Random Forests

Dr. Jeffrey Strickland

9/18/2018

Random Forest using R

Random Forest algorithm is built in randomForest package of R and same name function allows us to use the Random Forest in R.

Load libraries

```
if(!require(randomForest)) install.packages("randomForest")
if(!require(colorspace)) install.packages("colorspace")
if(!require(reshape)) install.packages("reshape")
if(!require(ggplot2)) install.packages("ggplot2")

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

library(colorspace)
library(reshape)
library(ggplot2)

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin
```

Some of the commonly used parameters of randomForest functions are - x : Random Forest Formula - data: Input data frame - ntree: Number of decision trees to be grown - replace: Takes True and False and indicates whether to take sample with/without replacement - sampsize: Sample size to be drawn from the input data for growing decision tree - importance: Whether independent variable importance in random forest be assessed - proximity: Whether to calculate proximity measures between rows of a data frame

Medical longevity Study of primary biliary cirrhosis (PBC)

Data was obtained from a Mayo Clinic randomized 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 (DPCA). The data and partial likelihood model is described in Fleming and Harrington (1991)

```
#if(!require(ggRandomForest)) install.packages("ggRandomForest")
#if(!require(randomForestSRC)) install.packages("randomForestSRC")
library(ggRandomForests)

## Loading required package: randomForestSRC

##
##  randomForestSRC 2.6.1
##
##  Type rfsrc.news() to see new features, changes, and bug fixes.
##


##
## Attaching package: 'ggRandomForests'

## The following object is masked from 'package:randomForestSRC':
##
##     partial.rfsrc

library(randomForestSRC)
data(pbc)
summary(pbc)

##       days           status         treatment          age
##  Min.   :  41   Min.   :0.0000   Min.   :1.000   Min.   : 9598
##  1st Qu.:1093   1st Qu.:0.0000   1st Qu.:1.000   1st Qu.:15644
##  Median :1730   Median :0.0000   Median :1.000   Median :18628
##  Mean   :1918   Mean   :0.3852   Mean   :1.494   Mean   :18533
##  3rd Qu.:2614   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:21273
##  Max.   :4795   Max.   :1.0000   Max.   :2.000   Max.   :28650
##                                  NA's   :106
##       sex            ascites           hepatom          spiders
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:1.0000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.0000   Median :0.00000   Median :1.0000   Median :0.0000
##  Mean   :0.8947   Mean   :0.07692   Mean   :0.5128   Mean   :0.2885
##  3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
##                   NA's   :106       NA's   :106      NA's   :106
##      edema             bili             chol           albumin
##  Min.   :0.0000   Min.   : 0.300   Min.   : 120.0   Min.   :1.960
##  1st Qu.:0.0000   1st Qu.: 0.800   1st Qu.: 249.5   1st Qu.:3.243
##  Median :0.0000   Median : 1.400   Median : 309.5   Median :3.530
##  Mean   :0.1005   Mean   : 3.221   Mean   : 369.5   Mean   :3.497
##  3rd Qu.:0.0000   3rd Qu.: 3.400   3rd Qu.: 400.0   3rd Qu.:3.770
##  Max.   :1.0000   Max.   :28.000   Max.   :1775.0   Max.   :4.640
##                                    NA's   :134
##      copper            alk               sgot             trig
```

```
##  Min.   :  4.00   Min.   :  289.0   Min.   : 26.35   Min.   : 33.00
##  1st Qu.: 41.25   1st Qu.:  871.5   1st Qu.: 80.60   1st Qu.: 84.25
##  Median : 73.00   Median : 1259.0   Median :114.70   Median :108.00
##  Mean   : 97.65   Mean   : 1982.7   Mean   :122.56   Mean   :124.70
##  3rd Qu.:123.00   3rd Qu.: 1980.0   3rd Qu.:151.90   3rd Qu.:151.00
##  Max.   :588.00   Max.   :13862.4   Max.   :457.25   Max.   :598.00
##  NA's   :108      NA's   :106       NA's   :106      NA's   :136
##     platelet      prothrombin        stage
##  Min.   : 62.0   Min.   : 9.00   Min.   :1.000
##  1st Qu.:188.5   1st Qu.:10.00   1st Qu.:2.000
##  Median :251.0   Median :10.60   Median :3.000
##  Mean   :257.0   Mean   :10.73   Mean   :3.024
##  3rd Qu.:318.0   3rd Qu.:11.10   3rd Qu.:4.000
##  Max.   :721.0   Max.   :18.00   Max.   :4.000
##  NA's   :11      NA's   :2       NA's   :6
```

Transform variable values: years to days; 0-1 to T-F

```
pbc1<-pbc
pbc1$Years<-pbc$days/365
pbc1$age<-pbc$age/365
pbc1$Status <- NULL
pbc1$status
```

```
##   [1] 1 0 1 1 0 1 0 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1
##  [36] 0 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 0 1 0
##  [71] 0 0 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1 1 1 1 0 1 1 0 1 0 0 1 0 0 1 1 0
## [106] 1 0 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0 0
## [141] 0 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 0 0 1 0 0 1 1 1 1 0 1 0 1 0 0 0 0 0 0
## [176] 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
## [211] 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0
## [246] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
## [281] 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0
## [316] 1 0 0 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 1 0 0 1 1 0 0 1
## [351] 1 0 0 1 0 1 0 0 0 1 0 0 0 1 0 1 0 1 1 1 1 0 0 0 0 1 0 1 1 0 0 1 0 0 0
## [386] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0
```

Get transformed data

```
pbc2<-pbc1[,2:20]
head(pbc2)
```

```
##   status treatment      age sex ascites hepatom spiders edema bili chol
## 1      1         1 58.80548   1       1       1       1   1.0 14.5  261
## 2      0         1 56.48493   1       0       1       1   0.0  1.1  302
## 3      1         1 70.12055   0       0       0       0   0.5  1.4  176
## 4      1         1 54.77808   1       0       1       1   0.5  1.8  244
## 5      0         2 38.13151   1       0       1       1   0.0  3.4  279
## 6      1         2 66.30411   1       0       1       0   0.0  0.8  248
##   albumin copper    alk   sgot trig platelet prothrombin stage     Years
## 1    2.60    156 1718.0 137.95  172      190        12.2     4  1.095890
## 2    4.14     54 7394.8 113.52   88      221        10.6     3 12.328767
## 3    3.48    210  516.0  96.10   55      151        12.0     4  2.772603
```

```
## 4    2.54     64 6121.8  60.63   92      183        10.3     4  5.273973
## 5    3.53    143  671.0 113.15   72      136        10.9     3  4.120548
## 6    3.98     50  944.0  93.00   63       NA        11.0     3  6.857534
```

Reshape continuous variable data for exploratory analysis

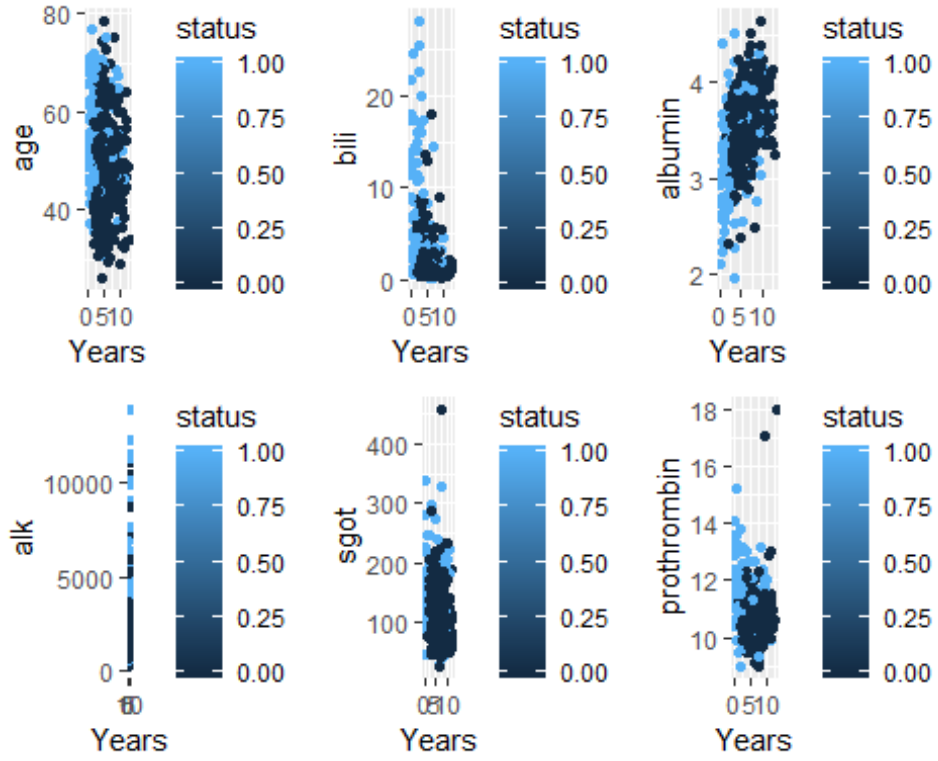
```
dtb1<- melt(pbc2, id.vars=c("age","Years","status"))
dtb2<- melt(pbc2, id.vars=c("bili","Years","status"))
dtb3<- melt(pbc2, id.vars=c("albumin","Years","status"))
dtb4<- melt(pbc2, id.vars=c("alk","Years","status"))
dtb5<- melt(pbc2, id.vars=c("sgot","Years","status"))
dtb6<- melt(pbc2, id.vars=c("prothrombin","Years","status"))
dtb7<- melt(pbc2, id.vars=c("chol","Years","status"))
dtb8<- melt(pbc2, id.vars=c("copper","Years","status"))
dtb9<- melt(pbc2, id.vars=c("trig","Years","status"))
dtb10<- melt(pbc2, id.vars=c("platelet","Years","status"))
```

Plot contuinuous variables

```
gg1<-ggplot(data=dtb1, aes(x=Years, y=age)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
gg2<-ggplot(data=dtb2, aes(x=Years, y=bili)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
gg3<-ggplot(data=dtb3, aes(x=Years, y=albumin)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
gg4<-ggplot(data=dtb4, aes(x=Years, y=alk)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
gg5<-ggplot(data=dtb5, aes(x=Years, y=sgot)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
gg6<-ggplot(data=dtb6, aes(x=Years, y=prothrombin)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
gg7<-ggplot(data=dtb7, aes(x=Years, y=chol)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
gg8<-ggplot(data=dtb8, aes(x=Years, y=copper)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
gg9<-ggplot(data=dtb9, aes(x=Years, y=trig)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
gg10<-ggplot(data=dtb10, aes(x=Years, y=platelet)) +
      geom_point(aes(x=Years, color=status)) +
      scale_fill_brewer(type="seq", palette = "Set1")
```

Show multiple pots in a window

```
if(require(!gridExtra)) install.packages("gridExtra")

## Warning in if (!loaded) {: the condition has length > 1 and only the first
## element will be used

## c("Loading required package: !", "Loading required package: gridExtra")

library(gridExtra)

##
## Attaching package: 'gridExtra'
## The following object is masked from 'package:randomForest':
##
##     combine

grid.arrange(gg1,gg2,gg3,gg4,gg5,gg6,nrow=2)
```



Include only the randomized patients.

```
pbc.trial <- pbc2[-which(is.na(pbc2$treatment)),]
```

Create a test set from the remaining patients

```
pbc.test <- pbc2[which(is.na(pbc2$treatment)),]
head(pbc.trial)
```

```
##   status treatment      age sex ascites hepatom spiders edema bili chol
## 1      1         1 58.80548   1       1       1       1   1.0 14.5  261
## 2      0         1 56.48493   1       0       1       1   0.0  1.1  302
## 3      1         1 70.12055   0       0       0       0   0.5  1.4  176
## 4      1         1 54.77808   1       0       1       1   0.5  1.8  244
## 5      0         2 38.13151   1       0       1       1   0.0  3.4  279
## 6      1         2 66.30411   1       0       1       0   0.0  0.8  248
##   albumin copper    alk   sgot trig platelet prothrombin stage     Years
## 1    2.60    156 1718.0 137.95  172      190        12.2     4  1.095890
## 2    4.14     54 7394.8 113.52   88      221        10.6     3 12.328767
## 3    3.48    210  516.0  96.10   55      151        12.0     4  2.772603
## 4    2.54     64 6121.8  60.63   92      183        10.3     4  5.273973
## 5    3.53    143  671.0 113.15   72      136        10.9     3  4.120548
## 6    3.98     50  944.0  93.00   63       NA        11.0     3  6.857534
```

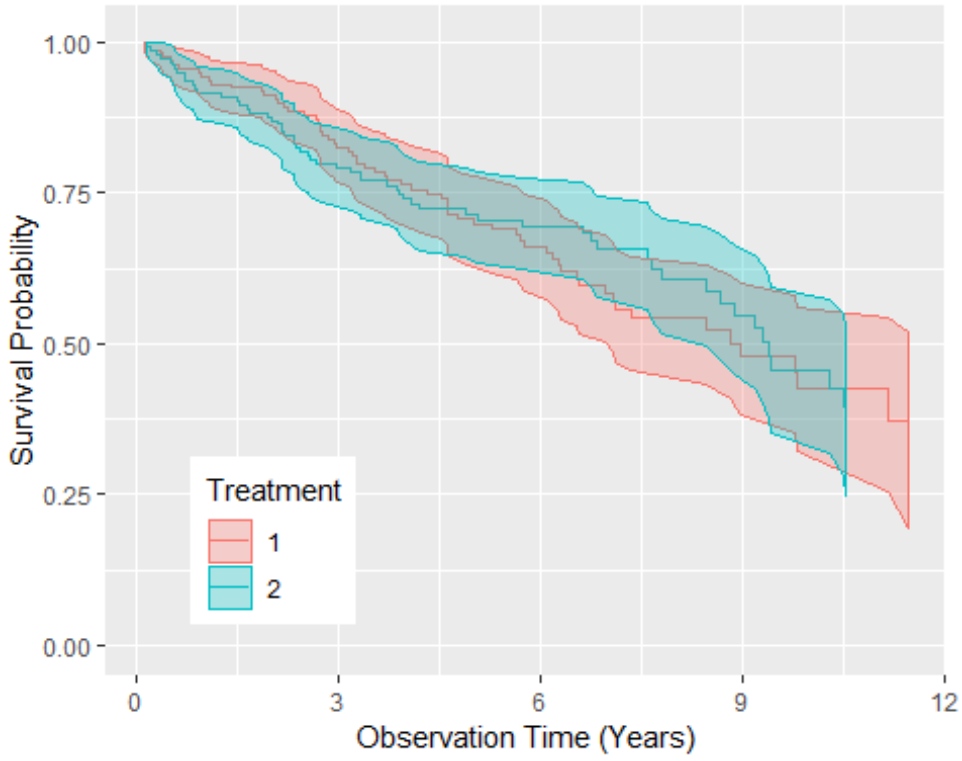
Create a test set from the remaining patients

Create the gg_survival object

```
gg_dta <- gg_survival(interval = "Years",
                      censor = "status",
                      by = "treatment",
                      data = pbc.trial,
                      conf.int = .95)
```

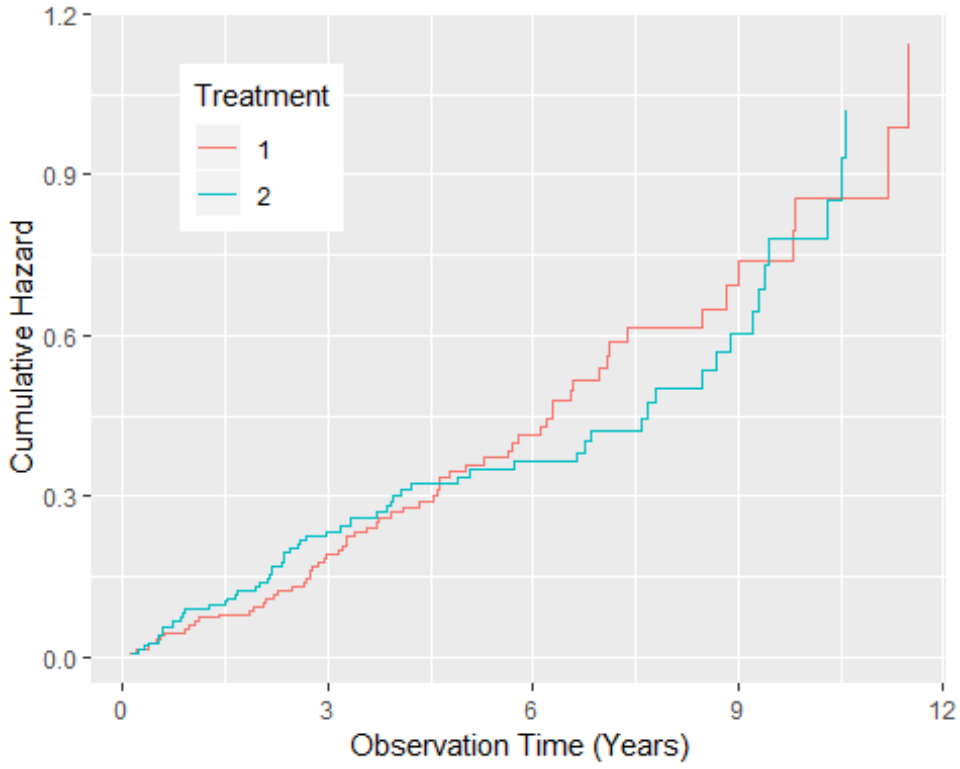
Pot the survival probability function

```
plot(gg_dta) +
  labs(y = "Survival Probability",
       x = "Observation Time (Years)",
       color = "Treatment", fill = "Treatment") +
  theme(legend.position = c(.2,.2)) +
  coord_cartesian(y = c(0,1.01))
```



Plot the cumulative hazard function

```
plot(gg_dta, type="cum_haz") +
  labs(y = "Cumulative Hazard",
       x = "Observation Time (Years)",
       color = "Treatment", fill = "Treatment") +
  theme(legend.position = c(.2,.8))
```



Duplicate the trial data

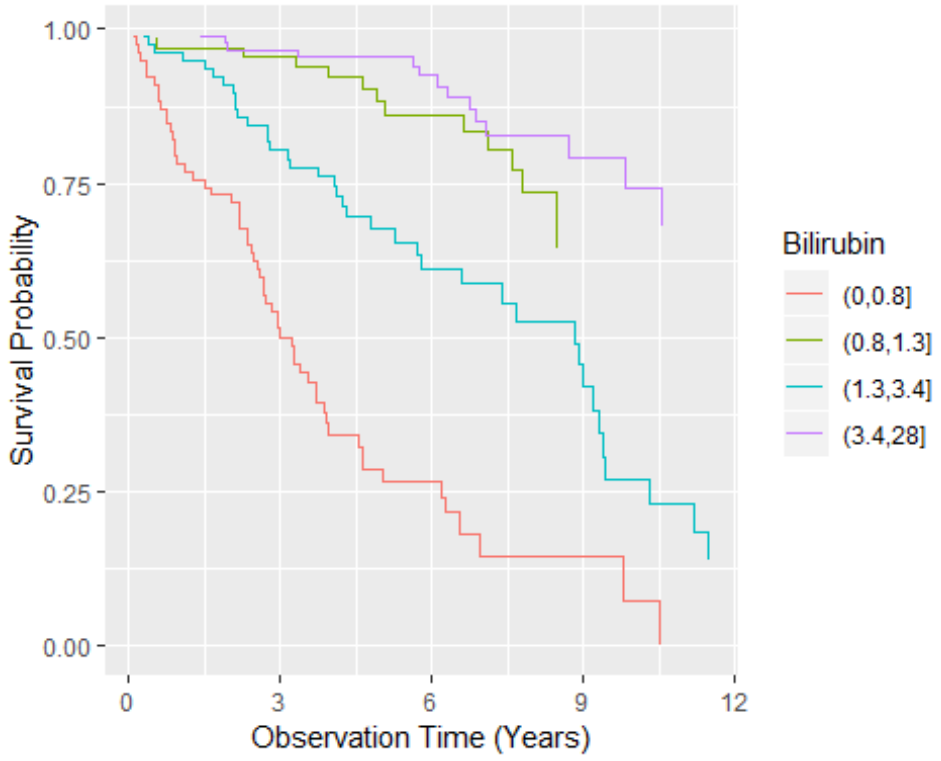
```
pbc.bili <- pbc.trial
```

Group by bilirubin values

```
pbc.bili$bili_grp <- cut(pbc.trial$bili,
                         breaks = c(0, .8, 1.3, 3.4,
                                    max(pbc.trial$bili)))
```

