ggRandomForests: Random Forests for Survival

John Ehrlinger and **Eugene H. Blackstone**
Cleveland Clinic

Abstract

Random Forests (Breiman 2001) (RF) are a fully non-parametric statistical method requiring no distributional assumptions on covariate relation to the response. RF are a robust, nonlinear technique that optimizes predictive accuracy by fitting an ensemble of trees to stabilize model estimates. Random Forests for survival (Ishwaran and Kogalur 2007; Ishwaran, Kogalur, Blackstone, and Lauer 2008) (RF-S) are an extension of Breiman's RF techniques to survival settings, allowing efficient non-parametric analysis of time to event data. The **randomForestSRC** package (Ishwaran and Kogalur 2014) is a unified treatment of Breiman's random forests 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, tools for creating and plotting data structures to visually understand random forest models grown in R with the **randomForestSRC** 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 formatted as a tutorial for using the **randomForestSRC** for building random forests for survival and **ggRandomForests** package for investigating how the forest is constructed. This tutorial uses the Primary Biliary Cirrhosis (PBC) Data from the Mayo Clinic (Fleming and Harrington 1991) available in the **randomForestSRC** package. We use Variable Importance measure (VIMP) (Breiman 2001) as well as Minimal Depth (Ishwaran, Kogalur, Gorodeski, Minn, and Lauer 2010a), a property derived from the construction of each tree within the forest, to assess the impact of variables on forest prediction. We will also demonstrate the use of variable dependence plots (Friedman 2000a) to aid interpretation RF results in different response settings. We also will investigate interactions between covariates to demonstrate the strength of the Random Forest method in survival settings.

Keywords: random forest, survival, VIMP, minimal depth, R, **randomForestSRC**.

1. About this document

This document is an introduction to the **ggRandomForests** R package. The aim of this introduction is to provide a detailed user guide to **ggRandomForests** as well as provide a tutorial to building a Random Forest Survival model with the **randomForestSRC** package. Our attempt is to build simple, reproducible worked examples with the Primary Biliary Cirrhosis (PBC) Data from the Mayo Clinic.

This document is available as a vignette within **ggRandomForests** package. The latest version

is available from the Comprehensive R Archive Network via http://CRAN.R-project.org/package=ggRandomForests.

2. Introduction

Random Forests (Breiman 2001) (RF) are a robust, non-parametric statistical method that optimizes predictive accuracy by averaging an ensemble of tree models. Random Forests are not parsimonious, utilizing all provided variables in predicting the specified outcome. It does not require prior knowledge of the parametric relation of variables (linearity or non-linearity) to the response, or of interactions between variables. RF chooses the most important variables by assessing variable impact on the predictive ability of the forest of trees.

A Random Forest is built up by bagging (Breiman 1996a) a collection of classification and regression trees (Breiman, Friedman, Olshen, and Stone 1984) (CART). The method uses a set of B bootstrap (Efron and Tibshirani 1994) samples, growing a set of independent tree models on each sub-sample of the population. Trees are 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 chosen for the splitting. Each node is split into two daughter nodes by maximizing the separation of observations according the split rule. In regression trees, node impurity is measured by mean squared error, whereas in classification problems, the Gini index is used (Friedman 2000b). Each subsequent daughter node is then split until the process reaches the stopping criteria of either node purity or node member size defining the set of terminal (unsplit) nodes for the tree. Random Forests sort each observation into one unique terminal node per tree. The Random Forest estimate for each observation is calculated by aggregation, averaging (regression) or votes (classification), the terminal node results across the collection of B trees.

One advantage of Random Forests 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 (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 Forests allows us to obtain both model fit and validation in one pass of the algorithm.

2.1. Random Forests for Survival

Random Forests for survival (Ishwaran 2007; Ishwaran *et al.* 2008) (RF-S) are an extension of Breiman (2001) Random Forests for 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 Forests for survival adaptively discover nonlinear effects and interactions and are fully nonparametric. Averaging over trees, with randomizing while growing a tree, enables RF-S to approximate complex survival functions, including non-proportional hazards, while maintaining low prediction error. Ishwaran and Kogalur (2010) showed that RF-S is uniformly consistent and that survival forests have a uniform approximating property in finite-sample

settings, a property not possessed by individual survival trees.

2.2. ggRandomForests

The **randomForestSRC** package is a mature analysis and research random forest implementation under rapid development. The package includes diagnostic and post processing functions for analysis and visualizations of randomForest model properties. However, in our research we frequently found it difficult to manipulate the standard figures directly produced with the **randomForestSRC** package.

In order to simplify these manipulations, we developed the **ggRandomForests** package. We attempted to follow two design principles in this development:

- Model/View separation: The package originally designed to generating **ggplot2** Wickham (2009) figures for random forest objects. However, some users would prefer to use other graphing methods within R or outside of it. To help users, we separate the data generation and the figure generation into two separate operations.
- Modular: We strive to create a modular design by following the *do one thing well* philosphy. Each function operates on one **randomForestSRC** object to create only one data object or figure type.

To demonstrate using the **ggRandomForests** package, we organize this document as follows. In Section **??** we outline growing a random forest for each of the classification, regression and survival settings with the **randomForestSRC** package. We use the **ggRandomForests** package to begin exploring random forest convergence and prediction. In Section **??** we discuss how variables contribute to the random forest prediction using the Variable Importance (VIMP) and Minimal Depth measures.

Once we have an idea which variables are most informative in minimizing forest prediction error, we turn our focus to how the variables are related to the forest prediction. Because Random Forests are non-linear and non-parametric predictors, we can use variable dependence (Section 6.1) to examine where each observation contributes to model prediction as a function of specific covariate values. Partial dependence (Section 6.2) gives us a risk adjust view of the predictor dependence on a variable. We then find two way interactions using minimal depth in Section 7 and use conditional plots in Section **??** to look variable interactions in an intuitive manner.

3. Data Summary: Primary Biliary Cirrhosis (PBC) Data

Data from the Mayo Clinic trial in primary biliary cirrhosis (PBC) of the liver conducted between 1974 and 1984. A total of 424 PBC patients, referred to Mayo Clinic during that ten-year interval, met eligibility criteria for the randomized placebo controlled trial of the drug D-penicillamine. The first 312 cases in the data set participated in the randomized trial and contain largely complete data.

4. Growing the Random Forest

	label	type
status	censoring indicator	logical
treatment	1 = D-penicillamine, 2 = placebo	factor
age	age in years	numeric
sex	0 = male, 1 = female	logical
ascites	presence of asictes	logical
hepatom	presence of hepatomegaly	logical
spiders	presence of spiders	logical
edema	presence of edema	factor
bili	serum bilirubin in mg/dl	numeric
chol	serum cholesterol in mg/dl	integer
albumin	albumin in gm/dl	numeric
copper	urine copper in ug/day	integer
alk	alkaline phosphatase in U/liter	numeric
sgot	SGOT in U/ml	numeric
trig	triglicerides in mg/dl	integer
platelet	platelets per cubic ml/1000	integer
prothrombin	prothrombin time in seconds	numeric
stage	histologic stage of disease	factor
years	survival time in years	numeric

Table 1: PBC Data field descriptions

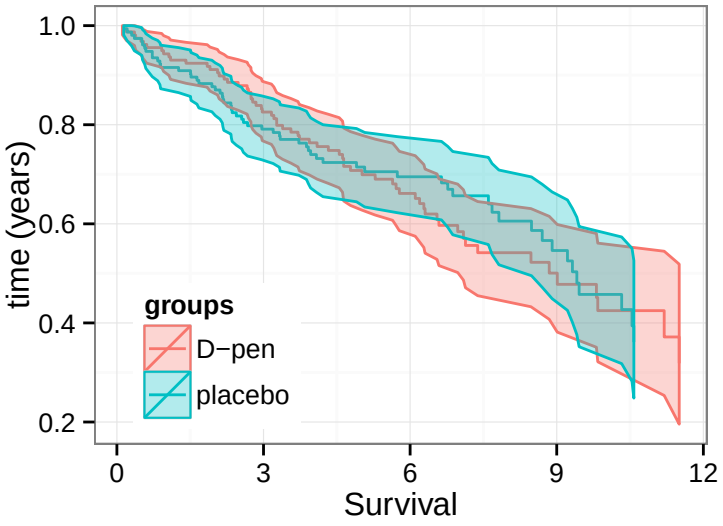

Figure 1: Kaplan-Meier survival estimates

```
R> pbc_rf <- rfsrc(Surv(years, status) ~ ., data = pbc,
+                  ntree = 500,nsplit = 10,
+                  na.action = "na.impute")

                         Sample size: 418
                    Number of deaths: 161
                    Was data imputed: yes
                         Missingness: 33.97%
```

```
                  Number of trees: 500
       Minimum terminal node size: 3
     Average no. of terminal nodes: 78.368
No. of variables tried at each split: 5
            Total no. of variables: 17
                         Analysis: RSF
                           Family: surv
                 Splitting rule: logrank *random*
     Number of random split points: 10
                       Error rate: 16.72%
```

