000. "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics* 29: 1189–1232.

Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Second. New York: Springer-Verlag.

Ishwaran, Hemant. 2007. "Variable Importance in Binary Regression Trees and Forests." *Electronic Journal of Statistics* 1: 519–37.

Ishwaran, Hemant, and Udaya B. Kogalur. 2007. "Random Survival Forests for R." *R News* 7 (2): 25–31.

———. 2010. "Consistency of Random Survival Forests." *Statistics and Probability Letters*

80: 1056–64.

———. 2014. "Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.6." http://CRAN.R-project.org/package=randomForestSRC.

Ishwaran, Hemant, Udaya B. Kogalur, Eugene H. Blackstone, and Michael S. Lauer. 2008. "Random Survival Forests." *The Annals of Applied Statistics* 2 (3): 841–60.

Ishwaran, Hemant, Udaya B. Kogalur, Xi Chen, and Andy J. Minn. 2011. "Random Survival Forests for High–Dimensional Data." *Statist. Anal. Data Mining* 4: 115–32.

Ishwaran, Hemant, Udaya B. Kogalur, Eiran Z. Gorodeski, Andy J. Minn, and Michael S. Lauer. 2010. "High–Dimensional Variable Selection for Survival Data." *J. Amer. Statist. Assoc.* 105: 205–17.

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by RandomForest." *R News* 2 (3): 18–22.

Rubin, D.B. 1976. "Inference and Missing Data." *Biometrika* 63: 581–92.

Tukey, John W. 1977. *Exploratory Data Analysis.* Pearson.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis.* New York: Springer-Verlag.

**Affiliation:**

John Ehrlinger
Microsoft
First line Second line
E-mail: john.ehrlinger@gmail.com
URL: https://github.com/ehrlinger/ggRandomForests

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:*
MMMMMM YYYY *Accepted:* yyyy-mm-dd