Plot the gg_survival object directly

```
plot(gg_survival(interval = "Years",censor = "status",
                 by = "bili_grp", data = pbc.bili),
     error = "none") +
  labs(y = "Survival Probability",
       x = "Observation Time (Years)",
       color = "Bilirubin")
```



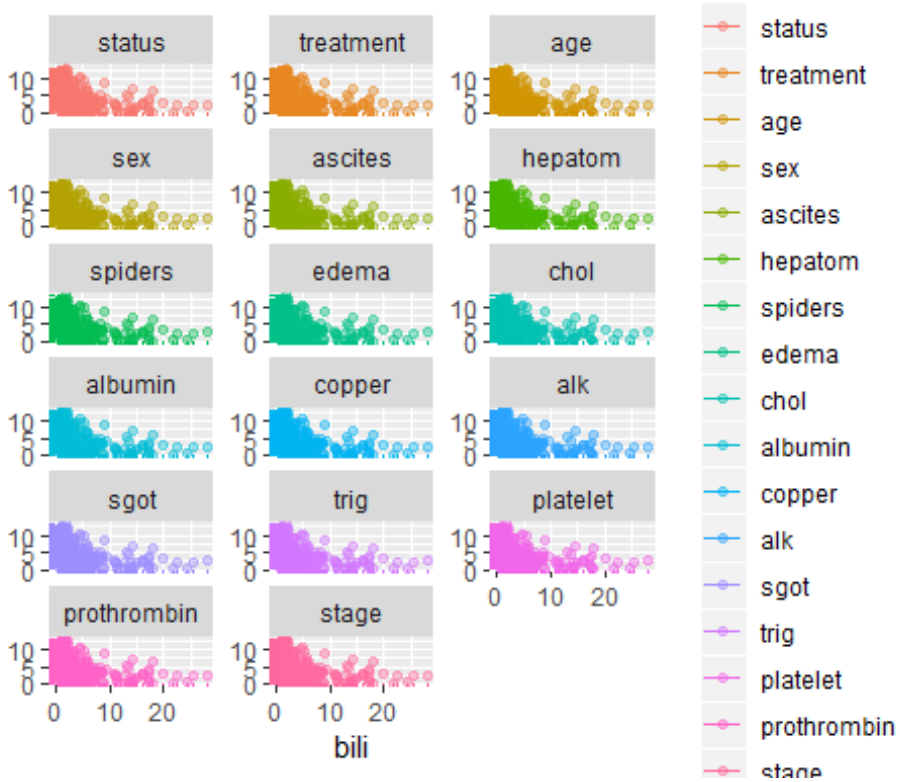

```
#if(!require(shape2)) install.packages(shape2)
library(reshape2)

##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:reshape':
##
##     colsplit, melt, recast

dta <- melt(pbc2, id.vars=c("bili","Years"))
dtb <- melt(pbc2, id.vars=c("Years","status"))
head(dtb)

##       Years status  variable value
## 1  1.095890      1 treatment     1
## 2 12.328767      0 treatment     1
## 3  2.772603      1 treatment     1
## 4  5.273973      1 treatment     1
## 5  4.120548      0 treatment     2
## 6  6.857534      1 treatment     2
```

Using shiny GUI for colorspace

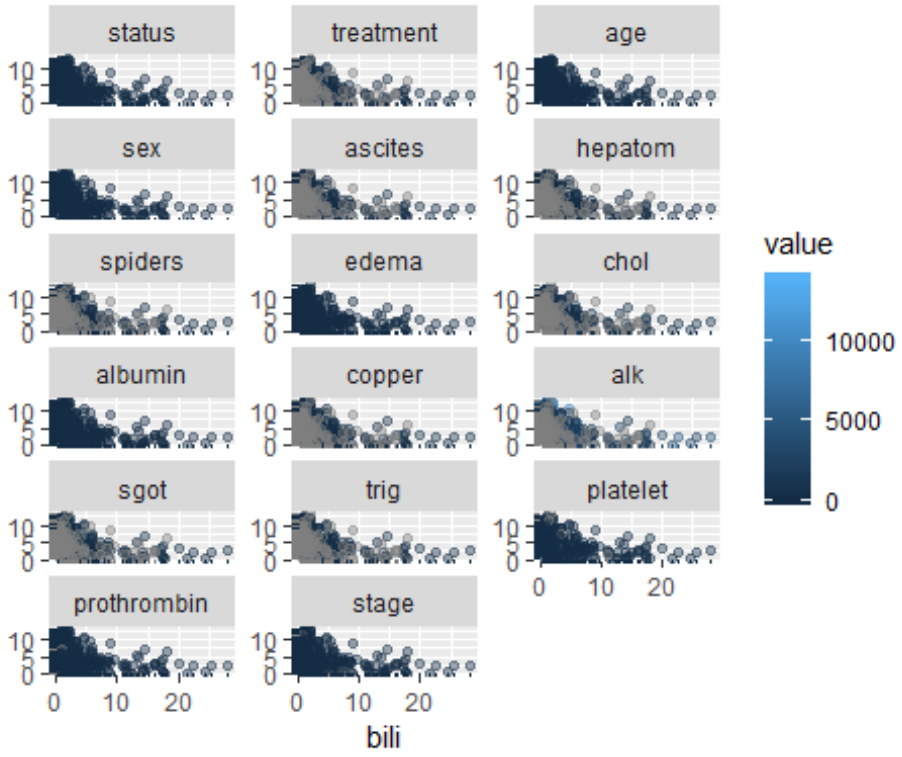
```
choose_palette("tcltk")
```

Analog to: choose_palette(gui = “shiny”)

```
ggplot(dta,  aes(x=bili, y=Years, color=variable)) + geom_point(alpha=.4) +
geom_rug(data=dta) +
    labs(y="", x="bili") + scale_fill_gradientn(colours =
colorspace::rainbow_hcl(17)) +
  facet_wrap(~variable, scales="free_y", ncol=3)
```




```
ggplot(dta,  aes(x=bili, y=Years, color=value)) + geom_point(alpha=.4) +
geom_rug(data=dta) +
  labs(y="", x="bili") + scale_fill_brewer (type = "seq", palette = "Set2") +
  facet_wrap(~variable, scales="free_y", ncol=3)
```



Grow and store the random survival forest

```
rfsrc_pbc <- rfsrc(Surv(Years, status) ~ .,
                   data = pbc.trial)
```

Use random splitting (nsplit = 10) and impute missing values (na.action = "na.impute")

```
rfsrc_pbc2 <- rfsrc(Surv(Years, status) ~ .,
                   data = pbc.trial,
                   nsplit = 10,
                   na.action = "na.impute")
```

Print the forest summary

```
rfsrc_pbc
```

```
##                          Sample size: 276
##                     Number of deaths: 111
##                      Number of trees: 1000
##            Forest terminal node size: 3
##        Average no. of terminal nodes: 66.925
## No. of variables tried at each split: 5
##               Total no. of variables: 17
##                             Analysis: RSF
##                               Family: surv
##                     Splitting rule: logrank
##                         Error rate: 17.46%
```

```
rfsrc_pbc2
```

```
##                          Sample size: 312
##                     Number of deaths: 125
##                     Was data imputed: yes
##                      Number of trees: 1000
##            Forest terminal node size: 3
##        Average no. of terminal nodes: 74.26
## 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: 16.49%
```

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). We used nsplit = 10 random split points to select random split rule, instead of an optimization on each variable at each split for performance reasons.

Predict survival for 106 patients not in randomized trial

```
pbc.test$status<-ifelse(pbc.test$status == "T",1,0)
head(pbc.test)
```

```
##     status treatment      age sex ascites hepatom spiders edema bili chol
## 313      0        NA 60.04110   1      NA      NA      NA   0.0  0.7   NA
## 314      0        NA 65.04384   1      NA      NA      NA   0.5  1.4   NA
## 315      0        NA 54.03836   1      NA      NA      NA   0.0  0.7   NA
## 316      0        NA 75.05205   1      NA      NA      NA   0.5  0.7   NA
## 317      0        NA 62.04384   1      NA      NA      NA   0.0  0.8   NA
## 318      0        NA 43.03014   1      NA      NA      NA   0.0  0.7   NA
##     albumin copper alk sgot trig platelet prothrombin stage     Years
## 313    3.65     NA  NA   NA   NA      378        11.0    NA 11.128767
## 314    3.04     NA  NA   NA   NA      331        12.1     4  9.756164
## 315    4.03     NA  NA   NA   NA      226         9.8     4  7.791781
## 316    3.96     NA  NA   NA   NA       NA        11.3     4  5.673973
## 317    2.48     NA  NA   NA   NA      273        10.0    NA  8.301370
## 318    3.68     NA  NA   NA   NA      306         9.5     2  4.602740
```

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

Print prediction summary

```
rfsrc_pbc_test
```

```
##   Sample size of test (predict) data: 106
##                Was test data imputed: yes
##                 Number of grow trees: 1000
##   Average no. of grow terminal nodes: 66.925
##           Total no. of grow variables: 17
```

```
##                                Analysis: RSF
##                                  Family: surv
```

Print prediction summary

```
rfsrc_pbc_test2 <- predict(rfsrc_pbc2,
                           newdata = pbc.test,
                           na.action = "na.impute")
```

Print prediction summary

```
rfsrc_pbc_test2
```

```
##   Sample size of test (predict) data: 106
##                 Was test data imputed: yes
##                  Number of grow trees: 1000
##   Average no. of grow terminal nodes: 74.26
##           Total no. of grow variables: 17
##                                Analysis: RSF
##                                  Family: surv
```

Extract VIMP measures for each of the variables used to grow the forest.

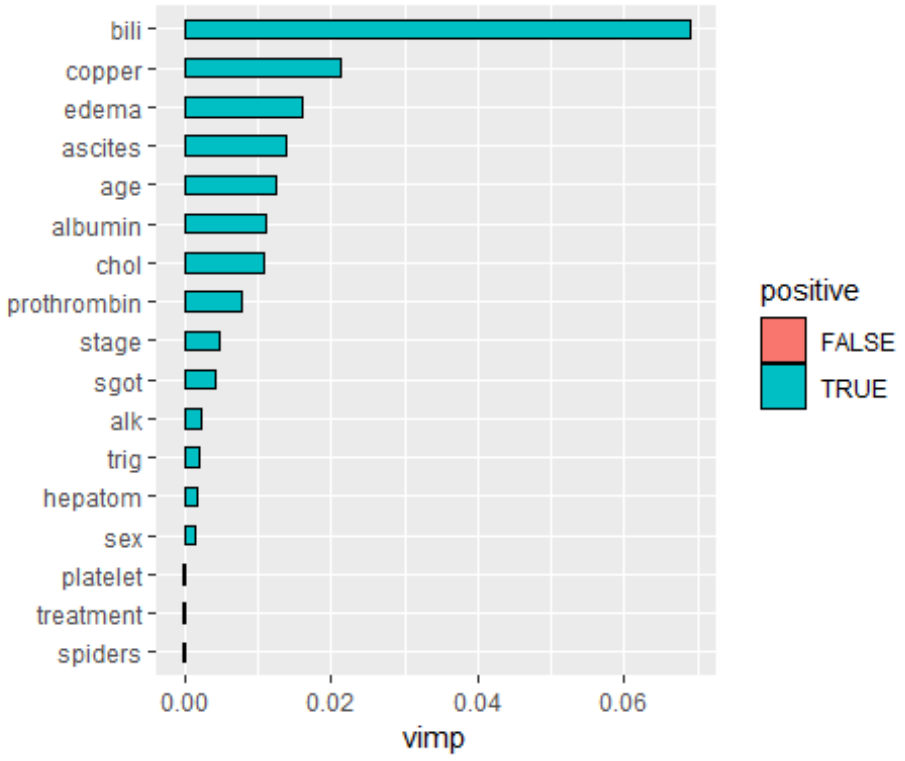
```
gg_variable(rfsrc_pbc)
gg_dta<-gg_vimp(rfsrc_pbc)
```

```
## Warning in gg_vimp.rfsrc(rfsrc_pbc): rfsrc object does not contain VIMP
## information. Calculating...
```

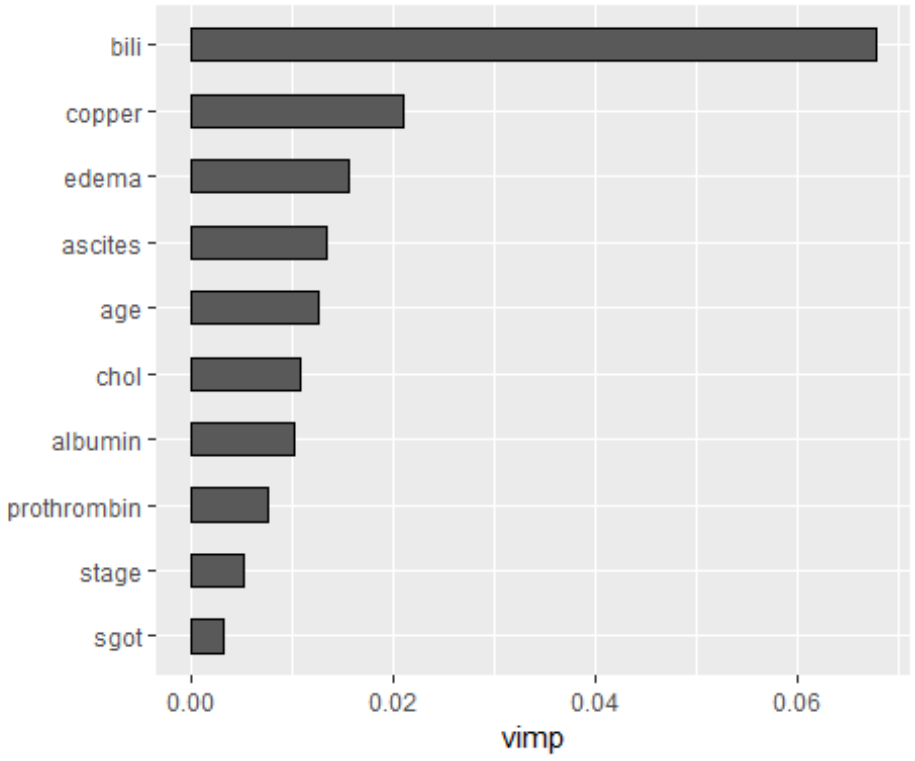
```
gg_dta
```

```
##           vars  set          vimp positive
## 1         bili VIMP  0.0689719956     TRUE
## 2       copper VIMP  0.0213892597     TRUE
## 3        edema VIMP  0.0161389117     TRUE
## 4      ascites VIMP  0.0138731402     TRUE
## 5          age VIMP  0.0124575963     TRUE
## 6      albumin VIMP  0.0112094109     TRUE
## 7         chol VIMP  0.0109248146     TRUE
## 8  prothrombin VIMP  0.0077009750     TRUE
## 9        stage VIMP  0.0047462763     TRUE
## 10        sgot VIMP  0.0040700859     TRUE
## 11         alk VIMP  0.0021514165     TRUE
## 12        trig VIMP  0.0019009172     TRUE
## 13     hepatom VIMP  0.0016792764     TRUE
## 14         sex VIMP  0.0013872642     TRUE
## 15    platelet VIMP -0.0001474261    FALSE
## 16   treatment VIMP -0.0002531301    FALSE
## 17     spiders VIMP -0.0003258470    FALSE
```

```
plot(gg_dta)
```




```
gg_dta_10<-gg_vimp(rfsrc_pbc, nvar=10)

## Warning in gg_vimp.rfsrc(rfsrc_pbc, nvar = 10): rfsrc object does not
## contain VIMP information. Calculating...

gg_dta_10

##           vars  set        vimp positive
## 1         bili VIMP 0.067855241     TRUE
## 2       copper VIMP 0.021075369     TRUE
## 3        edema VIMP 0.015652838     TRUE
## 4      ascites VIMP 0.013474512     TRUE
## 5          age VIMP 0.012696169     TRUE
## 6         chol VIMP 0.010789392     TRUE
## 7      albumin VIMP 0.010124282     TRUE
## 8  prothrombin VIMP 0.007659104     TRUE
## 9        stage VIMP 0.005156435     TRUE
## 10        sgot VIMP 0.003290305     TRUE

plot(gg_dta_10)
```




```
plot(rfsrc_pbc, lbls = st.labs) +
  theme(legend.position = c(0.8,0.2)) +
  labs(fill = "VIMP > 0") +
  scale_fill_brewer(palette = "Set1")
```

Return an object with both minimal depth and vimp measures

```
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        : 276
## ntree              : 1000
## nsplit             : 0
## mtry               : 5
## nodesize           : 3
## refitted forest    : FALSE
## model size         : 12
## depth threshold    : 7.9681
## PE (true OOB)      : 17.4626
```

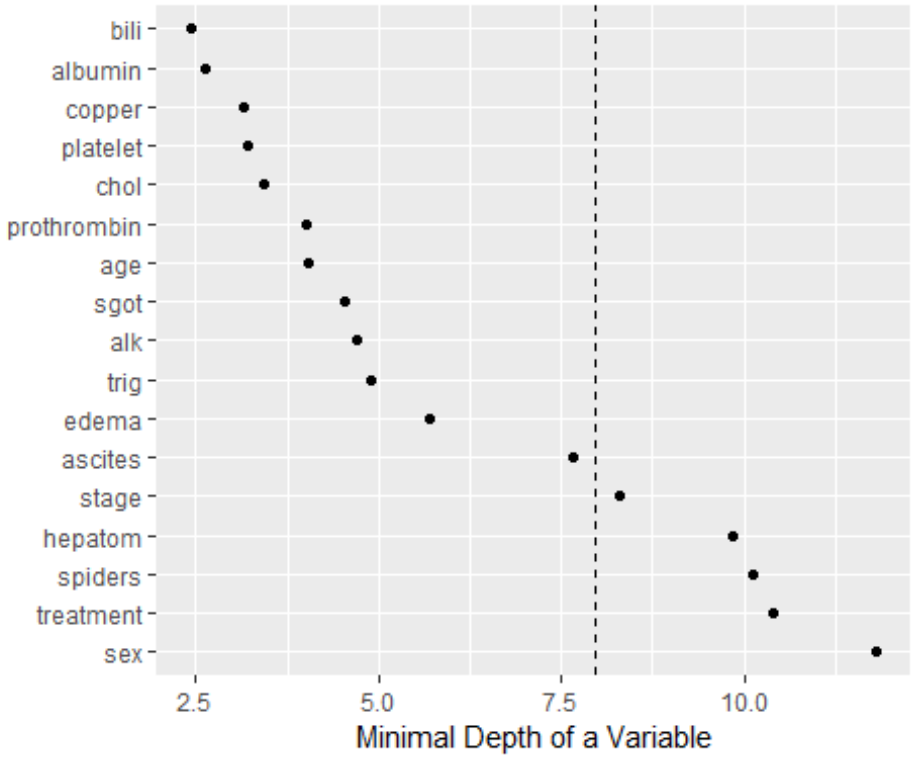
```
##
##
## Top variables:
##             depth vimp
## bili        2.431   NA
## albumin     2.617   NA
## copper      3.139   NA
## platelet    3.193   NA
## chol        3.434   NA
## prothrombin 3.995   NA
## age         4.029   NA
## sgot        4.531   NA
## alk         4.708   NA
## trig        4.893   NA
## edema       5.706   NA
## ascites     7.658   NA
## ------------------------------------------------------------
```

```
ggMindepth <- gg_minimal_depth(varsel_pbc, lbls = Years)
print(ggMindepth)
```

```
## ------------------------------------------------------------
## gg_minimal_depth
## model size         : 12
## depth threshold    : 7.9681
##
## PE :[1] 17.463
## ------------------------------------------------------------
##
## Top variables:
##             depth vimp
## bili         2.43   NA
## albumin      2.62   NA
## copper       3.14   NA
## platelet     3.19   NA
## chol         3.43   NA
## prothrombin  4.00   NA
## age          4.03   NA
## sgot         4.53   NA
## alk          4.71   NA
## trig         4.89   NA
## edema        5.71   NA
## ascites      7.66   NA
## ------------------------------------------------------------
```