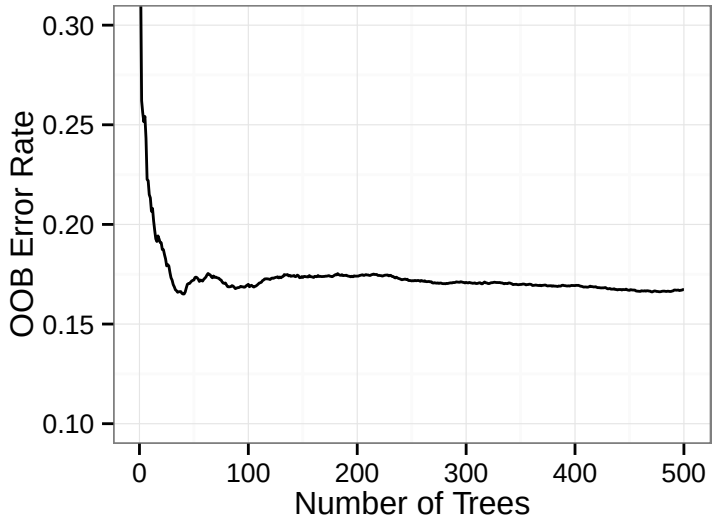

Figure 2: RSF prediction error estimates

Figure 3 shows the predicted survival from an RF-S model, where censored device prediction is colored in blue, and devices experiencing an event are colored in red.

4.1. Forest Imputation for missing values

The randomForests package (Liaw and Wiener 2002) include a forest imputation method within the randomForest package.

We impute missing data (both x and y-variables) using a modification of the missing data algorithm of Ishwaran *et al.* (2008). Prior to splitting a node, missing data for a variable is imputed by randomly drawing values from non-missing in-bag data. The purpose of the imputed data is to make it possible to assign cases to daughter nodes in the event the node is split on a variable with missing data. Imputed data is however not used to calculate the split-statistic which uses non-missing data only. Following a node split, imputed data are reset to missing and the process is repeated until terminal nodes are reached. Missing data is then imputed using OOB non-missing terminal node data. For integer valued variables and censoring indicators, imputation uses a maximal class rule, whereas continuous variables and survival time use a mean rule.

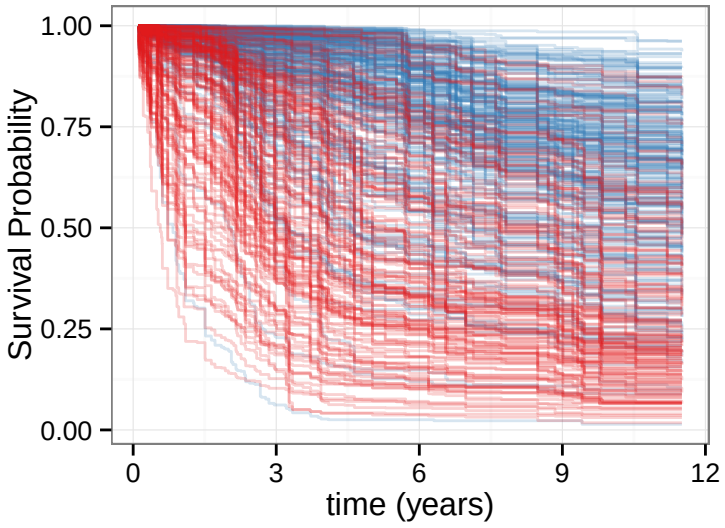

Figure 3: PBC Survival

The proximity matrix from the randomForest is used to update the imputation of the NAs. For continuous predictors, the imputed value is the weighted average of the non-missing obervations, where the weights are the proximities. For categorical predictors, the imputed value is the category with the largest average proximity. This process is iterated iter times.

Regardless of what method is used, records in which all outcome and x-variable information are missing are removed from the forest analysis. Variables having all missing values are also removed.

5. Variable Selection

Unlike in the linear model settings, Random Forests does not require explicitly specify the functional form of the covariates to the response. Instead, we ascertain which variables contribute to the Random Forest estimates by querying the forest for variable usage.

5.1. Variable Importance

Unlike in the linear model settings, Random Forests does not require explicitly specify the functional form of the covariates to the response. Instead, we ascertain which variables contribute to the Random Forest estimates by querying the forest for variable usage.

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 to date, adopts a prediction error approach involving "noising-up" a variable. VIMP for a variable x_v is the difference between prediction error when x_v is noised up by permuting its value randomly, compared to prediction error under the original predictor (Breiman 2001; Liaw and Wiener 2002; Ishwaran 2007; Ishwaran *et al.* 2008).

Since VIMP is the absolute difference between prediction errors before and after permutation,

a large VIMP value indicates that misspecification of that variable detracts from the predictive accuracy of 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 mispecified. In the later case, we assume noise is more informative than the 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.

In Figure 4, we plot VIMP measures for each of the variables used to grow the forest estimates of Figure 3. Variables are shown in VIMP rank order, largest (op_yr) at the top, to smallest (iv_lospr) at the bottom. In this case, we would focus attention on the top three variables (op_yr (surgical date), ld and devno).

```
R> plot.gg_vimp(pbc_rf) +
+   theme(legend.position = "none")
```

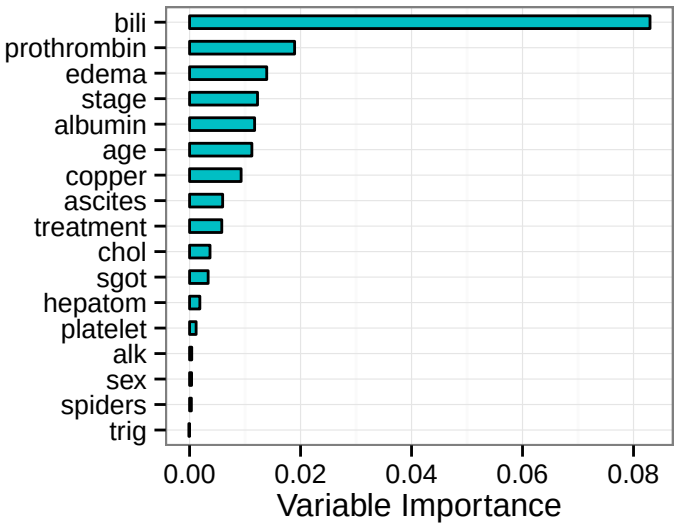

Figure 4: Variable Importance

5.2. Minimal Depth

In VIMP, prognostic risk factors are determined by inspection of the forest, ranking the most important variables according to impact on predictive ability of the forest. An alternative method recognizes that most important variables for prediction are those that most frequently split nodes nearest to the trunks of the trees (ie, at the root node) since they partition the largest portions of the population.

Node levels are numbered based on their relative distance to the trunk of the tree (ie. 0, 1, 2). A measure of important risk factors is determined by averaging the depth of first split for each variable over all trees within the forest. Lower values of this measure indicate variables that split larger groups of patients.

The maximal subtree for a variable x is the largest subtree whose root node splits on x. Thus, all parent nodes of x's maximal subtree have nodes that split on variables other than x. The largest maximal subtree possible is the root node. In general, however, there can be more than one maximal subtree for a variable. A maximal subtree may also not exist if there are no splits on the variable. The minimal depth of a maximal subtree (the first order depth) measures predictiveness of a variable x. It equals the shortest distance (the depth) from the root node to the parent node of the maximal subtree (zero is the smallest value possible). The smaller the minimal depth, the more impact x has on prediction. The mean of the minimal depth distribution is used as the threshold value for deciding whether a variable's minimal depth value is small enough for the variable to be classified as strong.