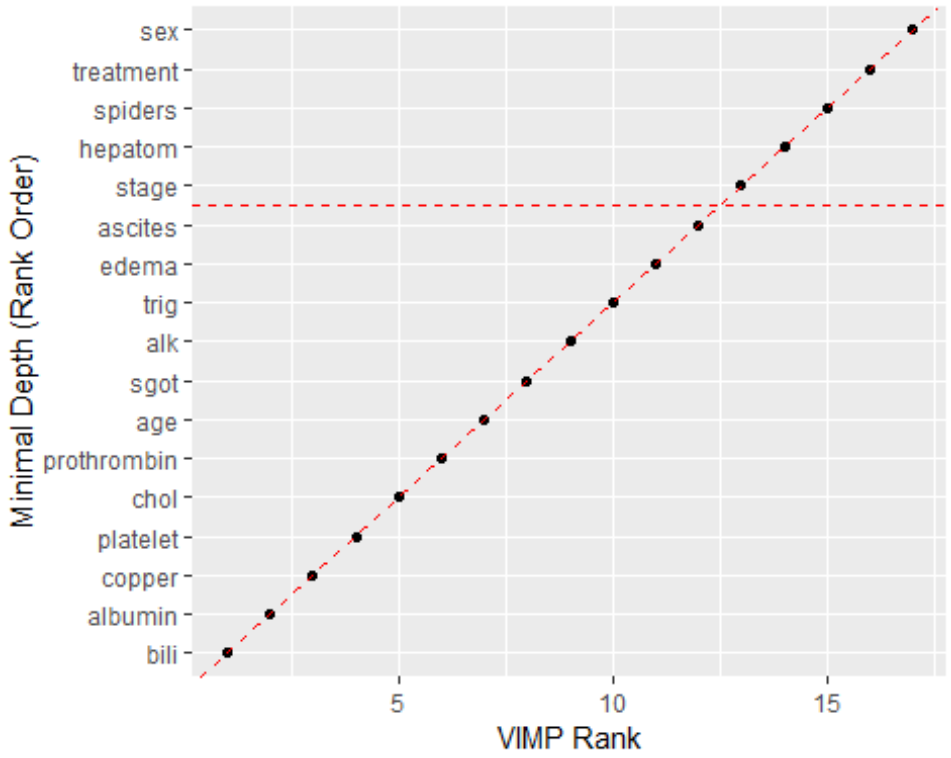
```
plot(ggMindepth)
```

```
## Coordinate system already present. Adding new coordinate system, which
will replace the existing one.
```



Both minimal depth and VIMP

```
plot(gg_minimal_vimp(ggMindepth))
```



Get the minimal depth selected variables

```
xvar <- varsel_pbc$topvars
xvar
```

```
##  [1] "bili"        "albumin"     "copper"      "platelet"    "chol"
##  [6] "prothrombin" "age"         "sgot"        "alk"         "trig"
## [11] "edema"       "ascites"
```

Data generation

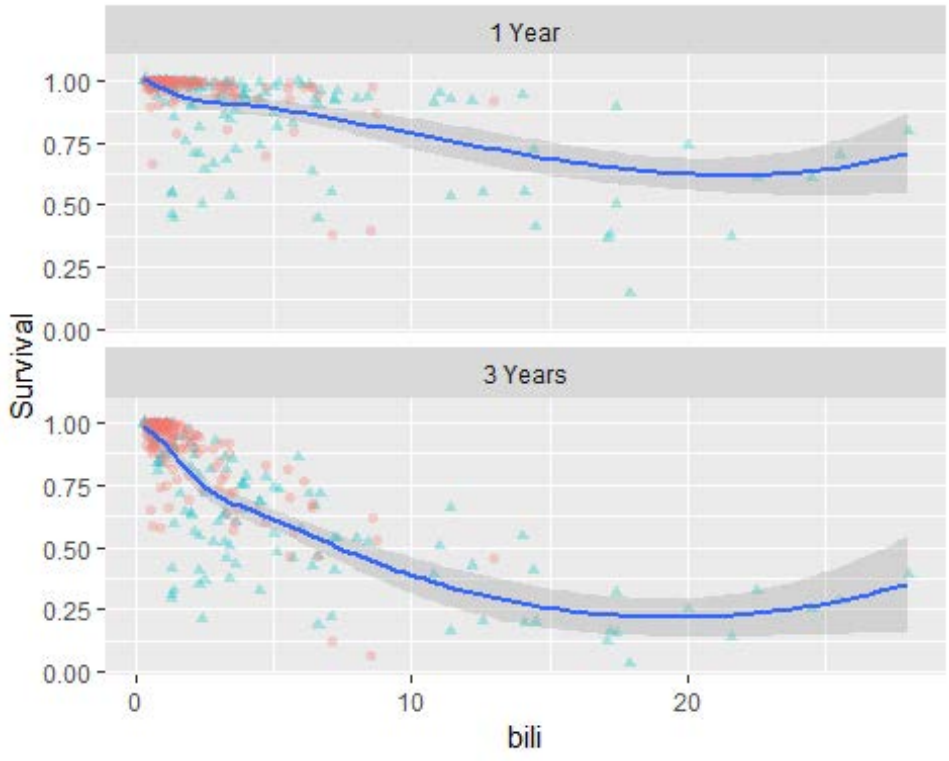
```
ggrf <- gg_variable(rfsrc_pbc, time = c(1, 3),
                    time.labels = c("1 Year", "3 Years"))
```

Plot the bilirubin variable dependence plot

```
plot(ggrf, xvar = "bili", se = .95, alpha = .3) +
  labs(y = "Survival", x = "bili") +
  theme(legend.position = "none")
```

```
## Warning: Ignoring unknown parameters: se
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

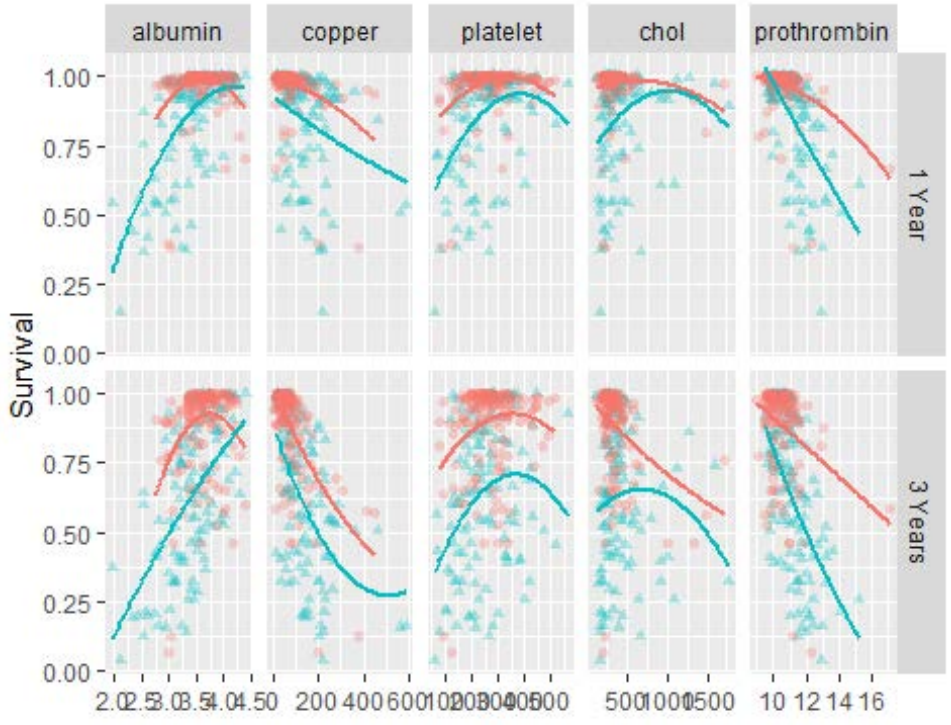


Pull the categorical variables

```
xvar.cat <- c("edema", "stage")
xvar <- xvar[-which(xvar %in% xvar.cat)]
```

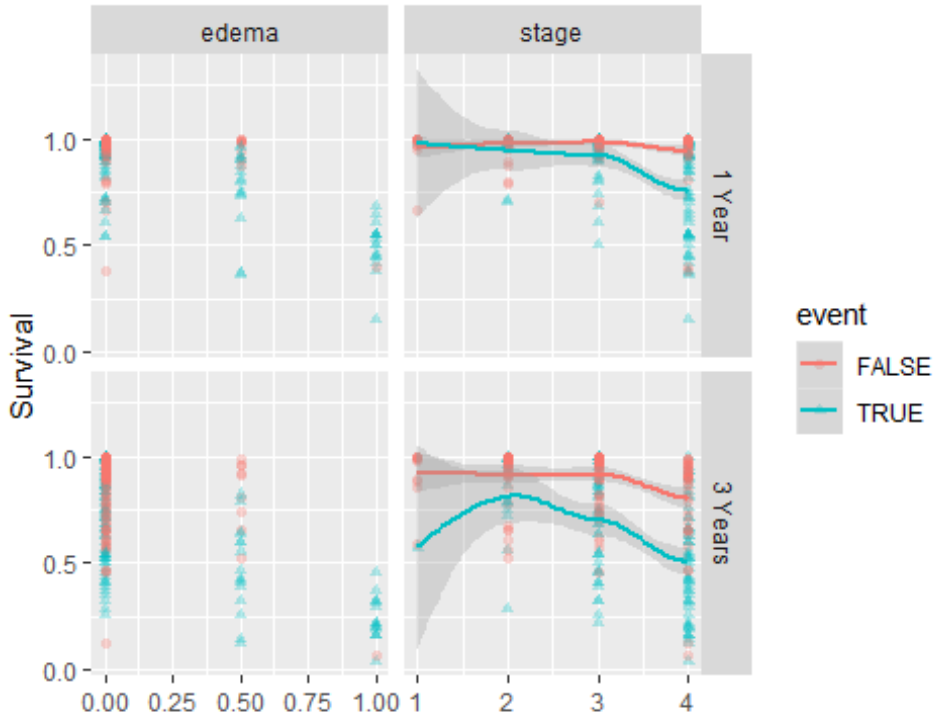
plot the next 5 continuous variable dependence plots.

```
plot(ggrf, xvar = xvar[2:6], panel = TRUE,
     se = FALSE, alpha = .3,
     method = "glm", formula = y~poly(x,2)) +
  labs(y = "Survival") +
  theme(legend.position = "none") #optional

## Warning: Ignoring unknown parameters: se, method, formula
```



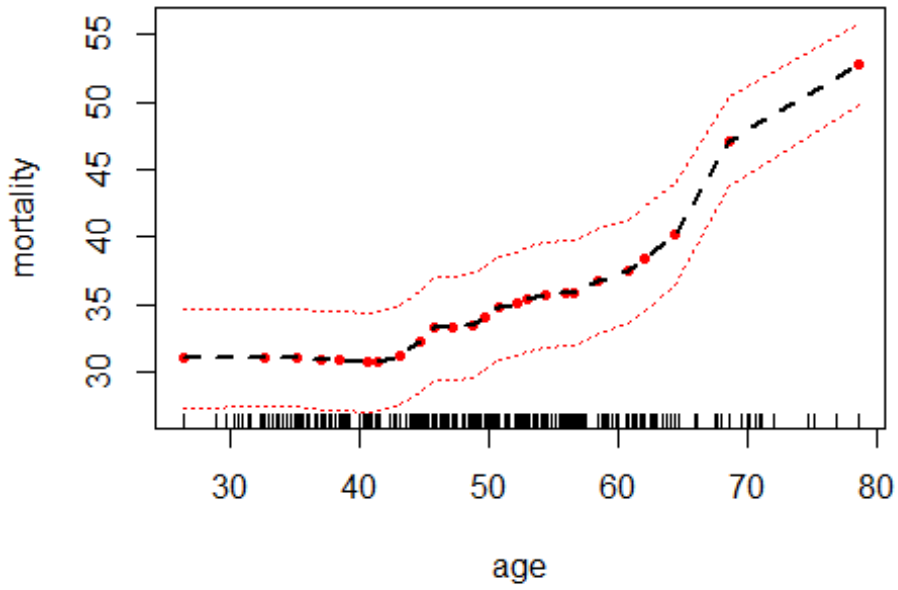
Variable dependence plots for categorical variables are constructed using boxplots to show the distribution of the predictions within each category.

```
plot(ggrf, xvar = xvar.cat, panel = TRUE, notch = TRUE, alpha = .3) +
  labs(y = "Survival") + scale_fill_gradientn(colours =
colorspace::rainbow_hcl(17))
```

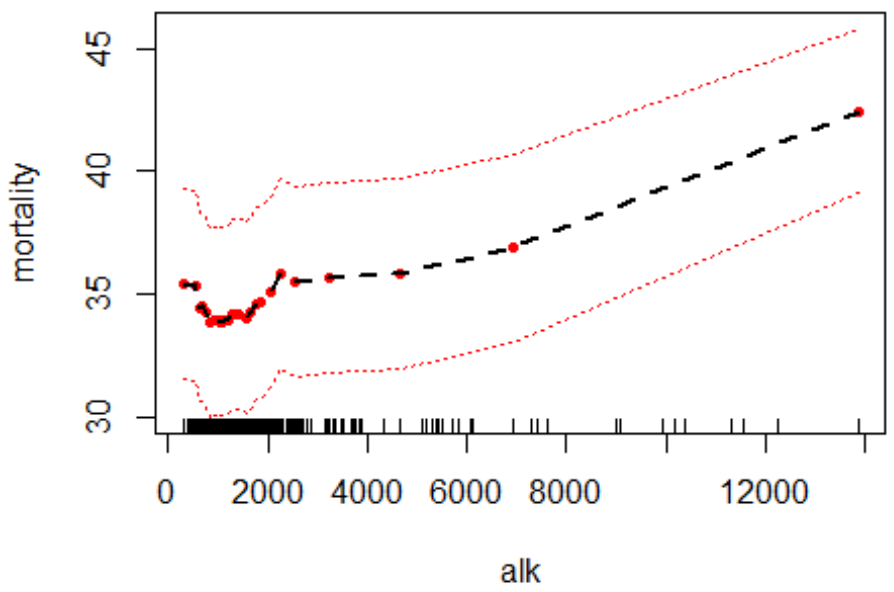


Calculate the 1 and 3 year partial dependence

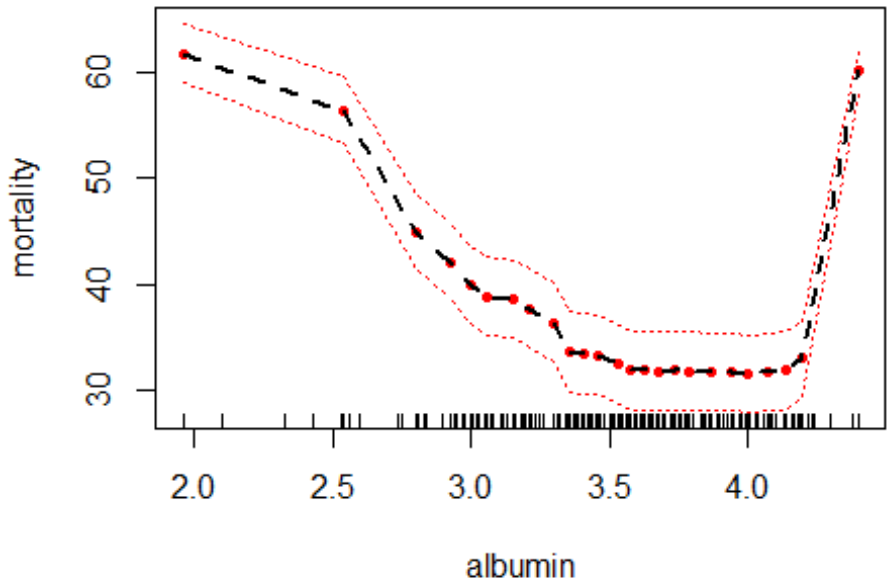
```
partial_age  <- plot.variable(rfsrc_pbc, xvar.names = "age",     partial=TRUE)
```



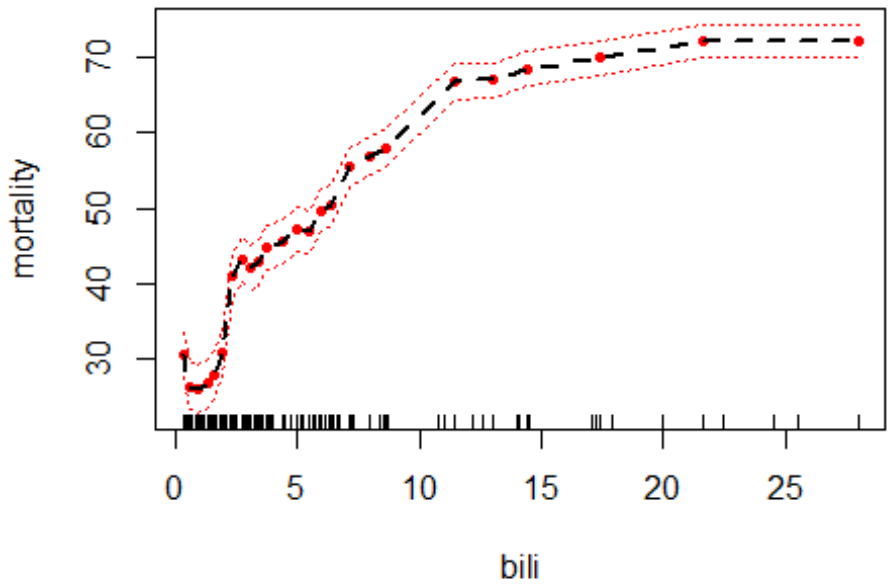
```
partial_alk  <- plot.variable(rfsrc_pbc, xvar.names = "alk",     partial=TRUE)
```



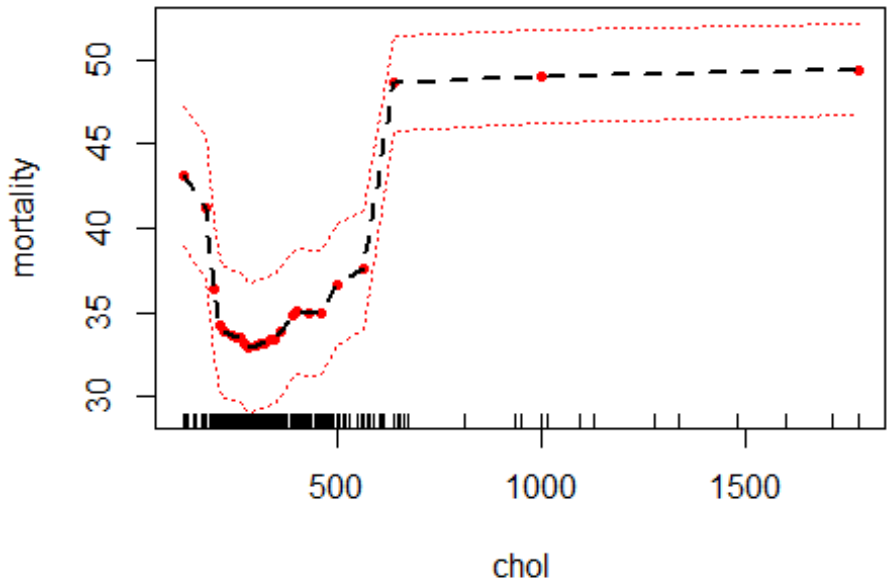