The minimal depth plot of Figure **??** is similar to the VIMP plot in Figure 4, ranking variables from most important at the top (minimal depth measure), to least at the bottom (maximal minimal depth). Since the VIMP and Minimal Depth measures use different criteria, we expect the variable ranking to be slightly different. In this case, minimal depth indicates seven most important variables (op_yr (surgical date), age, ld, ht, wt, iv_lospr (length of stay) and inr). The vertical dashed line indicates the minimal depth threshold where smaller minimal depth values indicate higher importance and larger indicate lower importance.

```
R> pbc_vs <- var.select(pbc_rf)
R> ggMindepth <- gg_minimal_depth(pbc_vs)
R> print(ggMindepth)

-----------------------------------------------------------
gg_minimal_depth
model size         : 12
depth threshold    : 5.9439

PE :[1] 16.724
-----------------------------------------------------------

Top variables:
            depth  vimp
bili        1.548 0.083
prothrombin 2.454 0.019
albumin     2.512 0.012
edema       2.806 0.014
copper      2.970 0.009
age         3.032 0.011
stage       3.346 0.012
chol        3.410 0.004
platelet    3.570 0.001
sgot        3.958 0.003
alk         4.296 0.000
trig        4.594 0.000
-----------------------------------------------------------

R> plot(ggMindepth)
```

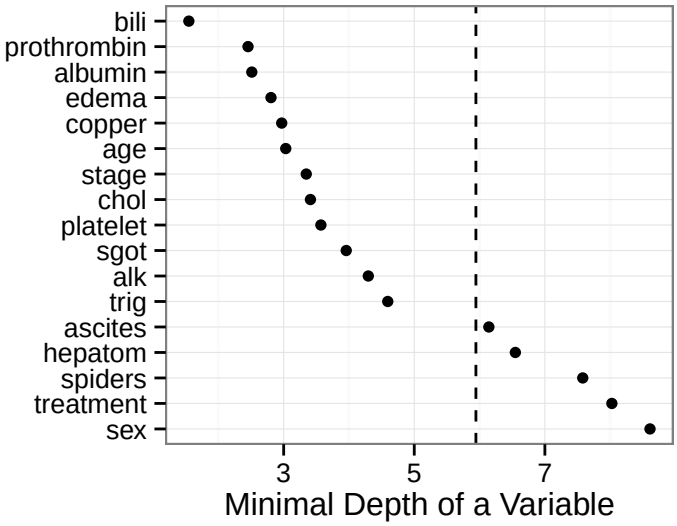

Figure 5: Minimal Depth Plot

6. Variable Dependence

Once we have an idea of which variables contribute to the predictive accuracy of the forest, it is useful to get some idea of form of this contribution. We use graphical methods to show the predicted response given dependence on covariates. We can plot the marginal effect of an covariate on the class probability (classification), response (regression), mortality (survival), or the expected years lost (competing risk) for a RF analysis. We plot the ensemble predicted value on the vertical axis and covariates along the horizontal axis.

6.1. Marginal Dependence

Marginal variable dependence plots the predicted response as a function of the covariate, showing each subject as a point on the plot. For classification and regression, this is straight forward predicting the response. In survival settings, we must account for the additional dimension of time. In this case, we plot the response at a specific time point of interest, for example survival at three months shown by the vertical dashed line in Figure 6. We take the predicted value of each curve at that time, and plot that against the covariate value for that observations, shown in Figure **??**. Again censored cases are shown in 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, so interpretation of these variable dependence plots can only be in general terms. The smooth loess line (Cleveland 1981; Cleveland and Devlin 1988) indicates the trend of the prediction over surgical date progression.

```
R> ggRFsrc +
+   geom_vline(aes(xintercept = c(1, 3)), linetype = "dashed") +
+   coord_cartesian(x = c(0, 4))
```

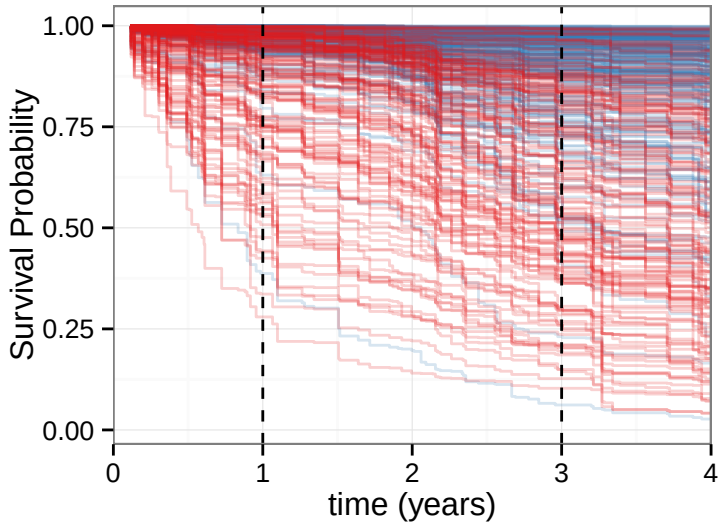

Figure 6: PBC Survival

```
R> xvar <- pbc_vs$topvars[1:6]
R> ind = 1
R> ggrf <- gg_variable(pbc_rf, time = c(1, 3),
+                      time.labels = c("1 Year", "3 Years"))
R>
R> plot(ggrf, x_var = xvar[ind], se=FALSE, alpha=.3) +
+   labs(y = "Survival") +
+   theme(legend.position = "none") +
+   scale_color_manual(values = strCol, labels = event.labels) +
+   scale_shape_manual(values = event.marks, labels = event.labels)


R> plot(ggrf, x_var = xvar[c(2,3,5,6)], panel = TRUE,
+      se=FALSE, alpha=.3,
+      method="glm", formula=y~poly(x,2)) +
+   labs(y = "Survival") +
+   theme(legend.position = "none") +
+   scale_color_manual(values = strCol, labels = event.labels) +
+   scale_shape_manual(values = event.marks, labels = event.labels)+
+   coord_cartesian(y=c(1,102))
```

6.2. Partial Dependence

Partial dependence plots are a risk adjusted alternative to marginal 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 X variable. For each of these values (X = x), we calculate the average Random Forest

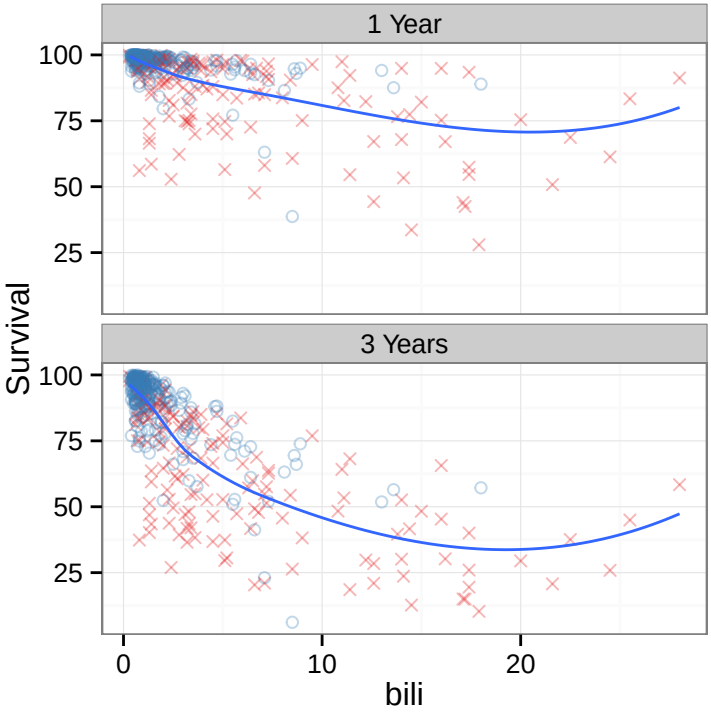

Figure 7: Variable dependence Survival vs. Bilirubin

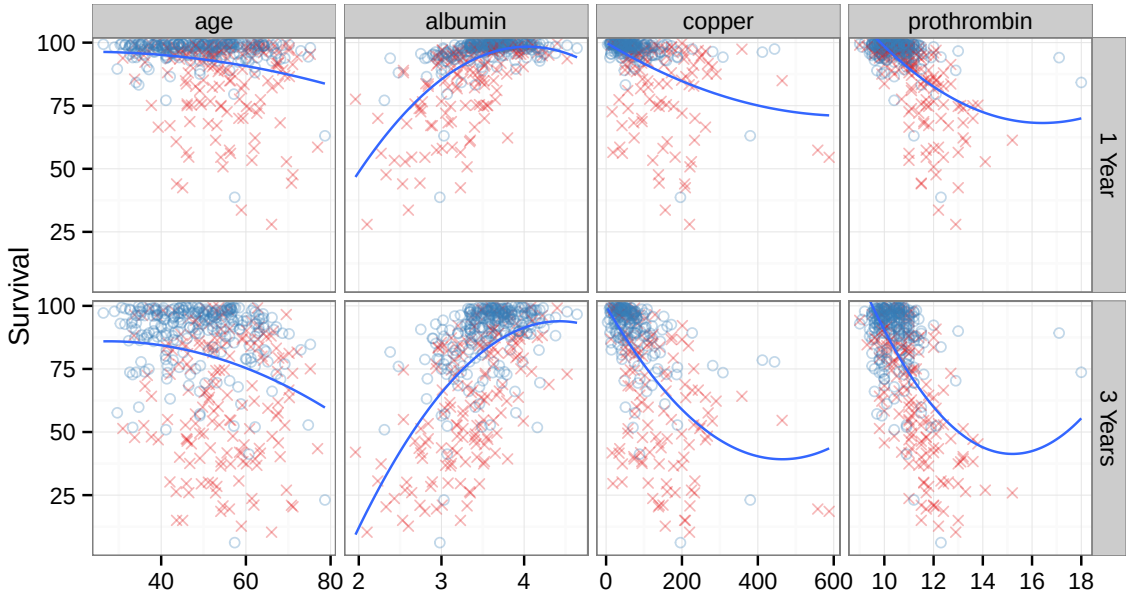

Figure 8: Variable dependence Survival panel

prediction over all other covariates in X by (1).