```
partial_alb  <- plot.variable(rfsrc_pbc, xvar.names = "albumin",partial=TRUE)
```



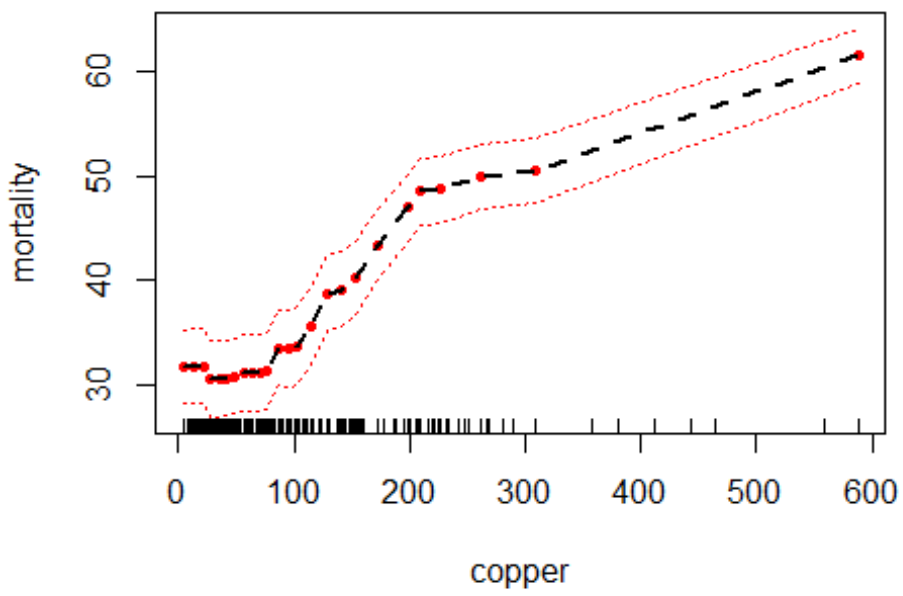
```
partial_bili <- plot.variable(rfsrc_pbc, xvar.names = "bili",   partial=TRUE)
```



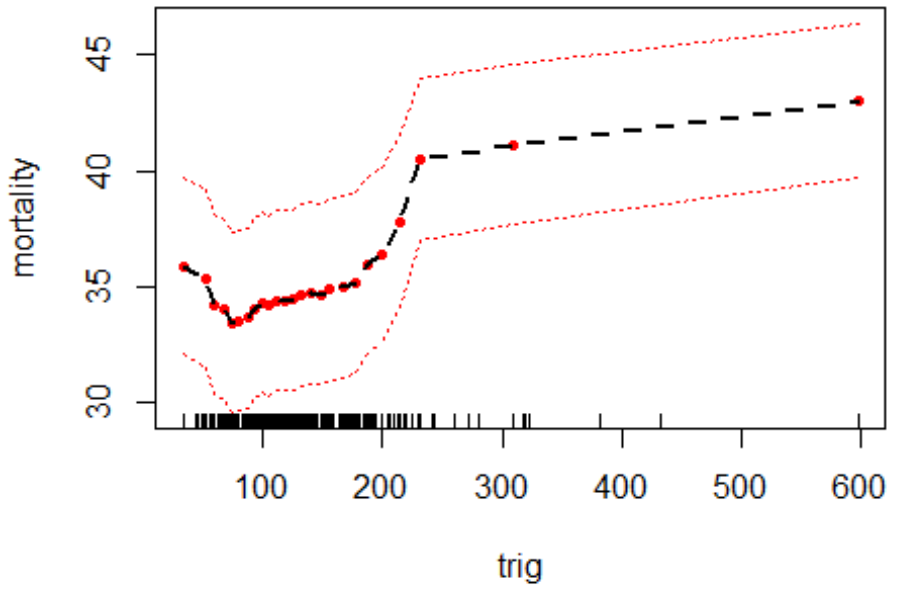

```
partial_chol <- plot.variable(rfsrc_pbc, xvar.names = "chol",   partial=TRUE)
```



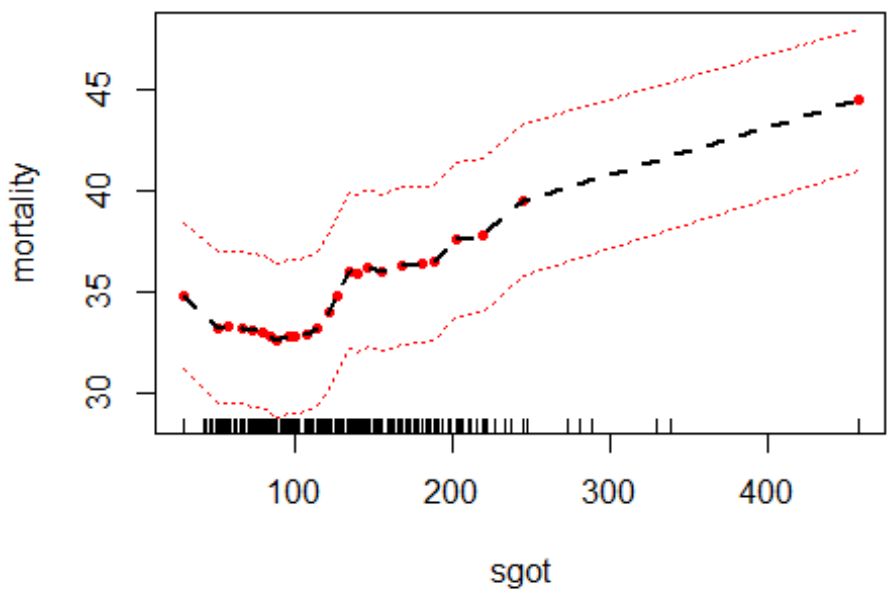
```
partial_copp <- plot.variable(rfsrc_pbc, xvar.names = "copper", partial=TRUE)
```



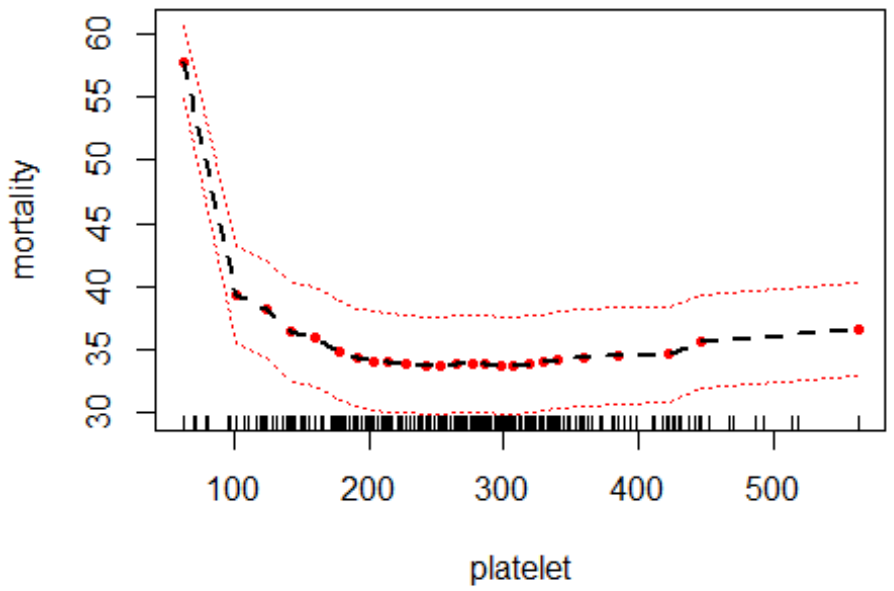

```
partial_trig <- plot.variable(rfsrc_pbc, xvar.names = "trig",   partial=TRUE)
```



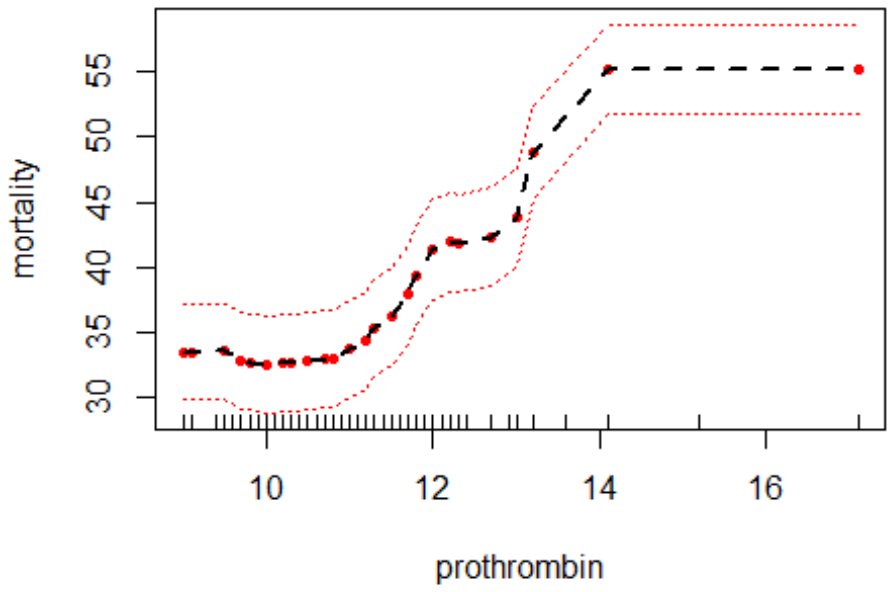
```
partial_sgot <- plot.variable(rfsrc_pbc, xvar.names = "sgot",   partial=TRUE)
```



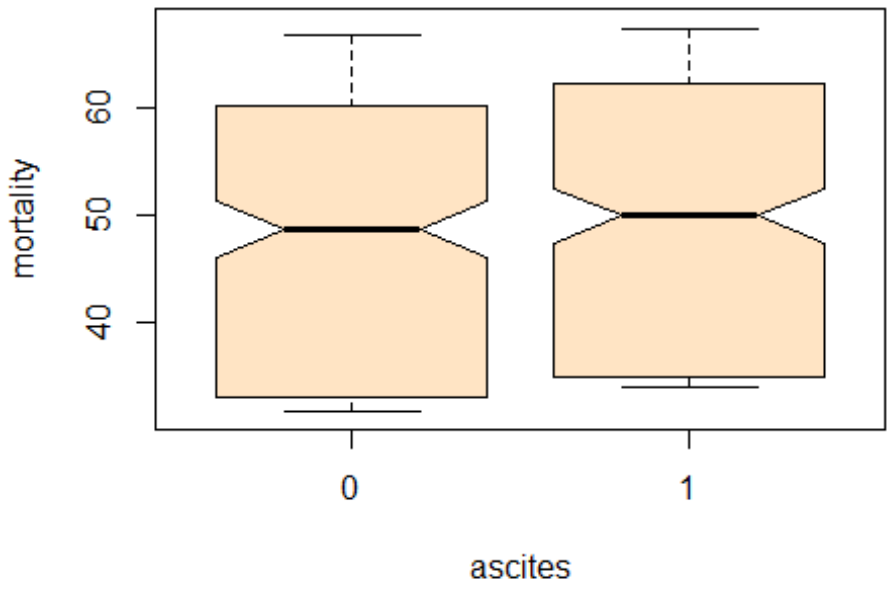

```
partial_plat <- plot.variable(rfsrc_pbc, xvar.names = "platelet",
partial=TRUE)
```



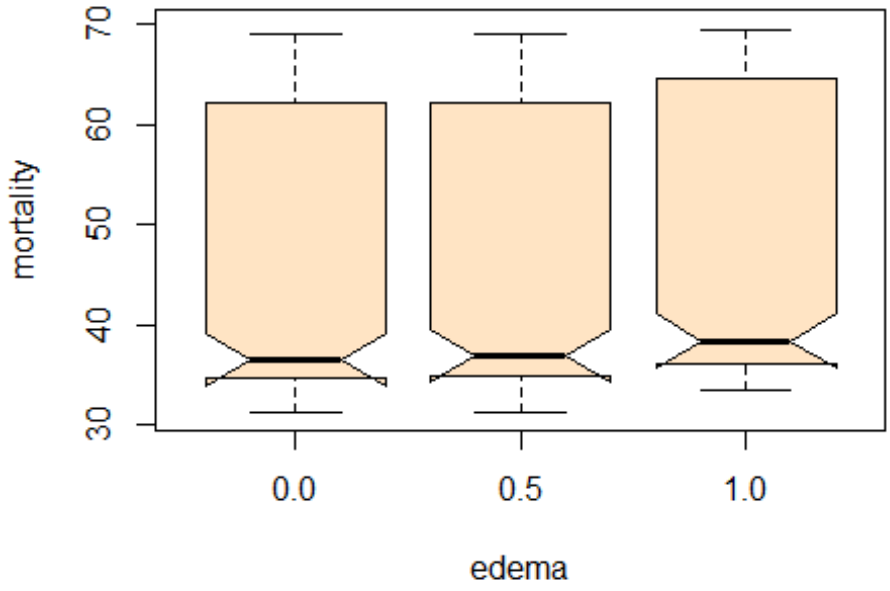

```
partial_pro <-  plot.variable(rfsrc_pbc, xvar.names = "prothrombin",
partial=TRUE)
```



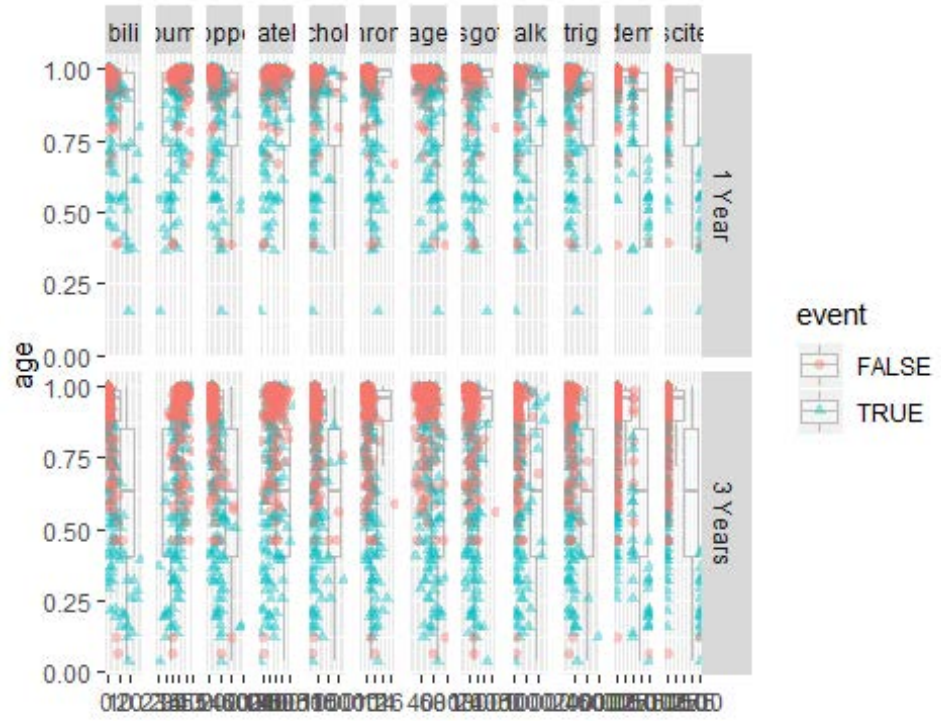
```
partial_asci <- plot.variable(rfsrc_pbc, xvar.names = "ascites",partial=TRUE)
```




```
partial_edema<- plot.variable(rfsrc_pbc, xvar.names = "edema",  partial=TRUE)

## Warning in bxp(list(stats = structure(c(31.2293708546758,
## 34.6231334170118, : some notches went outside hinges ('box'): maybe set
## notch=FALSE
```



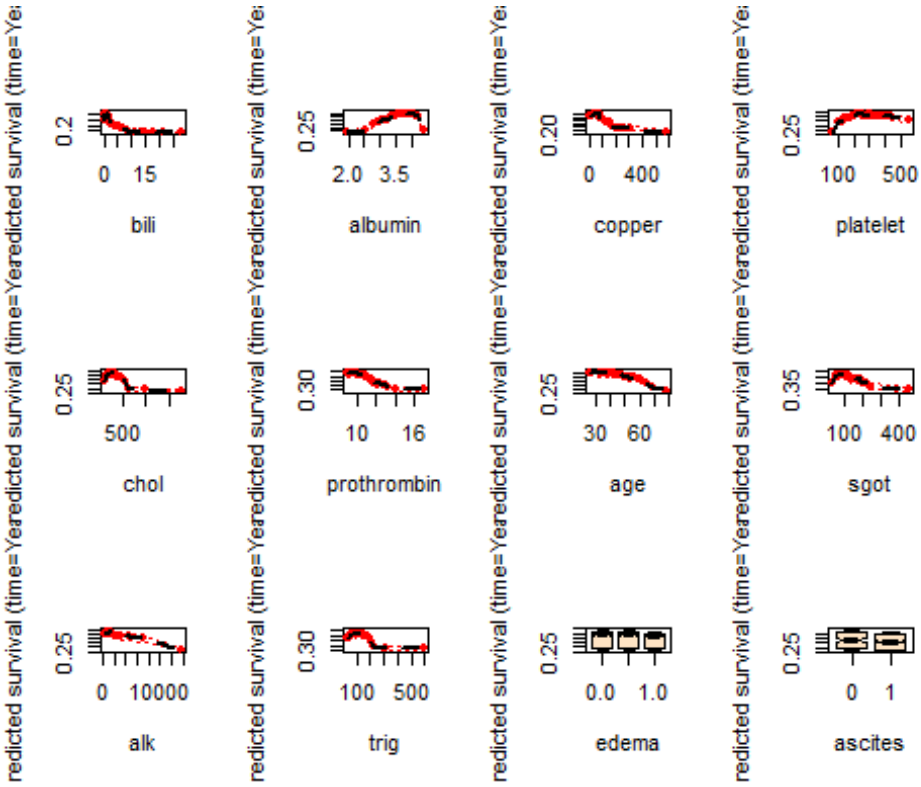

```
xvar <- ggMindepth$topvars
plot(ggrf, xvar=xvar, panel=TRUE, alpha=.4) + labs(y="age", x="")