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

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 the observation i (Friedman 2000b). Partial dependence plots in time to event settings are shown at specific time points, similar to variable dependence. Figure 9 shows the partial dependence of three month survival on bilirubin.

```
R> # Calculate the 1, 3 and 5 year partial dependence
R> pbc_prtl_time <- mclapply(c(1,3,5), function(tm){
+   plot.variable(pbc_rf, surv.type = "surv",
+                           time = tm,
+                           xvar.names = xvar, partial = TRUE,
+                           show.plots = FALSE)
+ })
```

```
R> # Convert all partial plots to gg_partial objects
R> gg_dta <- lapply(pbc_prtl_time, gg_partial)
R>
R> # Combine the objects to get multiple time curves
R> # along variables on a single figure.
R> pbc_ggpart <- combine(gg_dta[[1]],gg_dta[[2]],
+                        lbls = c("1 Year", "3 Years"))
R>
R> plot(pbc_ggpart[["bili"]], se = FALSE) +
+   theme(legend.position = c(.8, .5)) +
+   labs(y = "Survival",
+        x = dta.labs["bili", "label"])+
+   scale_color_brewer(palette="Set1")
```

Non-proportional hazards are evident in Figure 9.

```
R> ggpart <- pbc_ggpart
R> ggpart$bili <- ggpart$edema <- NULL
R> plot(ggpart, se = FALSE, panel = TRUE) +
+   labs(x = "", y = "Survival") +
+   scale_color_brewer(palette="Set1")
```

```
R> plot(pbc_ggpart$edema, notch=TRUE, alpha=.3) +
+   labs(x = "Edema", y = "Survival (%)")+
+   scale_color_brewer(palette="Set1")+
+   facet_grid(~group)
```

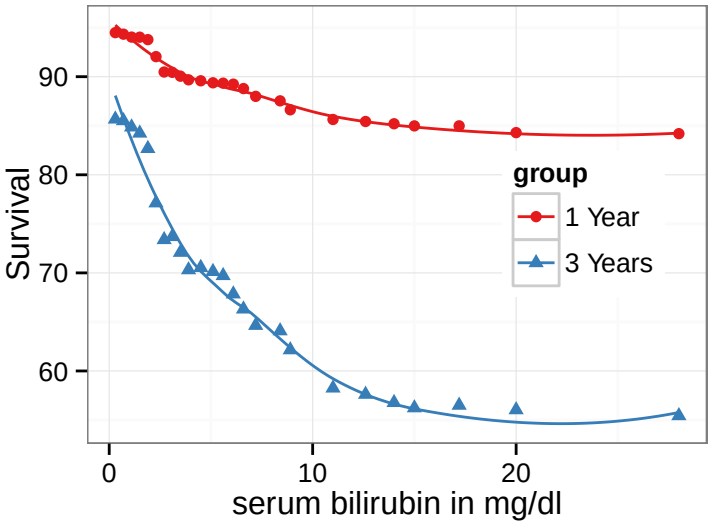

Figure 9: Risk adjusted Survival

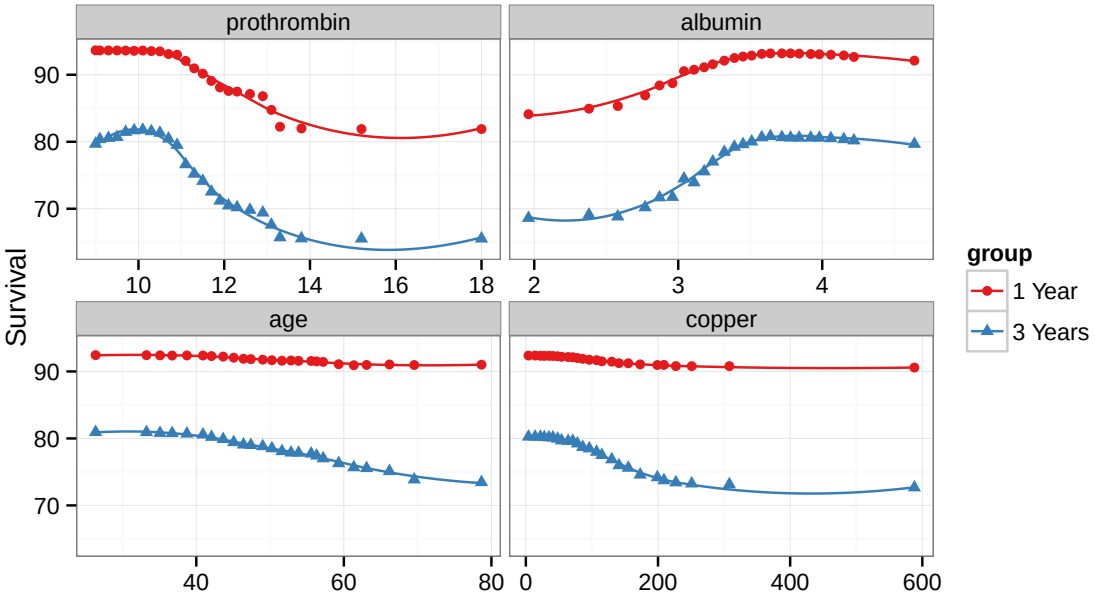

Figure 10: Risk adjusted Survival - panel plot

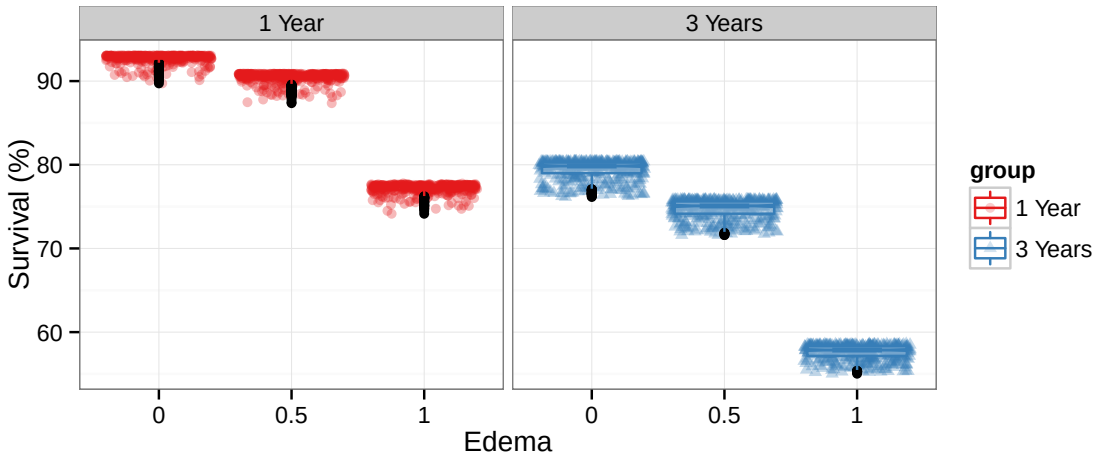

Figure 11: Risk adjusted Survival - Edema

7. Variable Interactions

Using the different variable dependence measures, we can calculate pairwise interactions for any pair of variables. Minimal depth is calculated as the maximal subtree using the normalized minimal depth of variable i relative to the root node (normalized with respect to the size of the tree). For interactions, we calculate the maximal subtree interaction measure as the normalized minimal depth of a variable j with respect to the maximal subtree for variable i (normalized with respect to the size of i's maximal subtree) (Ishwaran, Kogalur, Gorodeski, Minn, and Lauer 2010b; H., U.B., X., and A.J. 2011).

```
R> pbc_interaction <- find.interaction(pbc_rf)
R>
R> ggint <- gg_interaction(pbc_interaction)
R> plot(ggint, x_var = "bili") +
+   labs(y = "Interactive Minimal Depth")
```

Measuring interactions with minimal depth results a $p \times p$ matrix of interaction measures, with smaller diagonal measures relative to the root node, and off diagonal measures of pairwise interaction. We expect the covariate with smallest minimal depth to have the highest interactive depth measures, so viewed alone may not be as informative as looking at other interactive depth plots. Figure 13 combines the remaining top ranked minimal depth measures for comparison.

```
R> plot(gg_interaction(pbc_interaction), x_var = xvar[2:5]) +
+   labs(y = "Interactive Minimal Depth") +
+   theme(legend.position = "none")
```

7.1. Conditional Dependence Plots

By plotting the resulting interaction measures for each variable (Figure 12), we can detect the "most interactive" pairs, and develop conditional plots Chambers (1992); Cleveland (1993).

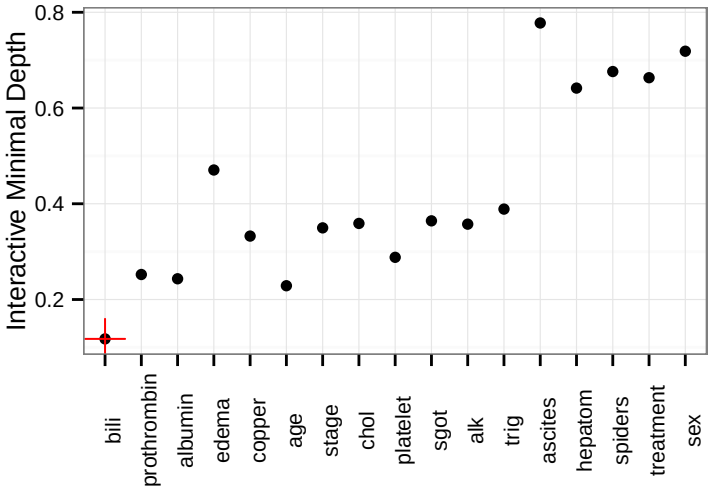

Figure 12: Minimal Depth interaction for Surgical Date

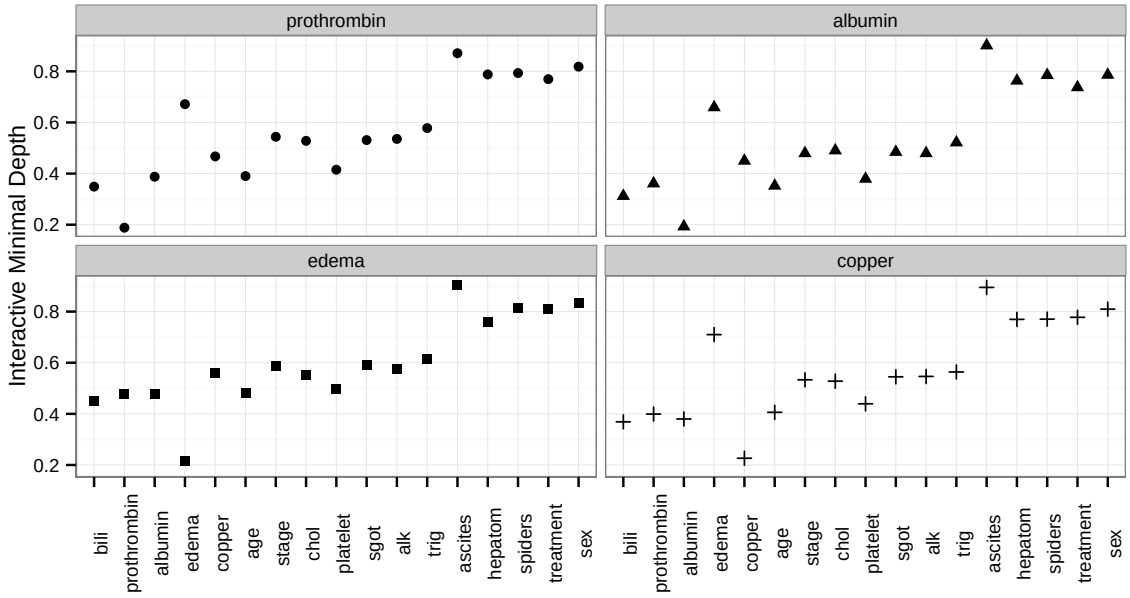

Figure 13: Risk adjusted Survival - panel plot

These plots are similar to stratified results, arranged in a set of panels by the interactive variable of interest.

Interactions with categorical data are more straight forward, and can be generated directly from variable dependence plots. Recall the 1 year variable dependence for Billirubin, shown in Figure 14.

```
R> ggrf <- gg_variable(pbc_rf, time = 1)
R>
R> ggvar <- ggrf
R> ggvar$treatment <- as.numeric(ggvar$treatment)
R> ggvar$treatment[which(ggvar$treatment==1)] <- "D-pen"
R> ggvar$treatment[which(ggvar$treatment==2)] <- "placebo"
R> ggvar$treatment <- factor(ggvar$treatment)
R>
R> ggvar$stage <- paste("stage=", ggvar$stage, sep="")
R>
R> var_dep <- plot(ggvar, x_var = "bili", smooth = TRUE,
+                  method = "loess", span=1.5,alpha = .5, se = FALSE) +
+   labs(y = "Survival",
+        x = dta.labs["bili", "label"]) +
+   theme(legend.position = "none") +
+   scale_color_manual(values = strCol, labels = event.labels) +
+   scale_shape_manual(values = event.marks, labels = event.labels)
R>
R> show(var_dep)
```

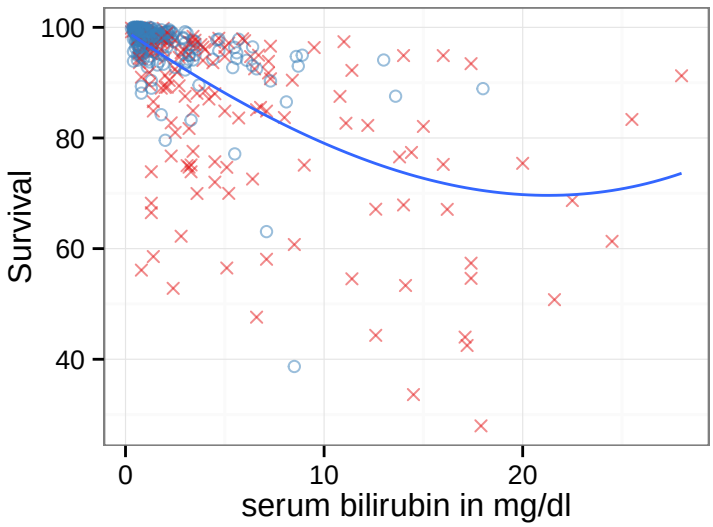

Figure 14: Bilirubin Variable Dependence at 1 year.

We can view the conditional dependence of survival against bilirubin, versus other categorical covariates, say treatment (binary) and stage (categorical), by adding a facet argument.

```
R> var_dep +
+   facet_grid(treatment~stage)
```

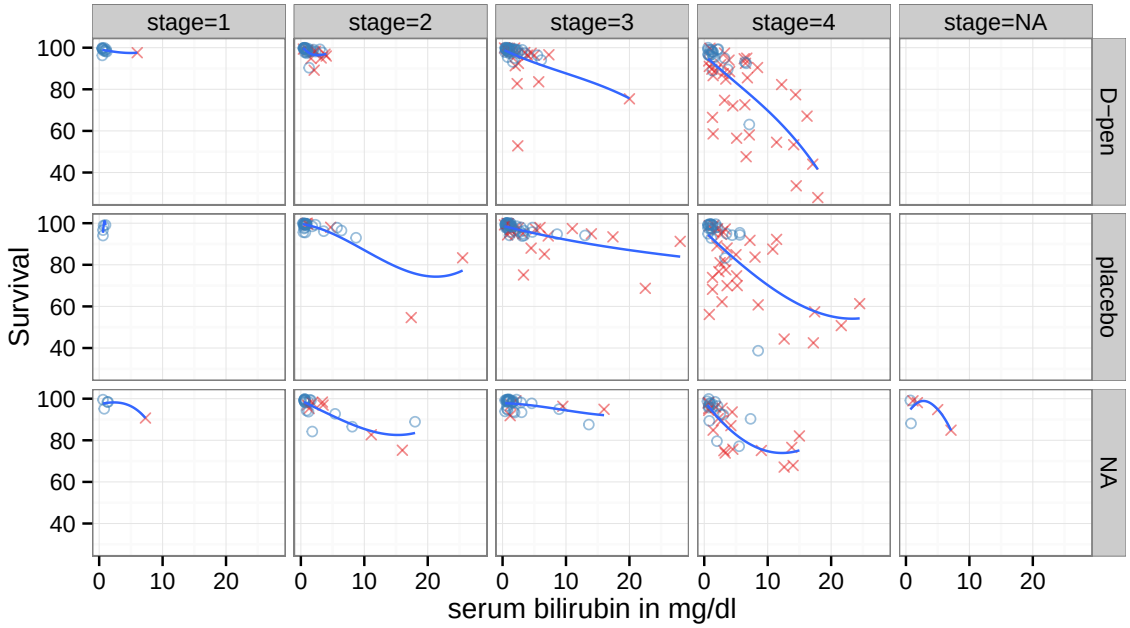

Figure 15: Conditional Variable Dependence. Interactions between bilrubin with treatment and stage variables.