## Warning in plot.gg_variable(ggrf, xvar = xvar, panel = TRUE, alpha = 0.4):
## Mismatched variable types for panel plots... assuming these are all factor
## variables.
```



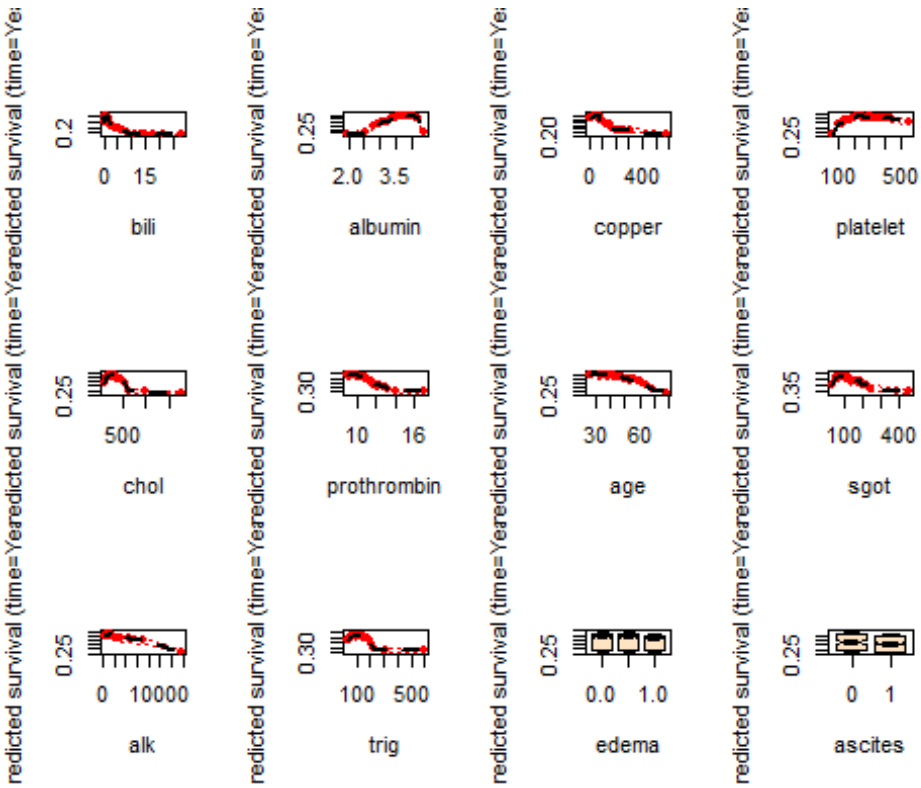
Calculate the 1, 3 and 5 year partial dependence

```
partial_pbc <- lapply(c(1,3,5), function(tm){
  plot.variable(rfsrc_pbc, surv.type = "surv",
                time = "Years",
                xvar.names = xvar, partial = TRUE,
                show.plots = TRUE)
  })
```

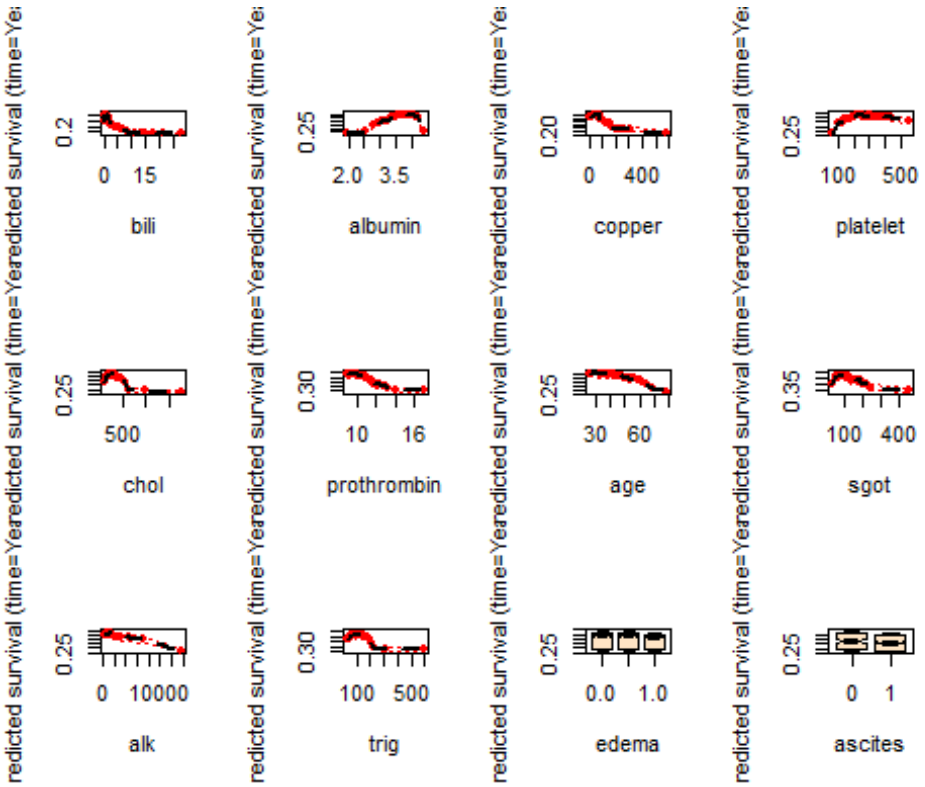
```
## Warning in bxp(list(stats = structure(c(0.253056959916302,
## 0.294541272957182, : some notches went outside hinges ('box'): maybe set
## notch=FALSE
```




```
## Warning in bxp(list(stats = structure(c(0.253056959916302,
## 0.294541272957182, : some notches went outside hinges ('box'): maybe set
## notch=FALSE
```

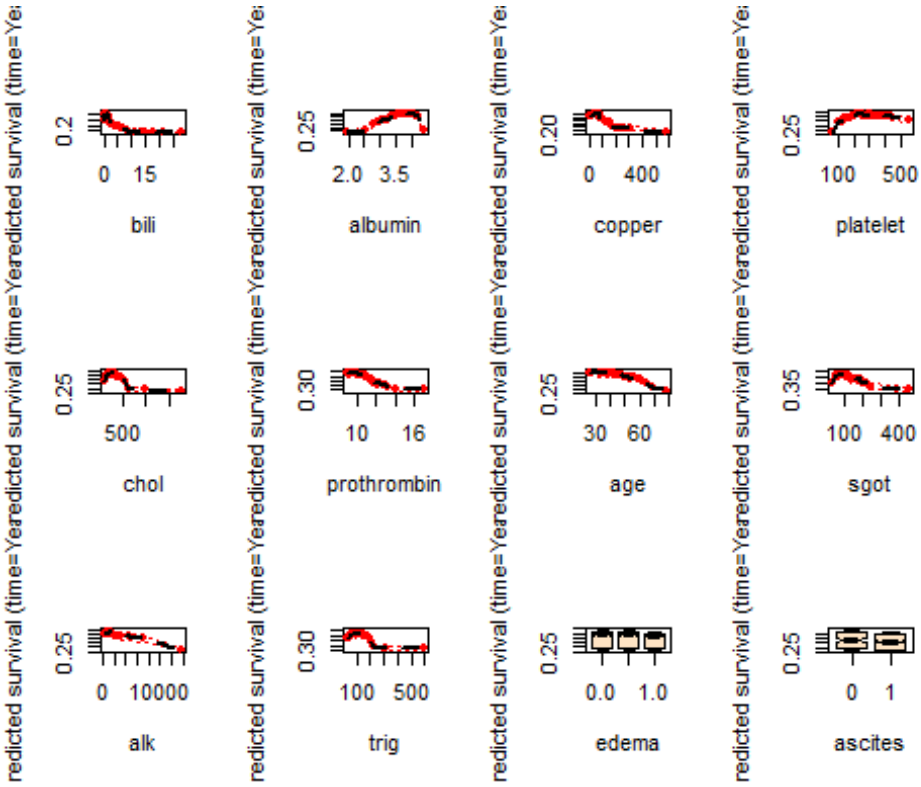


```
## Warning in bxp(list(stats = structure(c(0.253056959916302,
## 0.294541272957182, : some notches went outside hinges ('box'): maybe set
## notch=FALSE
```




```
plot.variable(rfsrc_pbc, surv.type = "surv",
              time = "Years",
              xvar.names = xvar, partial = TRUE,
              show.plots = TRUE)
```

```
## Warning in bxp(list(stats = structure(c(0.253056959916302,
## 0.294541272957182, : some notches went outside hinges ('box'): maybe set
## notch=FALSE
```

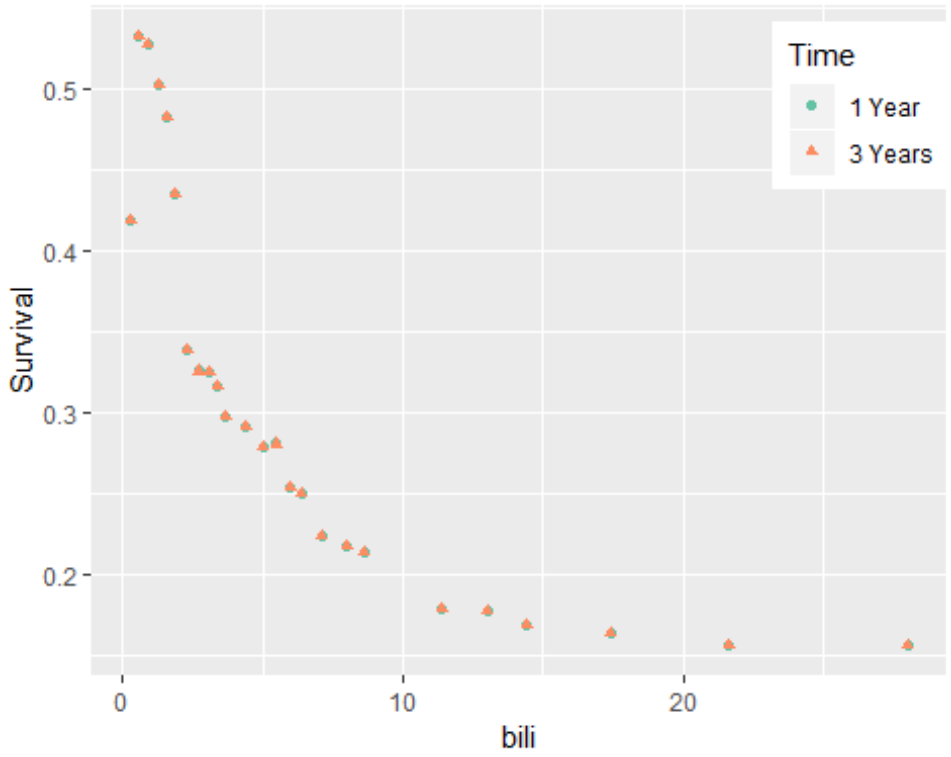


Convert all partial plots to gg_partial objects

```
gg_dta <- lapply(partial_pbc, gg_partial)
```

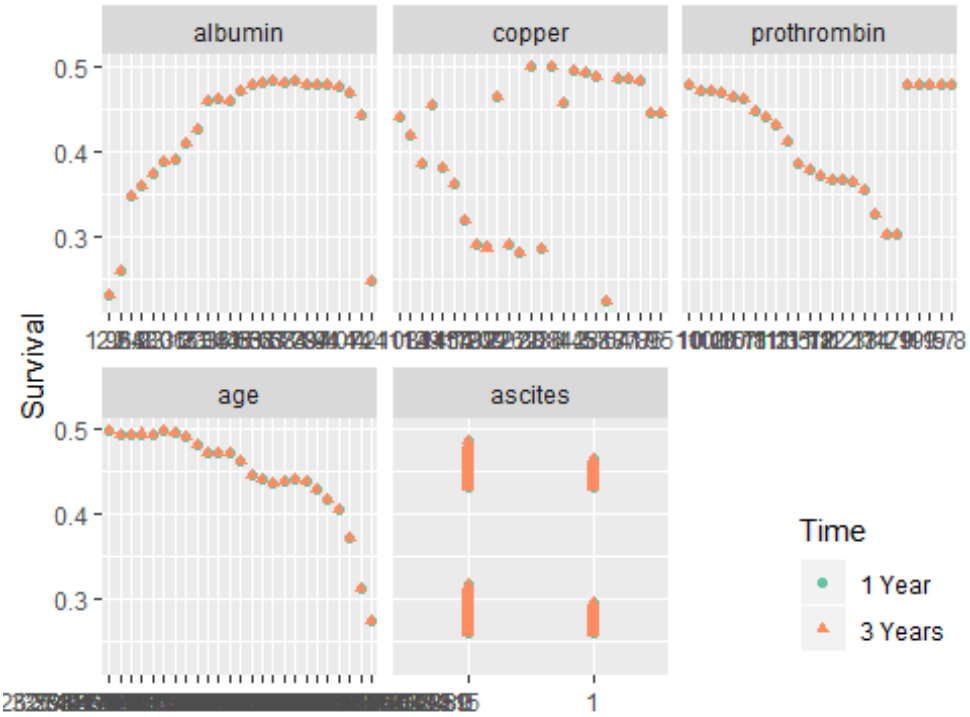
Combine the objects to get multiple time curves along variables on a single figure.

```
gg_dta <- lapply(partial_pbc, gg_partial)
pbc_ggpart <- combine.gg_partial(gg_dta[[1]], gg_dta[[2]], lbls = c("1 Year",
"3 Years"))
#pbc_ggpart2 <- combine.gg_partial(ggRandomForests::gg_partial(gg_dta[[1]],
gg_dta[[2]]), lbls = c("1 Year", "3 Years"))
plot(pbc_ggpart[["bili"]]) +
  theme(legend.position = c(.9, .85)) +
  labs(y = "Survival",
       x = "bili",
       color = "Time", shape = "Time") +
  scale_color_brewer(palette = "Set2")
```



Create a temporary holder and remove the stage and edema data

```
ggpart <- pbc_ggpart
ggpart$edema <- ggpart$stage <- NULL
ggpart$bili <- ggpart$sgot <- ggpart$chol <- NULL
ggpart$platelet <- ggpart$trig <- ggpart$alk <- NULL
# Panel plot the remainder.
plot(ggpart, panel = TRUE) +
  labs(x = "", y = "Survival", color = "Time", shape = "Time") +
  scale_color_brewer(palette = "Set2") +
  theme(legend.position = c(.9, .15))
```




```
ggpart <- pbc_ggpart
head(ggpart$edema)
```

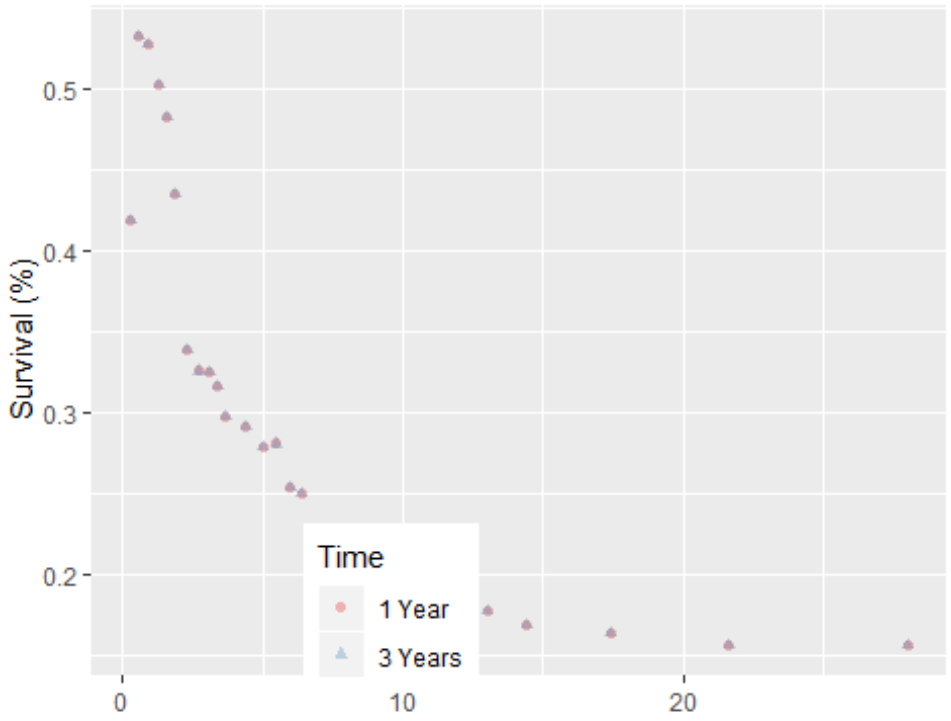
```
##           yhat edema   group
## 1    0.4341768     0  1 Year
## 2    0.4317110     0  1 Year
## 3    0.2670310     0  1 Year
## 4    0.4519583     0  1 Year
## 5    0.4299843     0  1 Year
```

```
ggpart$stage
```

```
## NULL
```

```
names(ggpart) <- c("edema", "stage")
class(ggpart) <- c("gg_partial_list", class(ggpart))
plot(ggpart$edema, panel=TRUE, notch = TRUE, alpha = .3) +
  labs(x = "", y = "Survival (%)", color="Time", shape="Time") +
  scale_color_brewer(palette = "Set1") +
  theme(legend.position = c(.35, .1))
```

```
## Warning: Ignoring unknown parameters: panel, notch
```



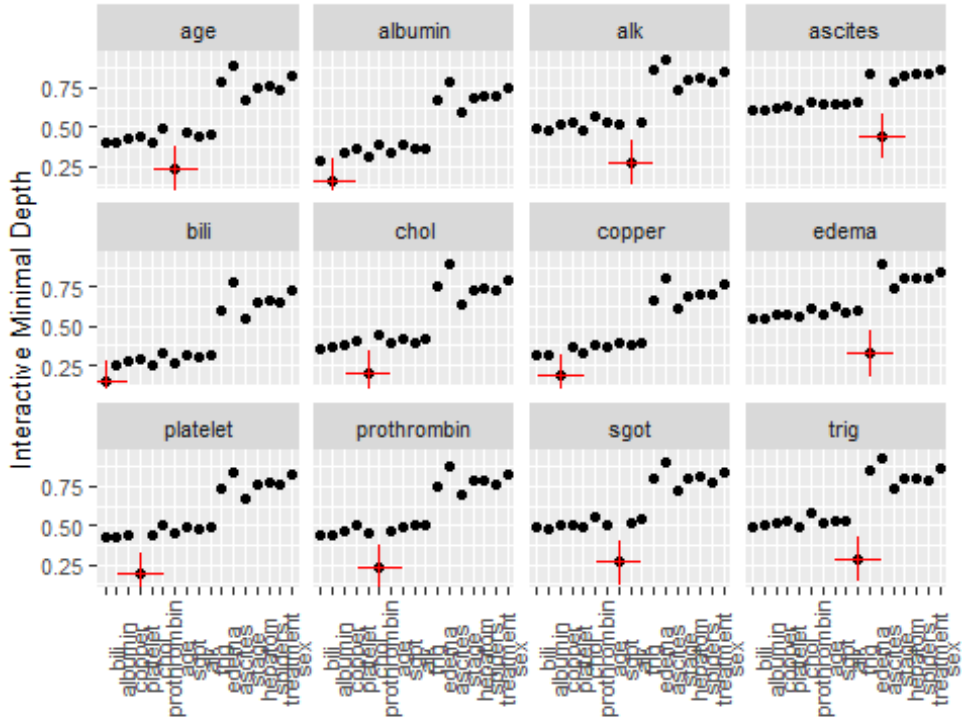
The gg_interaction function wraps the find.interaction matrix for use with the provided S3 plot and print functions.