Interactions with continuous variables requires stratification at some level.

7.2. Age

```
R> age_grp <- cut(pbc_rf$xvar$age, breaks=seq(0,100,10))
R> ggvar$age_grp <- paste("age=",age_grp, sep="")
R>
R> var_dep <- plot(ggvar, x_var = "bili", smooth = TRUE,
+                  method = "loess", span=1.5,alpha = .5, se = FALSE) +
+   labs(y = "Survival", x = dta.labs["bili", "label"]) +
+   theme(legend.position = "none") +
+   scale_color_manual(values = strCol, labels = event.labels) +
+   scale_shape_manual(values = event.marks, labels = event.labels)+
+   facet_wrap(~age_grp)
R>
R> var_dep


R> # Get the training data to work with...
R> dta.train  <- pbc_rf$xvar
R> dta.train$age_grp <- age_grp
```

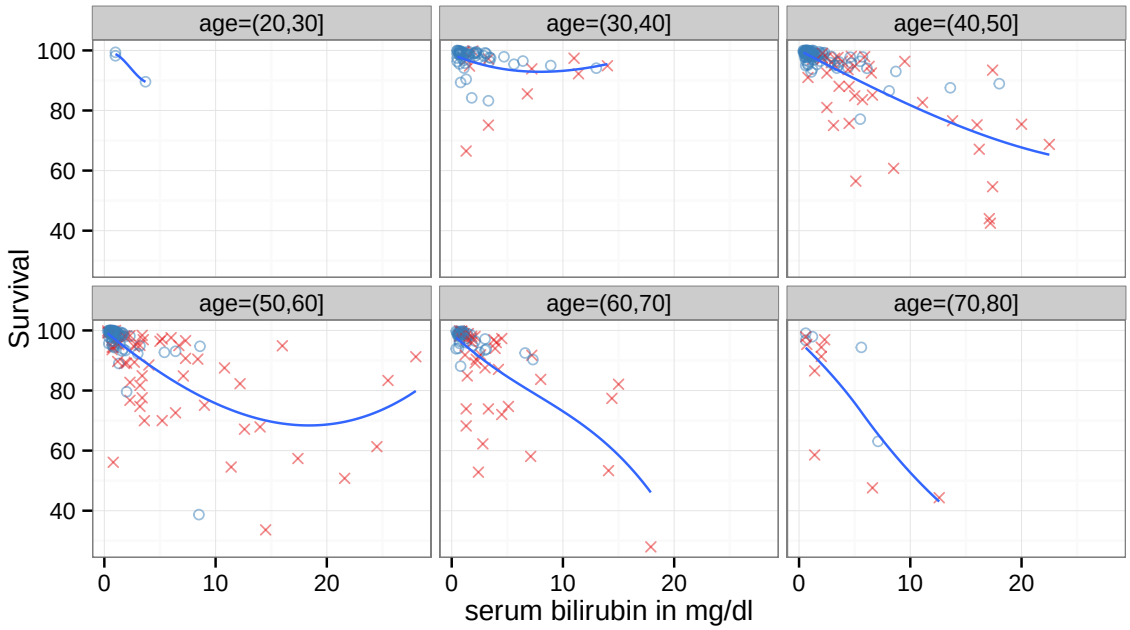

Figure 16: Bilirubin-age interaction coplot at 1 year.

```
R>
R> # Create a series of coplot subsets....
R> lng <- length(levels(age_grp))
R> sbst <- mclapply(1:lng, function(ind){
+   st <- which(dta.train$age_grp==levels(age_grp)[ind])
+   if(length(st) == 0) NULL
+   else st
+   })
R>
R> lvl <- levels(age_grp)
R> # Collapse the subset list to interesting items
R> # (those with observations)
R> # If you work backwards, you do extra tests, but it
R> # cuts the correct items. Cute.
R> for(ind in lng:1){
+   if(is.null(sbst[[ind]])){
+     sbst[[ind]] <- NULL
+
+     # reset the levels, so we can label things later
+     lvl <- lvl[-ind]
+     }
+ }
R>
R> pDat.partlist <- mclapply(1:length(sbst), function(ind){
+   plot.variable(pbc_rf, surv.type="surv", time=1,
```

```
+                              subset = sbst[[ind]],
+                            xvar.names="bili", partial=TRUE,
+                            show.plots = FALSE)
+   })
R>
R> gg_part <- mclapply(pDat.partlist, gg_partial)
R>
R> # Flip y-axis
R> cls <- class(gg_part)
R> class(gg_part) <-  c("gg_partial_list", cls)
R>
R> for(ind in 1:length(gg_part)){
+ gg_part[[ind]]$age <- lvl[ind]
+ }
R> gg_merge <- do.call(rbind, gg_part)
R> gg_merge$age <- paste("Age=", gg_merge$age)
R> gg_merge$age <- factor(gg_merge$age)
R>
R> ggpl <- ggplot(gg_merge, aes(x=bili, y=yhat, shape=age, color=age))+
+   geom_point()+geom_smooth(se=FALSE)+
+   labs(x="Surgical Date", y="Survival 1 year")+
+   scale_color_brewer(palette="Set1")
R> ggpl
```

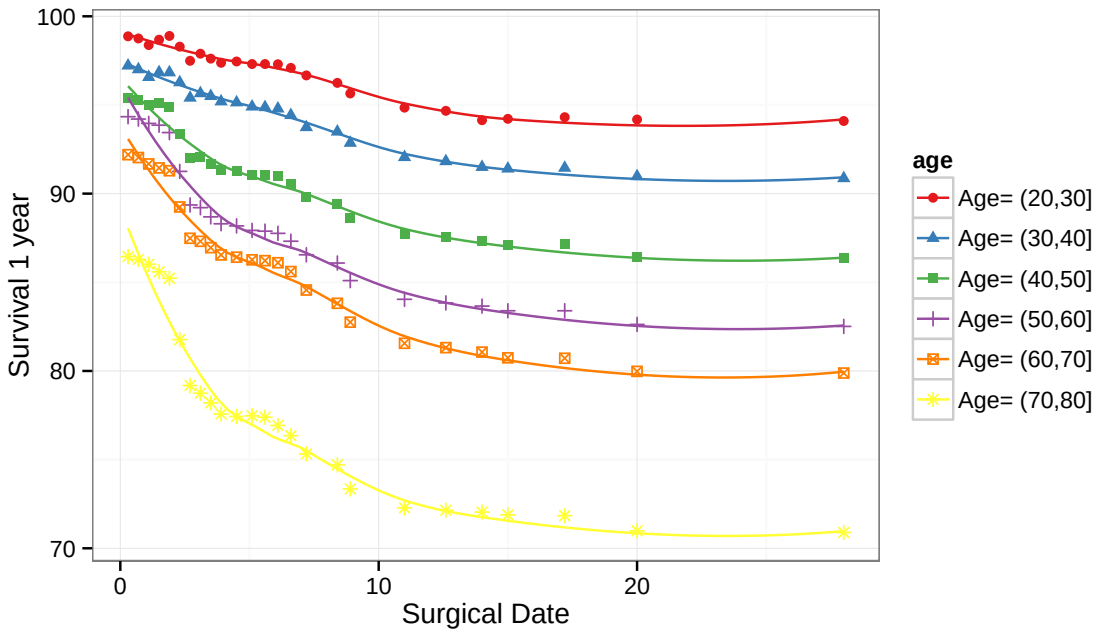

Figure 17: Bilirubin-age interaction partial coplot at 1 year.

7.3. Albumin

```
R> albumin_grp <- cut(pbc_rf$xvar$albumin, breaks=c(0,seq(3,3.5,.5),5))
R> ggvar$albumin_grp <- paste("albumin=",albumin_grp, sep="")
R>
R> var_dep <- plot(ggvar, x_var = "bili", smooth = TRUE,
+                  method = "loess", span=1.5,alpha = .5, se = FALSE) +
+   labs(y = "Survival", x = dta.labs["bili", "label"]) +
+   theme(legend.position = "none") +
+   scale_color_manual(values = strCol, labels = event.labels) +
+   scale_shape_manual(values = event.marks, labels = event.labels)+
+   facet_wrap(~albumin_grp)
R>
R> var_dep
```

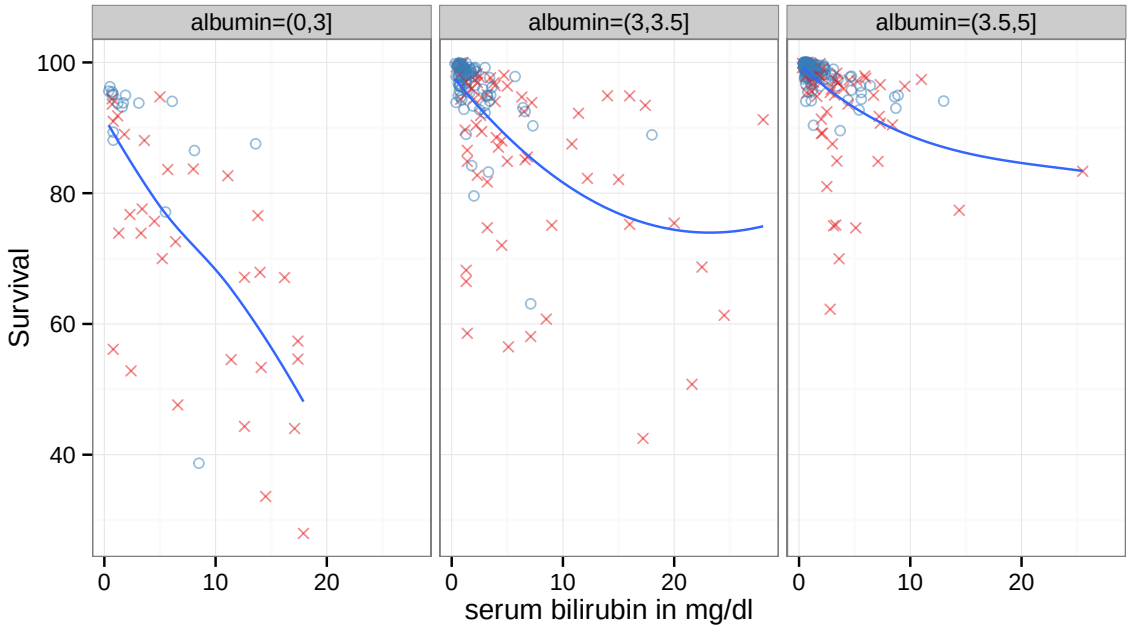

Figure 18: Bilirubin-age interaction coplot at 1 year.

```
R> # Get the training data to work with...
R> dta.train  <- pbc_rf$xvar
R> dta.train$albumin_grp <- albumin_grp
R>
R> # Create a series of coplot subsets....
R> lng <- length(levels(albumin_grp))
R> sbst <- mclapply(1:lng, function(ind){
+   st <- which(dta.train$albumin_grp==levels(albumin_grp)[ind])
+   if(length(st) == 0) NULL
```

```
+   else st
+   })
R>
R> lvl <- levels(albumin_grp)
R> # Collapse the subset list to interesting items
R> # (those with observations)
R> # If you work backwards, you do extra tests, but it
R> # cuts the correct items. Cute.
R> for(ind in lng:1){
+   if(is.null(sbst[[ind]])){
+     sbst[[ind]] <- NULL
+
+     # reset the levels, so we can label things later
+     lvl <- lvl[-ind]
+     }
+ }
R>
R> pDat.partlist <- mclapply(1:length(sbst), function(ind){
+   plot.variable(pbc_rf, surv.type="surv", time=1,
+                              subset = sbst[[ind]],
+                          xvar.names="bili", partial=TRUE,
+                          show.plots = FALSE)
+   })
R>
R> gg_part <- mclapply(pDat.partlist, gg_partial)
R>
R> # Flip y-axis
R> cls <- class(gg_part)
R> class(gg_part) <-  c("gg_partial_list", cls)
R>
R> for(ind in 1:length(gg_part)){
+ gg_part[[ind]]$albumin <- lvl[ind]
+ }
R> gg_merge <- do.call(rbind, gg_part)
R> gg_merge$albumin <- paste("albumin=", gg_merge$albumin)
R> gg_merge$albumin <- factor(gg_merge$albumin)
R>
R> ggpl <- ggplot(gg_merge, aes(x=bili, y=yhat, shape=albumin, color=albumin))+
+   geom_point()+geom_smooth(se=FALSE)+
+   labs(x="Surgical Date", y="Survival 1 year")+
+   scale_color_brewer(palette="Set1")
R> ggpl
```

7.4. prothrombin

```
R> ggvar$prothrombin_grp <- cut(pbc_rf$xvar$prothrombin, breaks=c(8.9,10,11,12,18))
```

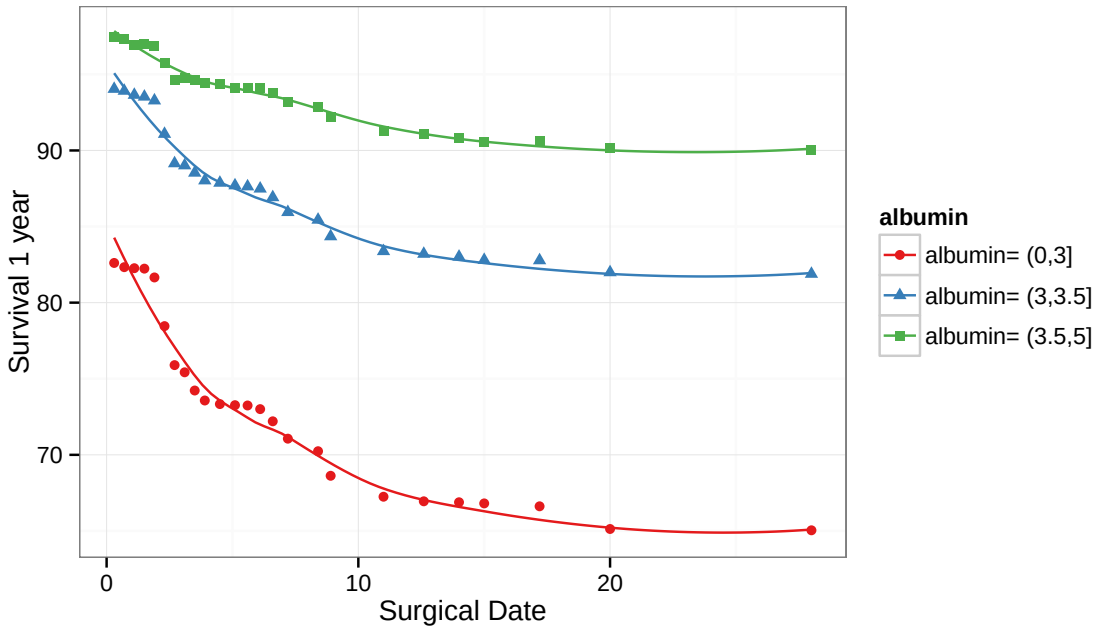

Figure 19: Bilirubin-albumin interaction partial coplot at 1 year.

```
R>
R> var_dep <- plot(ggvar[which(!is.na(ggvar$prothrombin_grp)),],
+                  x_var = "bili", smooth = TRUE,
+                  method = "loess", span=1.5,alpha = .5, se = FALSE) +
+   labs(y = "Survival", x = dta.labs["bili", "label"]) +
+   theme(legend.position = "none") +
+   scale_color_manual(values = strCol, labels = event.labels) +
+   scale_shape_manual(values = event.marks, labels = event.labels)+
+   facet_wrap(~prothrombin_grp)
R>
R> var_dep


R> # Get the training data to work with...
R> dta.train  <- pbc_rf$xvar
R> dta.train$prothrombin_grp <- prothrombin_grp
R>
R> # Create a series of coplot subsets....
R> lng <- length(levels(prothrombin_grp))
R> sbst <- mclapply(1:lng, function(ind){
+   st <- which(dta.train$prothrombin_grp==levels(prothrombin_grp)[ind])
+   if(length(st) == 0) NULL
+   else st
+   })
R>
```