```
interaction_pbc <- find.interaction(rfsrc_pbc)
```

```
##
##                               Method: maxsubtree
##                     No. of variables: 17
##   Variables sorted by minimal depth?: TRUE
##
##             bili albumin copper platelet chol prothrombin  age sgot  alk
## bili        0.14    0.25   0.28     0.29 0.25        0.33 0.26 0.32 0.30
## albumin     0.29    0.16   0.33     0.36 0.31        0.39 0.33 0.38 0.37
## copper      0.32    0.31   0.18     0.37 0.33        0.39 0.36 0.39 0.38
## platelet    0.43    0.43   0.44     0.19 0.44        0.50 0.46 0.49 0.48
## chol        0.36    0.36   0.38     0.41 0.20        0.44 0.39 0.41 0.40
## prothrombin 0.44    0.44   0.46     0.50 0.46        0.23 0.46 0.49 0.50
## age         0.40    0.40   0.43     0.44 0.40        0.49 0.24 0.47 0.44
## sgot        0.49    0.48   0.51     0.50 0.49        0.55 0.50 0.27 0.51
## alk         0.48    0.47   0.51     0.53 0.48        0.57 0.53 0.52 0.27
## trig        0.49    0.50   0.52     0.53 0.49        0.58 0.52 0.53 0.53
## edema       0.55    0.55   0.57     0.58 0.56        0.62 0.58 0.62 0.58
## ascites     0.61    0.61   0.62     0.63 0.61        0.66 0.65 0.65 0.64
## stage       0.71    0.73   0.73     0.75 0.72        0.75 0.73 0.74 0.73
## hepatom     0.82    0.83   0.82     0.83 0.81        0.83 0.82 0.83 0.83
## spiders     0.85    0.85   0.85     0.85 0.83        0.85 0.85 0.85 0.85
## treatment   0.91    0.91   0.91     0.90 0.91        0.91 0.91 0.90 0.90
```

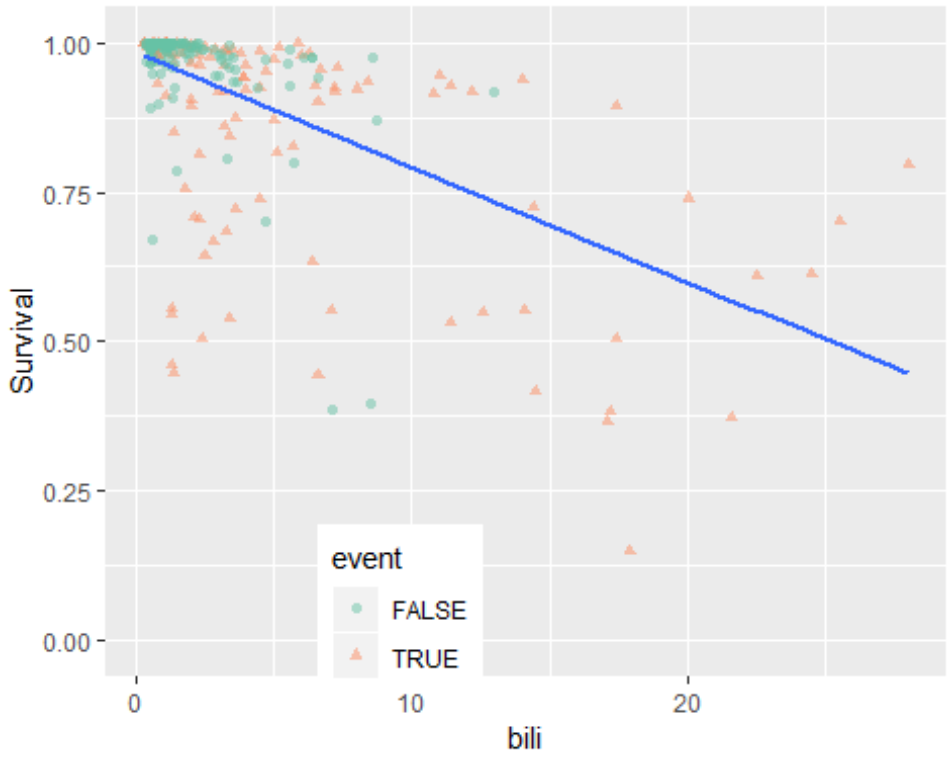
```
## sex          0.89    0.89   0.91     0.90 0.89        0.90 0.90 0.90 0.89
##              trig edema ascites stage hepatom spiders treatment  sex
## bili         0.32  0.59    0.78  0.55    0.65    0.66      0.65 0.73
## albumin      0.36  0.67    0.79  0.60    0.69    0.70      0.69 0.75
## copper       0.39  0.67    0.81  0.62    0.70    0.71      0.70 0.77
## platelet     0.49  0.74    0.84  0.68    0.76    0.78      0.76 0.82
## chol         0.42  0.75    0.89  0.63    0.72    0.74      0.73 0.79
## prothrombin 0.50  0.75    0.88  0.70    0.78    0.78      0.76 0.83
## age          0.45  0.79    0.89  0.67    0.74    0.76      0.74 0.83
## sgot         0.54  0.80    0.91  0.72    0.80    0.81      0.77 0.84
## alk          0.53  0.86    0.93  0.74    0.81    0.81      0.79 0.85
## trig         0.29  0.85    0.93  0.74    0.80    0.80      0.79 0.87
## edema        0.60  0.33    0.90  0.75    0.80    0.81      0.81 0.84
## ascites      0.66  0.84    0.44  0.79    0.83    0.84      0.83 0.87
## stage        0.75  0.90    0.97  0.48    0.88    0.90      0.88 0.92
## hepatom      0.84  0.95    0.98  0.90    0.57    0.93      0.93 0.95
## spiders      0.86  0.95    0.98  0.90    0.94    0.59      0.92 0.96
## treatment    0.91  0.98    0.99  0.95    0.96    0.96      0.61 0.97
## sex          0.90  0.97    0.99  0.95    0.96    0.96      0.95 0.68
```

```r
ggint <- gg_interaction(interaction_pbc)
plot(ggint, xvar = xvar) +
  labs(y = "Interactive Minimal Depth") +
  theme(legend.position = "none")
```



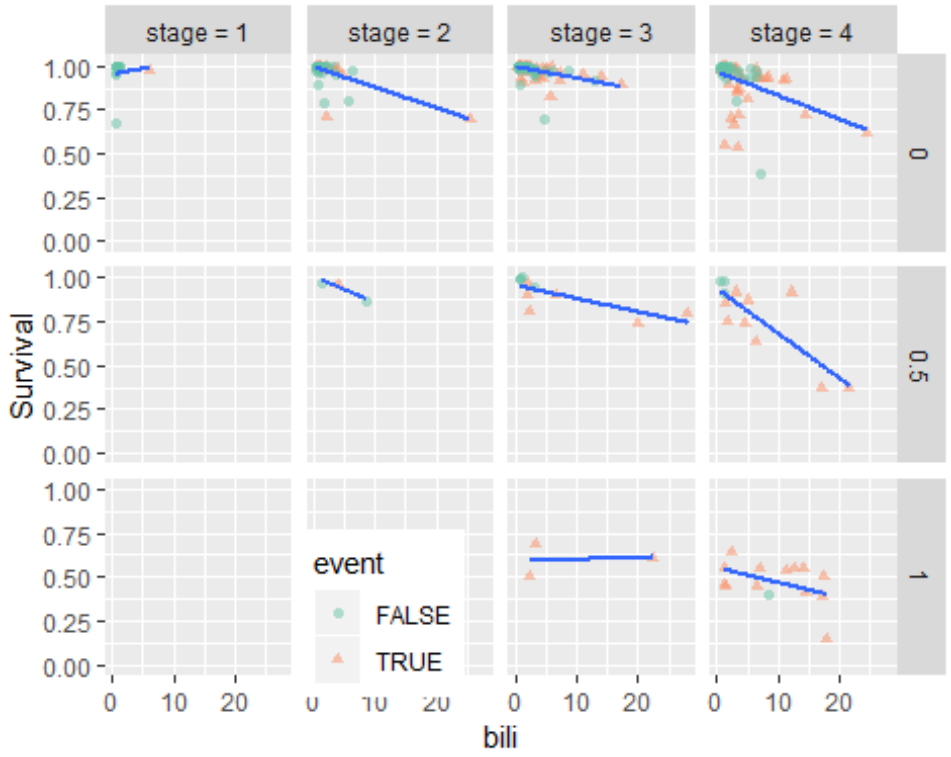
Conditional dependence plots

```
ggvar <- gg_variable(rfsrc_pbc, time = 1)
ggvar$stage <- paste("stage = ", ggvar$stage, sep = "")
var_dep <- plot(ggvar, xvar = "bili", method = "glm", alpha = .5, se = FALSE)
+
  labs(y = "Survival", x = "bili") +
  theme(legend.position = c(.35, .1)) +
  scale_color_brewer(palette = "Set2") +
  scale_shape(solid=TRUE) +
  coord_cartesian(y = c(-.01,1.01))

## Warning: Ignoring unknown parameters: method, se

var_dep
```



Conditional dependence of survival against bilirubin, versus other categorical covariates, say edema and stage.

```
var_dep +
  facet_grid(edema~stage)
```



Find intervals with similar number of observations.

```
copper_cts <-quantile_pts(ggvar$copper, groups = 6, intervals = TRUE)
```

Create the conditional groups and add to the gg_variable object

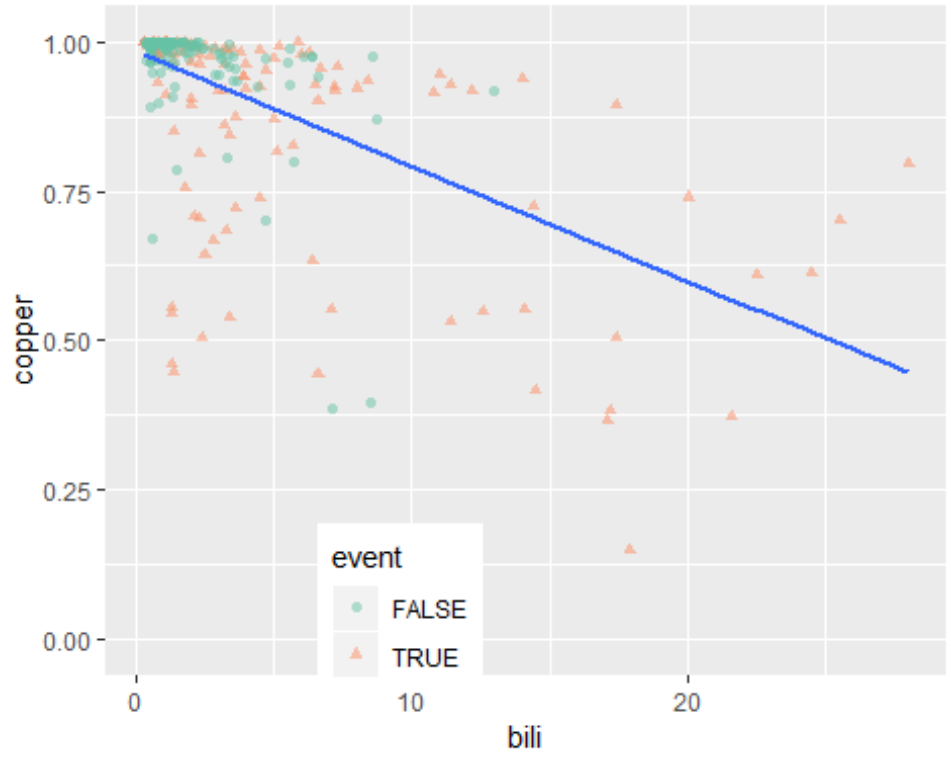
```
copper_grp <- cut(ggvar$copper, breaks = copper_cts)
ggvar$copper_grp <- copper_grp
```

Adjust naming for facets

```
levels(ggvar$copper_grp) <- paste("copper = ",levels(copper_grp), sep = "")
```

plot.gg_variable

```
var_dep <- plot(ggvar, xvar = "bili", method = "glm", alpha = .5, se = FALSE)
+
  labs(y = "copper", x = "bili") +
  theme(legend.position = c(.35, .1)) +
  scale_color_brewer(palette = "Set2") +
  scale_shape(solid=TRUE) +
  coord_cartesian(y = c(-.01,1.01))

## Warning: Ignoring unknown parameters: method, se

var_dep
```



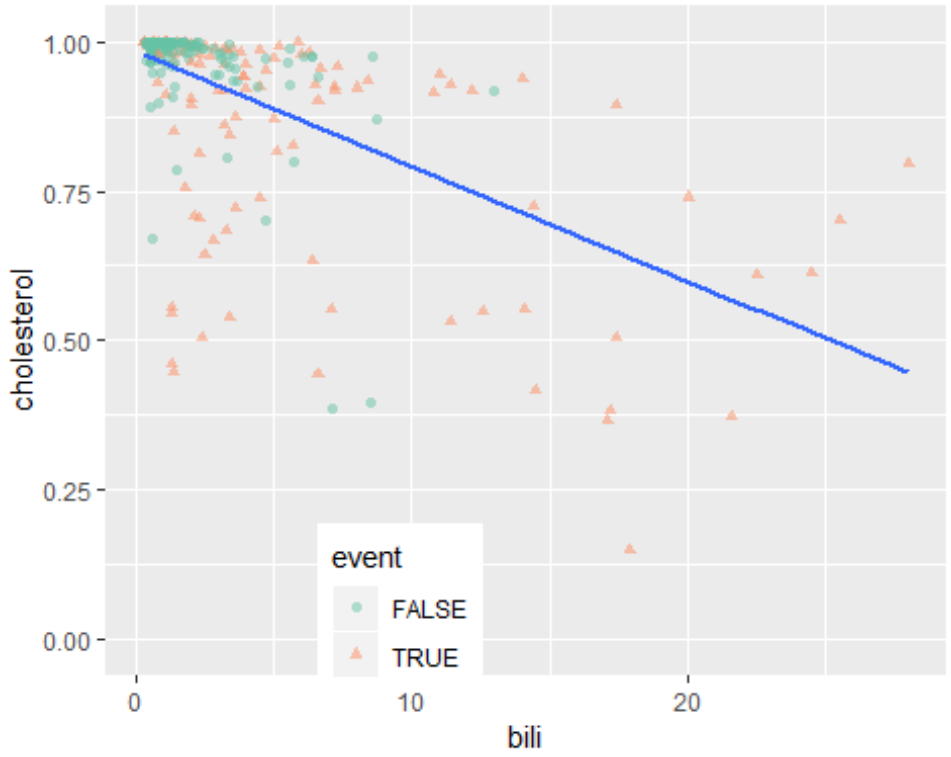

```
chol_cts <-quantile_pts(ggvar$chol, groups = 6, intervals = TRUE)
chol_grp <- cut(ggvar$chol, breaks = chol_cts)
ggvar$chol_grp <- chol_grp
```

Adjust naming for facets

```
levels(ggvar$chol_grp) <- paste("chol = ",levels(chol_grp), sep = "")
```

Plot.gg_variable

```
var_dep <- plot(ggvar, xvar = "bili", method = "glm", alpha = .5, se = FALSE)
+
  labs(y = "cholesterol", x = "bili") +
  theme(legend.position = c(.35, .1)) +
  scale_color_brewer(palette = "Set2") +
  scale_shape(solid=TRUE) +
  coord_cartesian(y = c(-.01,1.01))

## Warning: Ignoring unknown parameters: method, se

var_dep
```



Partial dependence coplots

```
data(rfsrc_pbc, package="ggRandomForests")

## Warning in data(rfsrc_pbc, package = "ggRandomForests"): data set
## 'rfsrc_pbc' not found
```

Create the variable plot.

```
ggvar <- gg_variable(rfsrc_pbc, time = 1)
```

Find intervals with similar number of observations.

```
copper_cts <-quantile_pts(ggvar$copper, groups = 6, intervals = TRUE)
```

Create the conditional groups and add to the gg_variable object

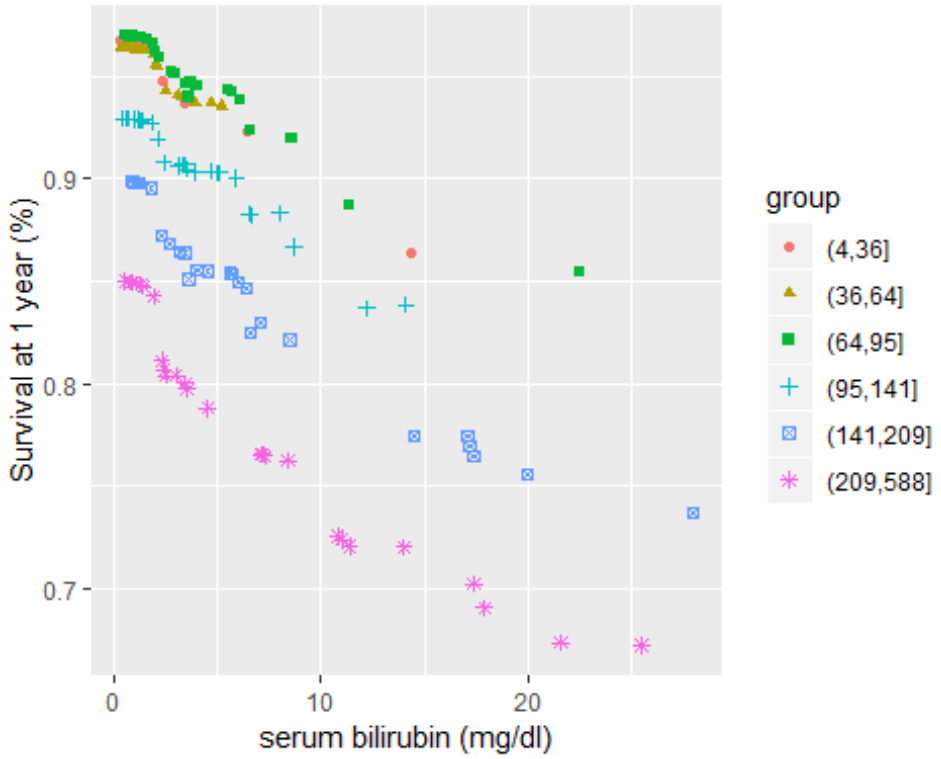
```
copper_grp <- cut(ggvar$copper, breaks = copper_cts)
partial_coplot_pbc <- gg_partial_coplot(rfsrc_pbc, xvar = "bili",
                                        groups = copper_grp,
                                        surv_type = "surv",
                                        time = 1,
                                        show.plots = FALSE)
```

so load the cached set

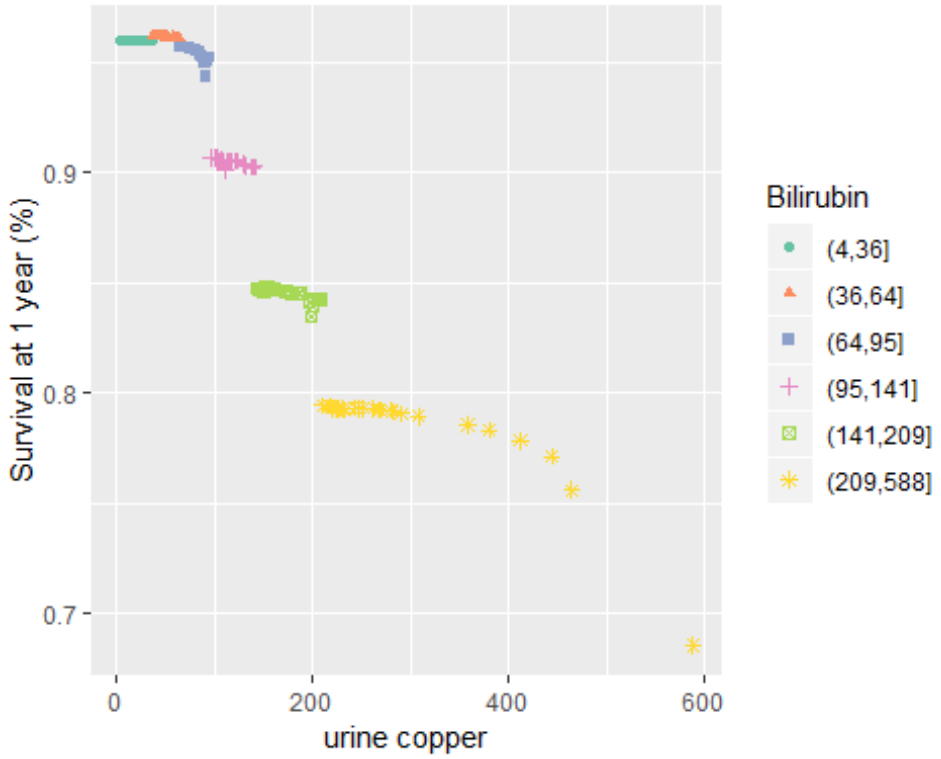
```
data(partial_coplot_pbc, package="ggRandomForests")

## Warning in data(partial_coplot_pbc, package = "ggRandomForests"): data set
## 'partial_coplot_pbc' not found
```

Partial coplot

```
plot(partial_coplot_pbc) +
  labs(x = "serum bilirubin (mg/dl)", y = "Survival at 1 year (%)")
```




```
partial_coplot_pbc <- gg_partial_coplot(rfsrc_pbc, xvar = "copper",
                                        groups = copper_grp,
                                        surv_type = "surv",
                                        time = 1,
                                        show.plots = FALSE)
data(partial_coplot_pbc, package = "ggRandomForests")

## Warning in data(partial_coplot_pbc, package = "ggRandomForests"): data set
## 'partial_coplot_pbc' not found

plot(partial_coplot_pbc) +
  labs(x = "urine copper",
       y = "Survival at 1 year (%)",
       color = "Bilirubin", shape = "Bilirubin") +
  scale_color_brewer(palette = "Set2")
```



Business Scenario and dataset

A marketing department of a bank runs various marketing campaigns for cross-selling products, improving customer retention and customer services. In this example, the bank wanted to cross-sell term deposit product to its customers.Contacting all customers is costly and does not create good customer experience. So, the bank wanted to build a predictive model which will identify customers who are more likely to respond to term deport cross sell campaign. We will use sample Marketing Data sample for building Random Forest based model using R.

Read and Explore data

```
bank<-
read.csv(file="C:/Users/Jeff/Documents/VIT_University/data/Banking.csv",heade
r = T)
names(bank)
```

```
##  [1] "job"       "marital"   "education" "age"       "balance"
##  [6] "homeowner" "loans"     "default"   "contact"   "length"
## [11] "campaign"  "pdays"     "previous"  "poutcome"  "RESP"
```

Input dataset has 20 independent variables and a target variable. The target variable y is binary.

```
table(bank$RESP)/nrow(bank)
```

```
##
##        NO       YES
## 0.8830178 0.1169822
```

Eleven % of the observations has target variable “yes” and remaining 89% observations take value “no”.

Now, we will split the data sample into development and validation samples.

```
sample.ind <- sample(2,
                     nrow(bank),
                     replace = T,
                     prob = c(0.5,0.5))
#sample.ind <- sample(2, 10000, replace=F)
cross.sell.dev <- bank[sample.ind==1,]
cross.sell.val <- bank[sample.ind==2,]

table(cross.sell.dev$RESP)/nrow(cross.sell.dev)

##
##        NO       YES
## 0.8824438 0.1175562

table(cross.sell.val$RESP)/nrow(cross.sell.val)

##
##        NO       YES
## 0.8835941 0.1164059

cross.sell.val$RESP

##     [1] NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO
##    [18] NO  NO  NO  YES NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  YES NO
##    [35] NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO
##    [52] NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO
##    [69] NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO
##    [86] NO  NO  NO  NO  NO  NO  NO  NO  YES NO  NO  NO  NO  NO  NO  NO  NO
##   [103] NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  YES NO  NO  NO  NO
##
##
##
## [22577] NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  NO  YES YES NO  NO
## [22594] NO  YES NO  NO  YES YES NO  YES NO  NO  YES NO  NO  YES YES NO  NO
## [22611] NO  YES YES NO  YES YES YES YES NO  NO  NO  NO  NO  YES NO  YES NO
## [22628] YES YES YES YES YES YES NO  YES NO  NO  YES NO  NO  NO  NO  NO  NO
## [22645] NO  NO  YES YES NO  YES YES YES YES
## Levels: NO YES
```

Both development and validation samples have similar target variable distribution. This is just a sample validation.

If target variable is factor, classification decision tree is built. We can check the type of response variable.

```
class(cross.sell.dev$RESP)

## [1] "factor"

class(cross.sell.val$RESP)

## [1] "factor"
```

Class of target or response variable is factor, so a classification Random Forest will be built. The current data frame has a list of independent variables, so we can make it formula and then pass as a parameter value for randomForest.

Make Formula

```
varNames <- names(cross.sell.dev)
```

Exclude ID or Response variable

```
varNames <- varNames[!varNames %in% c("RESP")]
```

add + sign between exploratory variables

```
varNames1 <- paste(varNames, collapse = "+")
```

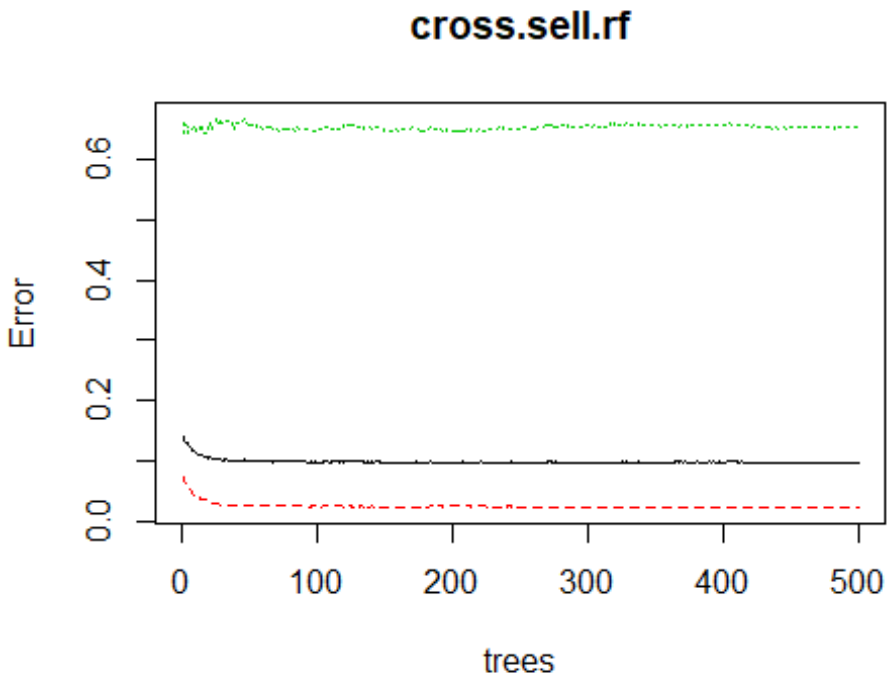
Add response variable and convert to a formula object

```
rf.form <- as.formula(paste("RESP", varNames1, sep = " ~ "))
```

Building Random Forest using R

Now, we have a sample data and formula for building Random Forest model. Letâ□□s build 500 decision trees using Random Forest.

```
cross.sell.rf <- randomForest(rf.form, cross.sell.dev,ntree=500,importance=T)

plot(cross.sell.rf)
```

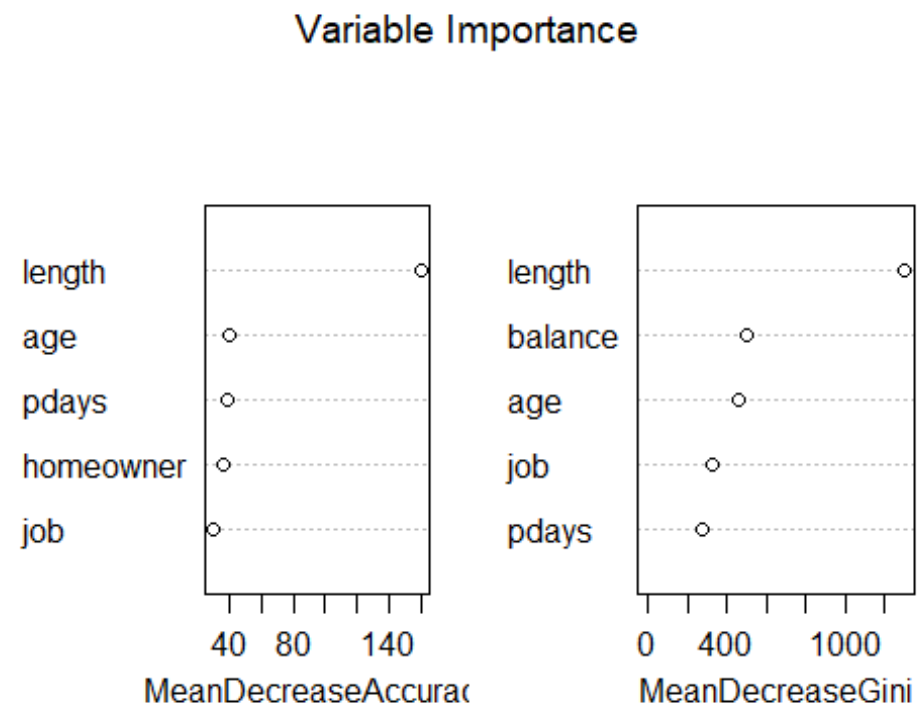


500 decision trees or a forest has been built using the Random Forest algorithm based learning. We can plot the error rate across decision trees. The plot seems to indicate that after 100 decision trees, there is not a significant reduction in error rate.

Variable importance plot is also a useful tool and can be plotted using varImpPlot function. Top 5 variables are selected and plotted based on Model Accuracy and Gini value. We can also get a table with decreasing order of importance based on a measure (1 for model accuracy and 2 node impurity)

Variable Importance Plot

```
varImpPlot(cross.sell.rf,
           sort = T,
           main="Variable Importance",
           n.var=5)
```



Variable Importance Table

```
var.imp <- data.frame(importance(cross.sell.rf,
           type=2))
```

Make row names as columns

```
var.imp$Variables <- row.names(var.imp)
var.imp[order(var.imp,decreasing = T),]
```

```
##           MeanDecreaseGini Variables
## NA                      NA      <NA>
## NA.1                    NA      <NA>
## NA.2                    NA      <NA>
## NA.3                    NA      <NA>
## NA.4                    NA      <NA>
## NA.5                    NA      <NA>
## NA.6                    NA      <NA>
## NA.7                    NA      <NA>
## NA.8                    NA      <NA>
## NA.9                    NA      <NA>
## NA.10                   NA      <NA>
## NA.11                   NA      <NA>
## NA.12                   NA      <NA>
## NA.13                   NA      <NA>
## length         1300.789733    length
## balance         497.901416   balance
## age             456.717286       age
```

```
## job             320.024358       job
## pdays           276.070140     pdays
## poutcome        217.999505  poutcome
## campaign        174.409370  campaign
## homeowner       125.707781 homeowner
## education       123.510237 education
## previous        119.876718  previous
## marital          99.500687   marital
## contact          98.514174   contact
## loans            47.061799     loans
## default           9.970018   default
```

Based on Random Forest variable importance, the variables could be selected for any other predictive modelling techniques or machine learning.

Now, we want to measure the accuracy of the Random Forest model. Some of the other model performance statistics are: - KS - Lift Chart - ROC curve

Predict Response Variable Value using Random Forest

Generic predict function can be used for predicting response variable using Random Forest object.

Predicting response variable

cross.sell.dev$predicted.response <- predict(cross.sell.rf ,cross.sell.dev)

Confusion Matrix

confusionMatrix function from caret package can be used for creating confusion matrix based on actual response variable and predicted value.

Load Library or packages

```
#if(!require(e1071)) install.packages("e1071")
#if(!require(caret)) install.packages("caret")
library(e1071)

##
## Attaching package: 'e1071'

## The following object is masked from 'package:randomForestSRC':
##
##     impute

library(caret)

## Loading required package: lattice
```

Create Confusion Matrix

```
confusionMatrix(data=cross.sell.dev$predicted.response,
                reference=cross.sell.dev$RESP)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    NO   YES
##        NO  19604  1188
##        YES   235  1473
##
##                Accuracy : 0.9368
##                  95% CI : (0.9335, 0.9399)
##     No Information Rate : 0.8817
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6411
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9882
##             Specificity : 0.5536
##          Pos Pred Value : 0.9429
##          Neg Pred Value : 0.8624
##              Prevalence : 0.8817
##          Detection Rate : 0.8713
##    Detection Prevalence : 0.9241
##       Balanced Accuracy : 0.7709
##
##        'Positive' Class : NO
```

It has accuracy of 99.81%, which is fantastic. Now we can predict response for the validation sample and calculate model accuracy for the sample.

Predicting response variable

```
cross.sell.val$predicted.response <- predict(cross.sell.rf ,cross.sell.val)
```

Create Confusion Matrix

```
confusionMatrix(data=cross.sell.val$predicted.response,
                reference=cross.sell.val$RESP)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    NO   YES
##        NO  19820  1156
##        YES   264  1472
##
##                Accuracy : 0.9375
##                  95% CI : (0.9343, 0.9406)
```

```
##     No Information Rate : 0.8843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6416
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9869
##             Specificity : 0.5601
##          Pos Pred Value : 0.9449
##          Neg Pred Value : 0.8479
##              Prevalence : 0.8843
##          Detection Rate : 0.8727
##    Detection Prevalence : 0.9236
##       Balanced Accuracy : 0.7735
##
##        'Positive' Class : NO
```

Accuracy level has dropped to 91.4% but still significantly higher.

##Fit a Random Forest to the fgl data and compare with SVM

The fgl data frame has 214 rows and 10 columns. It was collected by B. German on fragments of glass collected in forensic work.

```
#if(!require(MASS)) install.packages("MASS")
library(MASS)
data(fgl)
set.seed(17)
fgl.rf <- randomForest(type ~ ., data = fgl, mtry = 2, importance = TRUE,
do.trace = 100)
```

```
## ntree      OOB      1      2      3      4      5      6
##   100:  20.09% 14.29% 18.42% 58.82% 23.08% 22.22% 13.79%
##   200:  21.03% 10.00% 22.37% 64.71% 23.08% 33.33% 13.79%
##   300:  18.69% 10.00% 18.42% 64.71% 23.08% 11.11% 13.79%
##   400:  19.16% 11.43% 17.11% 64.71% 23.08% 22.22% 13.79%
##   500:  19.63% 11.43% 19.74% 58.82% 23.08% 22.22% 13.79%
```

```
print(fgl.rf)
```

```
##
## Call:
##  randomForest(formula = type ~ ., data = fgl, mtry = 2, importance = TRUE,
do.trace = 100)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 19.63%
## Confusion matrix:
##       WinF WinNF Veh Con Tabl Head class.error
```

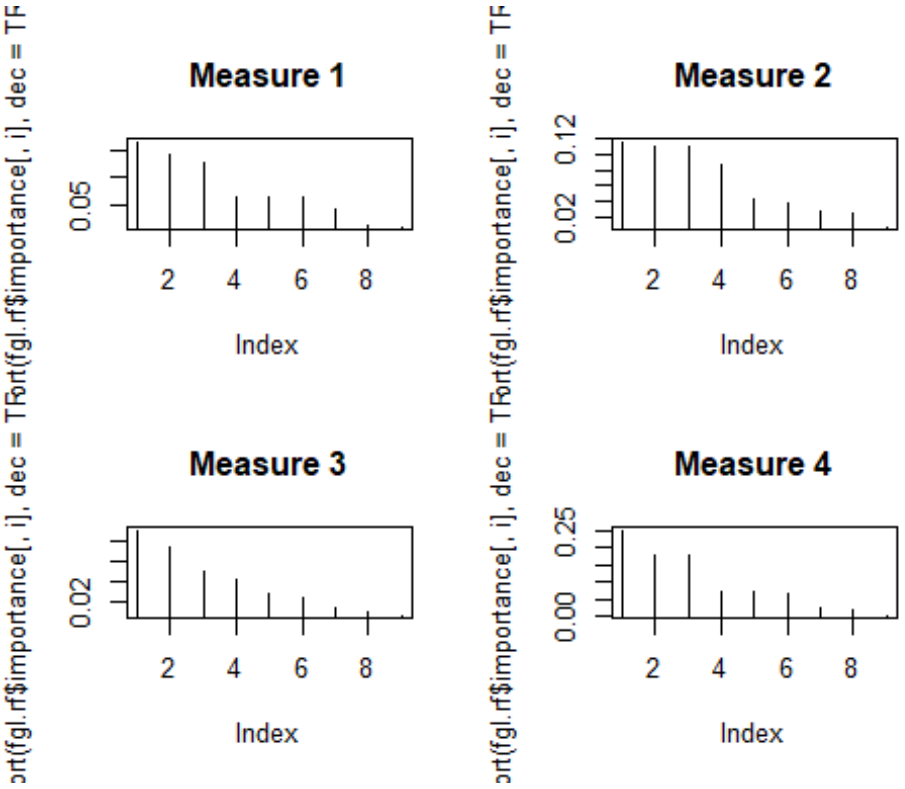
```
## WinF    62     6   2   0    0    0   0.1142857
## WinNF   11    61   1   1    1    1   0.1973684
## Veh      7     3   7   0    0    0   0.5882353
## Con      0     2   0  10    0    1   0.2307692
## Tabl     0     2   0   0    7    0   0.2222222
## Head     1     3   0   0    0   25   0.1379310
```

We can compare random forests with support vector machines by doing ten repetitions of 10-fold cross-validation, using the errorest functions in the ipred package. #errorest performs resampling based estimates of prediction error: misclassification error, root mean squared error or Brier score for survival data.

```
#if(!require(ipred)) install.packages("ipred")
library(ipred)
set.seed(131)
error.RF <- numeric(10)
for(i in 1:10) error.RF[i] <- errorest(type ~ ., data = fgl, model =
randomForest, mtry = 2)$error
summary(error.RF)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1822  0.1928  0.2009  0.2033  0.2173  0.2243

set.seed(563)
error.SVM <- numeric(10)
for (i in 1:10) error.SVM[i] <- errorest(type ~ ., data = fgl, model = svm,
cost = 10, gamma = 1.5)$error
summary(error.SVM)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3645  0.3785  0.3808  0.3794  0.3832  0.3879
```

We see that the random forest compares quite favorably with SVM. We have found that the variable importance measures produced by random forests can sometimes be useful for model reduction

```
par(mfrow = c(2, 2))
for (i in 1:4) plot(sort(fgl.rf$importance[,i], dec = TRUE), type = "h", main
= paste("Measure", i))
```



Random Forest in R example with Low Birth Weight

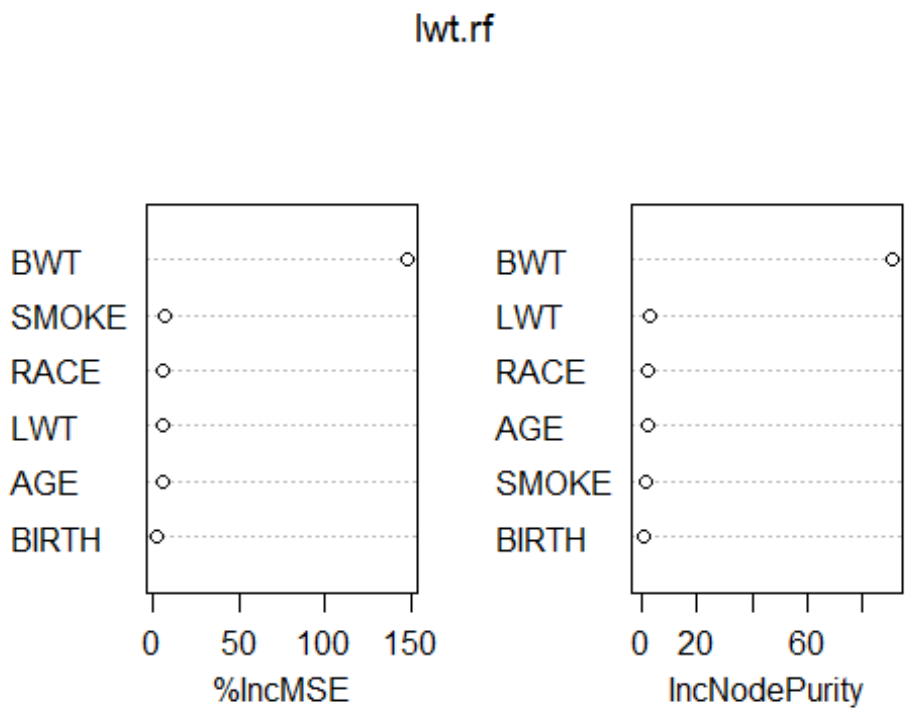
```
lwt<-
read.csv(file="C:/Users/jeff/Documents/VIT_Course_Material/Data_Analytics_2018/data/lowbwt.csv",header = T)
lwt.rf<- randomForest(LOW ~ BIRTH+AGE+RACE+LWT+SMOKE+BWT, data = lwt, mtry = 2, importance = TRUE, do.trace = 100)

## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?