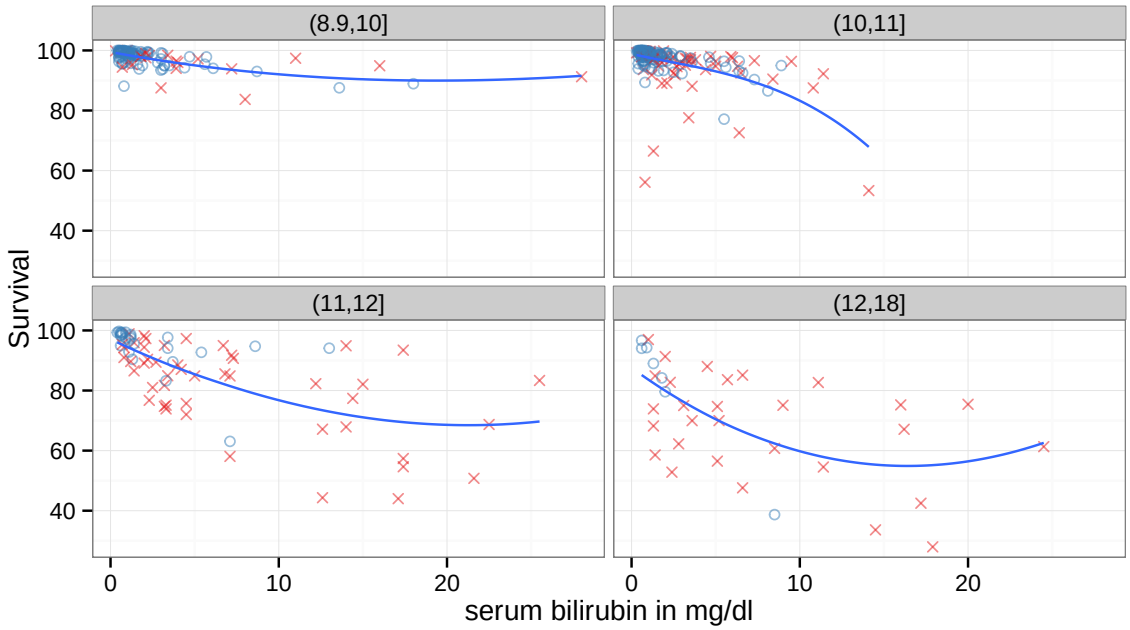

Figure 20: Bilirubin-prothrombin interaction coplot at 1 year.

```
R> lvl <- levels(prothrombin_grp)
R> # Collapse the subset list to interesting items
R> # (those with observations)
R> # If you work backwards, you do extra tests, but it
R> # cuts the correct items. Cute.
R> for(ind in lng:1){
+   if(is.null(sbst[[ind]])){
+     sbst[[ind]] <- NULL
+
+     # reset the levels, so we can label things later
+     lvl <- lvl[-ind]
+     }
+ }
R>
R> pDat.partlist <- mclapply(1:length(sbst), function(ind){
+   plot.variable(pbc_rf, surv.type="surv", time=1,
+                              subset = sbst[[ind]],
+                            xvar.names="bili", partial=TRUE,
+                            show.plots = FALSE)
+   })
R>
R> gg_part <- mclapply(pDat.partlist, gg_partial)
R>
R> # Flip y-axis
R> cls <- class(gg_part)
```

```
R> class(gg_part) <-  c("gg_partial_list", cls)
R>
R> for(ind in 1:length(gg_part)){
+ gg_part[[ind]]$prothrombin <- lvl[ind]
+ }
R> gg_merge <- do.call(rbind, gg_part)
R> gg_merge$prothrombin <- paste("prothrombin=", gg_merge$prothrombin)
R> gg_merge$prothrombin <- factor(gg_merge$prothrombin)
R>
R> ggpl <- ggplot(gg_merge, aes(x=bili, y=yhat, shape=prothrombin, color=prothrombin))+
+   geom_point()+geom_smooth(se=FALSE)+
+   labs(x="Surgical Date", y="Survival 1 year")+
+   scale_color_brewer(palette="Set1")
R> ggpl
```

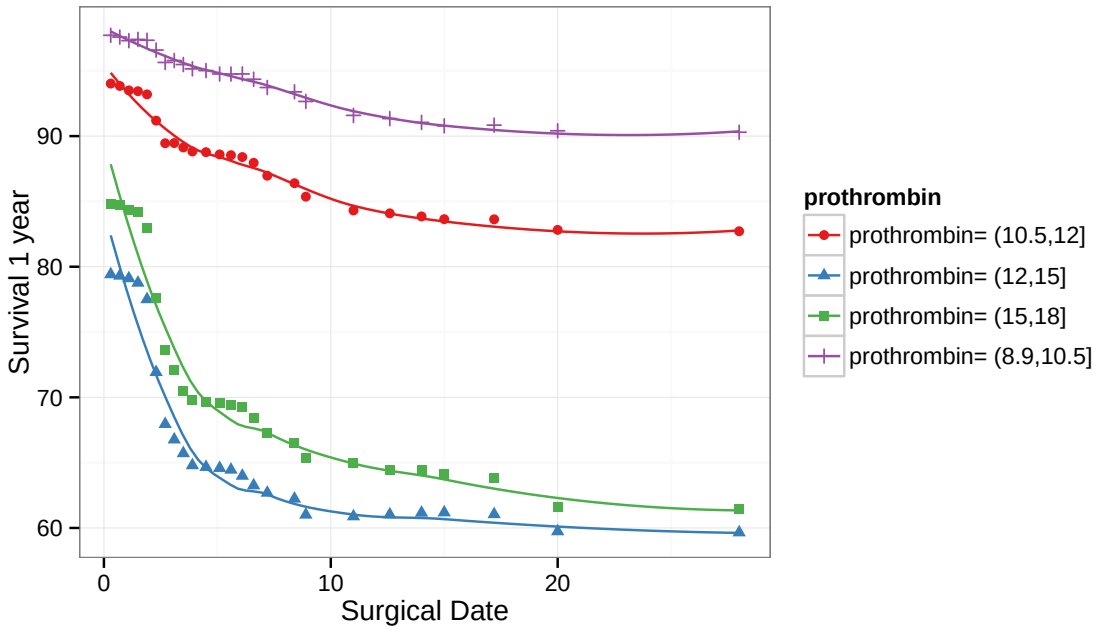

Figure 21: Bilirubin-prothrombin interaction partial coplot at 1 year.

8. Conclusion

References

Breiman L (1996a). "Bagging predictors." *Machine Learning*, **26**, 123–140.

Breiman L (1996b). "Out–Of–Bag Estimation." *Technical report*, Statistics Department, University of California,Berkeley, CA. 94708. URL ftp://ftp.stat.berkeley.edu/pub/users/breiman/OOBestimation.ps.Z.

Breiman L (2001). "Random Forests." *Machine Learning*, **45**(1), 5–32.

Breiman L, Friedman JH, Olshen R, Stone C (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Chambers JM (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole.

Cleveland WS (1981). "LOWESS: A program for smoothing scatterplots by robust locally weighted regression." *The American Statistician*, **35**(1), 54.

Cleveland WS (1993). *Visualizing Data*. Summit Press.

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

Efron B, Tibshirani R (1994). *An Introduction to the Bootstrap*. Chapman & Hall/CRC. ISBN 0412042312.

Fleming TR, Harrington DP (1991). *Counting processes and survival analysis*. Wiley, New York.

Friedman JH (2000a). "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics*, **29**, 1189–1232.

Friedman JH (2000b). "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics*, **29**, 1189–1232.

H I, UB K, X C, AJ M (2011). "Random survival forests for high-dimensional data." *Statist. Anal. Data Mining*, **4**, 115–132.

Hastie T, Tibshirani R, Friedman JH (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2 edition. Springer. ISBN 978-0-387-84857-0.

Ishwaran H (2007). "Variable importance in binary regression trees and forests." *Electronic Journal of Statistics*, **1**, 519–537.

Ishwaran H, Kogalur UB (2007). "Random survival forests for R." *R News*, **7**, 25–31.

Ishwaran H, Kogalur UB (2010). "Consistency of random survival forests." *Statistics and Probability Letters*, **80**, 1056–1064.

Ishwaran H, Kogalur UB (2014). "Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.6."

Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS (2008). "Random survival forests." *The Annals of Applied Statistics*, **2**(3), 841–860.

Ishwaran H, Kogalur UB, Gorodeski EZ, Minn AJ, Lauer MS (2010a). "High-dimensional variable selection for survival data." *J. Amer. Statist. Assoc.*, **105**, 205–217.

Ishwaran H, Kogalur UB, Gorodeski EZ, Minn AJ, Lauer MS (2010b). "High-Dimensional Variable Selection for Survival Data." *Journal of the American Statistical Association*, **105**(489).

Liaw A, Wiener M (2002). "Classification and Regression by randomForest." *R News*, **2**(3), 18–22.

Wickham H (2009). *ggplot2: elegant graphics for data analysis*. Springer New York. ISBN 978-0-387-98140-6.

Affiliation:

John Ehrlinger
Quantitative Health Sciences
Lerner Research Institute
Cleveland Clinic
9500 Euclid Ave
Cleveland, Ohio 44195
E-mail: john.ehrlinger@gmail.com
URL: http://www.lerner.ccf.org/qhs/people/ehrlinj/