##      |      Out-of-bag   |
## Tree |      MSE  %Var(y) |
##  100 | 0.002026     0.95 |
##  200 | 0.002281     1.07 |
##  300 | 0.001968     0.92 |
##  400 | 0.001734     0.81 |
##  500 | 0.001737     0.81 |

print(lwt.rf)

##
## Call:
##  randomForest(formula = LOW ~ BIRTH + AGE + RACE + LWT + SMOKE +      BWT,
data = lwt, mtry = 2, importance = TRUE, do.trace = 100)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
```

```
##
##           Mean of squared residuals: 0.001736956
##                     % Var explained: 99.19
```

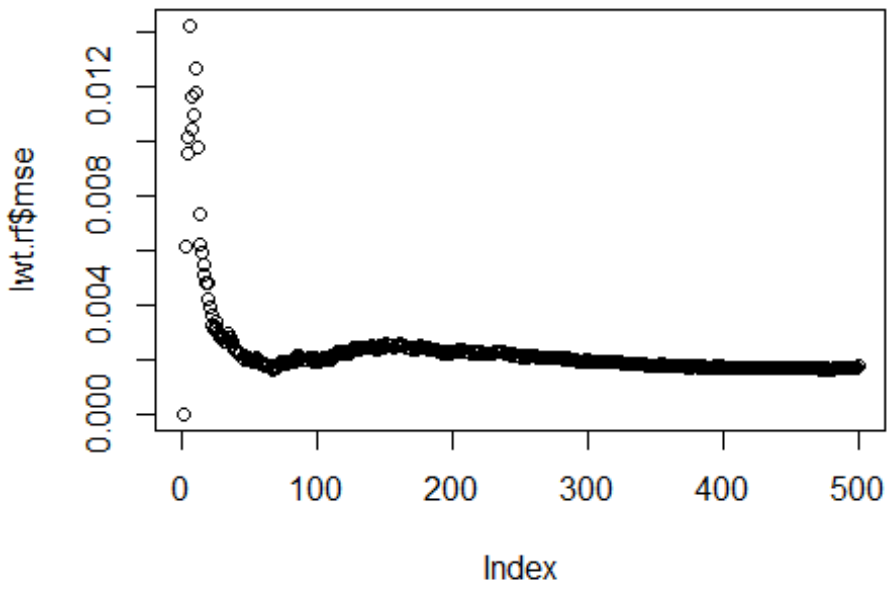
```
lwt.rf$importance
```

```
##            %IncMSE IncNodePurity
## BIRTH 0.0009670033     0.8453432
## AGE   0.0025629431     2.2702510
## RACE  0.0028287052     2.3008177
## LWT   0.0028878607     3.1086324
## SMOKE 0.0037506413     1.5727295
## BWT   0.3919234617    91.4027349
```

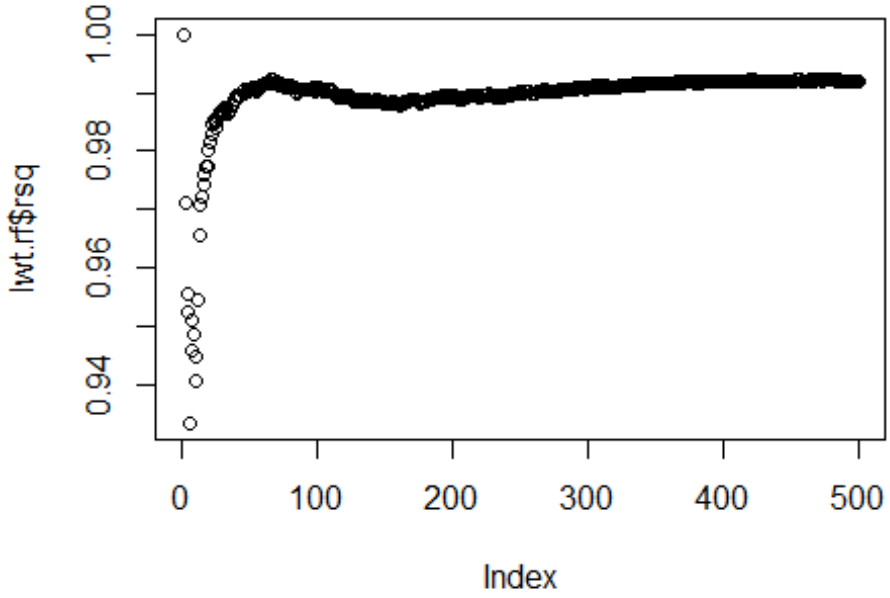
```
varImpPlot(lwt.rf)
```




```
plot(lwt.rf$mse)
```




```r
plot(lwt.rf$rsq)
```



```
set.seed(131)
error.RF <- numeric(10)
for(i in 1:10) error.RF[i] <- errorest(LOW ~ BIRTH+AGE+RACE+LWT+SMOKE+BWT,
data = lwt, model = randomForest, mtry = 2)$error

summary(error.RF)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.04250 0.04310 0.04383 0.04385 0.04473 0.04507

set.seed(563)
error.SVM <- numeric(10)
for (i in 1:10) error.SVM[i] <- errorest(LOW ~ BIRTH+AGE+RACE+LWT+SMOKE+BWT,
data = lwt, model = svm, cost = 10, gamma = 1.5)$error
summary(error.SVM)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2708  0.2732  0.2757  0.2752  0.2761  0.2798
```

Random Forest in R example with IRIS Data

Split iris data to Training data and testing data

```
ind <- sample(2,nrow(iris),replace=TRUE,prob=c(0.7,0.3))
trainData <- iris[ind==1,]
testData <- iris[ind==2,]
```

Generate Random Forest learning treee

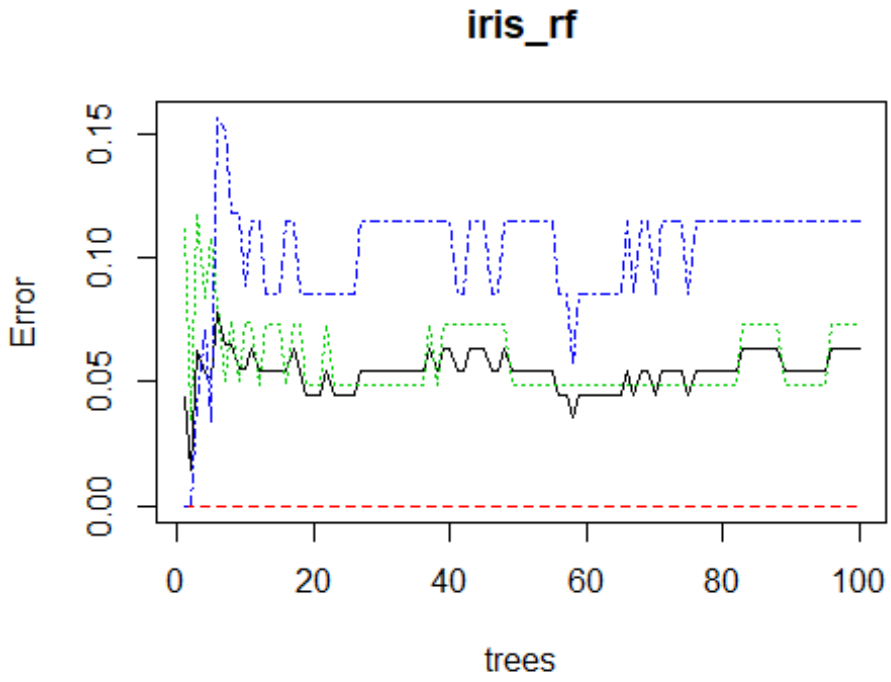
```
iris_rf <- randomForest(Species~.,data=trainData,ntree=100,proximity=TRUE)
table(predict(iris_rf),trainData$Species)

##
##              setosa versicolor virginica
##   setosa         35          0         0
##   versicolor      0         38         4
##   virginica       0          3        31
```

Try to print Random Forest model and see the importance features

```
print(iris_rf)

##
## Call:
##  randomForest(formula = Species ~ ., data = trainData, ntree = 100,
proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 6.31%
## Confusion matrix:
##            setosa versicolor virginica class.error
```

```
## setosa          35          0          0  0.00000000
## versicolor       0         38          3  0.07317073
## virginica        0          4         31  0.11428571
```

```
plot(iris_rf)
```

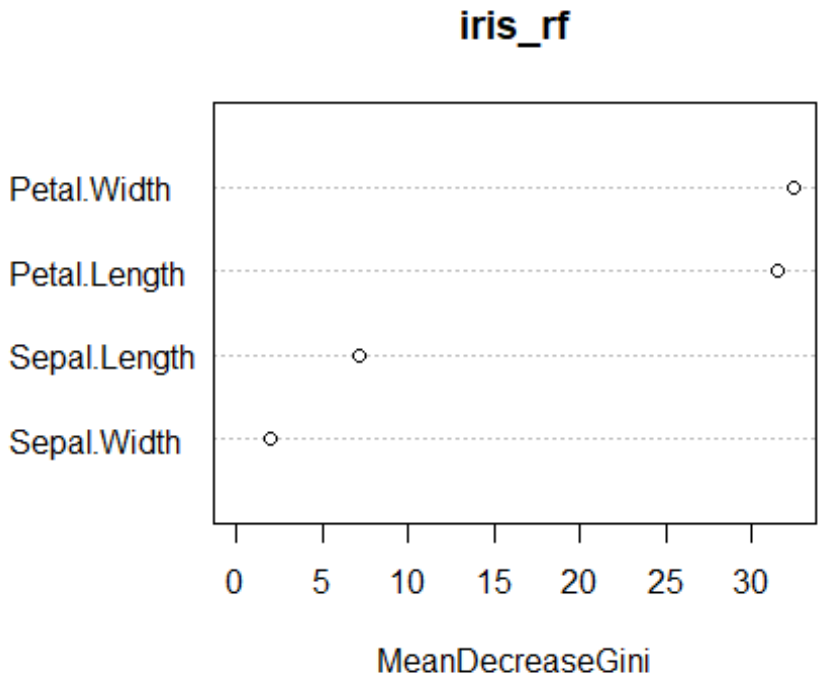


Plot of chunk unnamed-chunk-4

```
importance(iris_rf)
```

```
##              MeanDecreaseGini
## Sepal.Length         7.155701
## Sepal.Width          1.906201
## Petal.Length        31.476540
## Petal.Width         32.464396
```

```
varImpPlot(iris_rf)
```



Plot of chunk unnamed-chunk-4

Try to build random forest for testing data

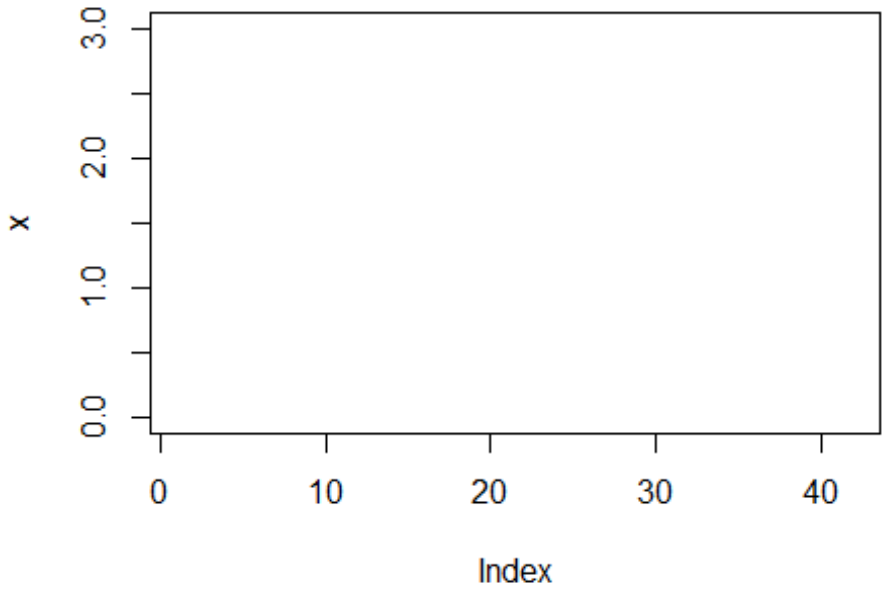
```
irisPred<-predict(iris_rf,newdata=testData)
table(irisPred, testData$Species)
```

```
##
## irisPred     setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0          9         2
##   virginica       0          0        13
```

Try to see the margin, positive or negative, if positif it means correct classification

```
plot(margin(testData$Species))
```

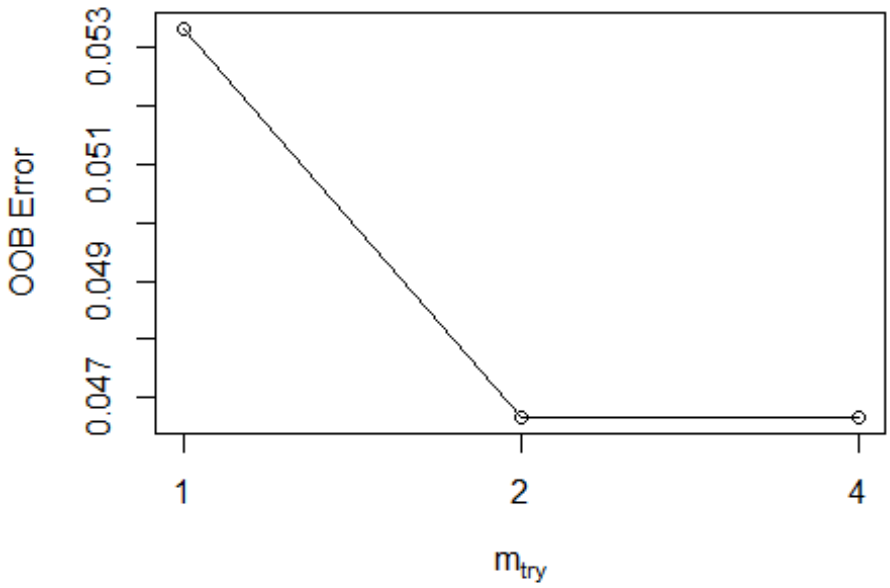
```
## Warning in RColorBrewer::brewer.pal(nlevs, "Set1"): minimal value for n is
3, returning requested palette with 3 different levels
```



Plot of chunk unnamed-chunk-6

Try to tune Random Forest

```
tune.rf <- tuneRF(iris[,-5],iris[,5], stepFactor=0.5)

## mtry = 2  OOB error = 4.67%
## Searching left ...
## mtry = 4     OOB error = 4.67%
## 0 0.05
## Searching right ...
## mtry = 1     OOB error = 5.33%
## -0.1428571 0.05
```



Plot of chunk unnamed-chunk-7

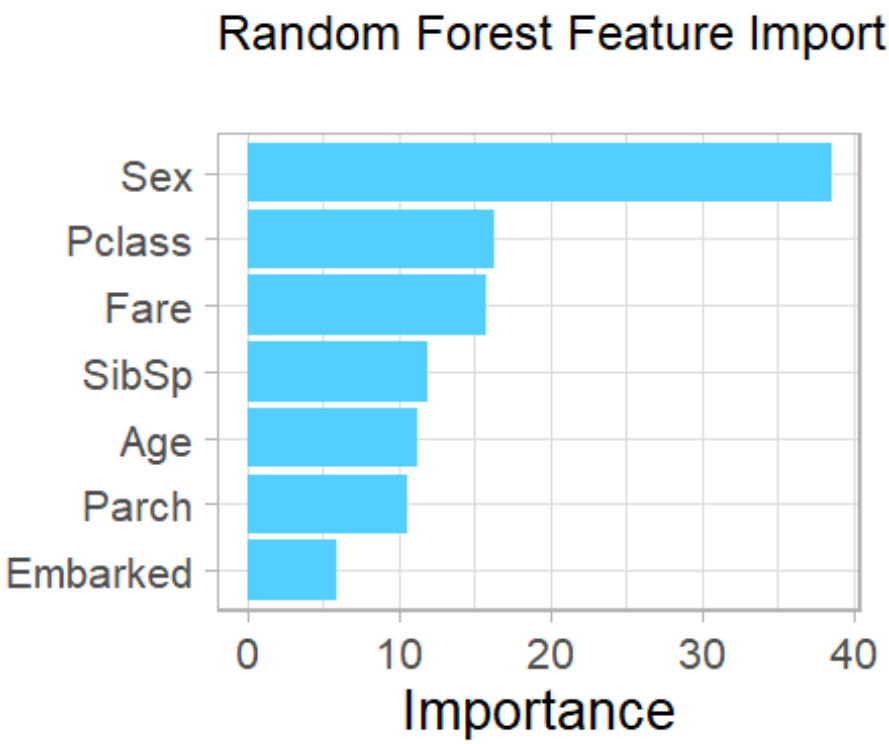
```
print(tune.rf)

##       mtry   OOBError
## 1.OOB    1 0.05333333
## 2.OOB    2 0.04666667
## 4.OOB    4 0.04666667
```

This script trains a Random Forest model based on the data, saves a sample submission, and plots the relative importance of the variables in making predictions

Download 1_random_forest_r_submission.csv through https://www.kaggle.com/c/titanic-gettingStarted/submissions/attach

```
set.seed(1)
train <-
read.csv("C:/Users/jeff/Documents/VIT_Course_Material/Data_Analytics_2018/data/titanic_train.csv", stringsAsFactors=FALSE)
test  <-
read.csv("C:/Users/jeff/Documents/VIT_Course_Material/Data_Analytics_2018/data/titanic_test.csv",  stringsAsFactors=FALSE)

extractFeatures <- function(data) {
  features <- c("Pclass",
                "Age",
                "Sex",
                "Parch",
```

```
                "SibSp",
                "Fare",
                "Embarked")
  fea <- data[,features]
  fea$Age[is.na(fea$Age)] <- -1
  fea$Fare[is.na(fea$Fare)] <- median(fea$Fare, na.rm=TRUE)
  fea$Embarked[fea$Embarked==""] = "S"
  fea$Sex      <- as.factor(fea$Sex)
  fea$Embarked <- as.factor(fea$Embarked)
  return(fea)
}

rf <- randomForest(extractFeatures(train), as.factor(train$Survived),
ntree=100, importance=TRUE)

submission <- data.frame(PassengerId = test$PassengerId)
submission$Survived <- predict(rf, extractFeatures(test))

imp <- importance(rf, type=1)
featureImportance <- data.frame(Feature=row.names(imp), Importance=imp[,1])
p <- ggplot(featureImportance, aes(x=reorder(Feature, Importance),
y=Importance)) +
     geom_bar(stat="identity", fill="#53cfff") +
     coord_flip() +
     theme_light(base_size=20) +
     xlab("") +
     ylab("Importance") +
     ggtitle("Random Forest Feature Importance\n") +
     theme(plot.title=element_text(size=18))
p
```


Random Forest Feature Import
Sex
Pclass
Fare
SibSp
Age
Parch
Embarked
0
10
20
30
40
Importance