Survival Ensembles

Torsten Hothorn\textsuperscript{1,*}, Peter Bühlmann\textsuperscript{2}, Sandrine Dudoit\textsuperscript{3}, Annette Molinaro\textsuperscript{4} and Mark J. van der Laan\textsuperscript{3}

\textsuperscript{1}Institut für Statistik
Ludwig-Maximilians-Universität München
Ludwigstraße 33, D-80539 München, Germany
Tel: ++49–9131–8522707
Fax: ++49–9131–8525740
Torsten.Hothorn@R-project.org

\textsuperscript{2}Seminar für Statistik, ETH Zürich, CH-8032 Zürich, Switzerland
buhlmann@stat.math.ethz.ch

\textsuperscript{3}Division of Biostatistics, University of California, Berkeley
140 Earl Warren Hall, #7360, Berkeley, CA 94720-7360, USA
sandrine@stat.Berkeley.EDU
laan@stat.Berkeley.EDU

\textsuperscript{4}Division of Biostatistics, Epidemiology and Public Health
Yale University School of Medicine, 206 LEPH
60 College Street PO Box 208034, New Haven CT 06520-8034
annette.molinaro@yale.edu

1 Illustrations and Applications

This document reproduces the data analyses presented in Hothorn et al. (2006). For a description of the theory behind applications shown here we refer to the original manuscript. The results differ slightly due to technical changes or bug-fixes in \texttt{mboost} that have been implemented after the paper was printed.

1.1 Acute myeloid leukemia

\textbf{Data preprocessing} Compute IPC weights, define risk score and set up learning sample:

R> ### compute IPC weights
R> AMLw <- IPCweights(Surv(clinical$\text{time}, clinical$\text{event}))
R> ### risk score
R> risk <- rep(0, nrow(clinical))
R> rlev <- levels(clinical[, "Cytogenetic.group"])
R> risk[clinical[, "Cytogenetic.group"] %in% rlev[c(7,8,4)]] <- "low"
R> risk[clinical[, "Cytogenetic.group"] %in% rlev[c(5, 9)]] <- "intermediate"
R> risk[clinical[, "Cytogenetic.group"] %in% rlev[-c(4,5, 7,8,9)]] <- "high"
R> risk <- as.factor(risk)
R> ### set-up learning sample
R> AMLlearn <- cbind(clinical[, c("time", "Sex", "Age", "LDH", "WBC", 
"FLT3.aberration.", "MLL.PTD", "Tx.Group.")],
risk = risk,
iexpressions[, colnames(iexpressions) %in% selgenes[["Clone.ID"]]])
R> cc <- complete.cases(AMLlearn)
R> AMLlearn <- AMLlearn[AMLw > 0 & cc,]
R> AMLw <- AMLw[AMLw > 0 & cc]

Model fitting  Fit random forest for censored data
R> ### controls for tree growing
R> ctrl <- ctree_control(testtype = "Teststatistic",
  teststat = "maximum", mincriterion = .1, minsplit = 5)
R> ### was: cforest_control(mincriterion = 0.1, mtry = 5, minsplit = 5, ntree = 250)
R>
R> ### fit random forest for censored data (warnings are OK here)
R> AMLrf <- cforest(log(time) ~ ., data = AMLlearn, control = ctrl,
  weights = AMLw, mtry = 5, ntree = 250,
  perturb = list(replace = TRUE, fraction = 0.632))

and L₂ Boosting for censored data
R> AMLl2b <- glmboost(I(log(time)) ~ ., data = AMLlearn, weights = AMLw,
  control = boost_control(mstop = 5000))

Compute fitted values
R> ### restrict number of boosting iterations and inspect selected variables
R> AMLl2b <- AMLl2b[mstop(aic)]
R> cAML <- coef(AMLl2b)
R> cAML[abs(cAML) > 0]
R> ### AIC criterion
R> plot(aic <- AIC(AML12b))

Figure 1: AIC criterion for AML data.
1.2 Node-positive breast cancer

Data preprocessing  Compute IPC weights and set up learning sample:

```r
R> ### attach data
R> data("GBSG2", package = "TH.data")
R> ### IPC weights
R> GBSG2w <- IPCweights(Surv(GBSG2$time, GBSG2$cens))
R> ### set-up learning sample
R> GBSG2learn <- cbind(GBSG2[, -which(names(GBSG2) %in% c("time", "cens")]),
                     ltime = log(GBSG2$time))
R> n <- nrow(GBSG2learn)
```

Model fitting

```r
R> ### linear model
R> LMmod <- lm(ltime ~ . , data = GBSG2learn, weights = GBSG2w)
R> LMerisk <- sum((GBSG2learn$ltime - predict(LMmod))^2*GBSG2w) / n
R> ### regression tree
R> pos <- GBSG2learn$GBSG2w > 0
```
Figure 2: AML data: Reproduction of Figure 1.
R> TRmod <- rpart(ltime ~ ., data = GBSG2learn, weights = GBSG2w, subset = pos)
R> TRerisk <- sum((GBSG2learn$ltime[pos] - predict(TRmod))^2*GBSG2w[pos]) / n
R> ### tree controls
R> ctrl <- ctree_control(testtype = "Teststatistic",
                       teststat = "maximum", mincriterion = qnorm(.95),
                       minsplit = 5)
R> ### was: cforest_control(mincriterion = qnorm(0.95), mtry = 5,
R> ### minsplit = 5, ntree = 100)
R>
R> ### fit random forest for censored data (warnings are OK here)
R> RFmod <- cforest(ltime ~ ., data = GBSG2learn, weights = GBSG2w,
                   control = ctrl, mtry = 5, ntree = 100,
                   perturb = list(replace = TRUE,
                                  fraction = 0.632 * sum(GBSG2w > 0)))
R> ### fit L2 boosting for censored data
R> L2Bmod <- glmboost(ltime ~ ., data = GBSG2learn, weights = GBSG2w,
                     control = boost_control(mstop = 250))
R> ### with Huber loss function
R> L2BHubermod <- glmboost(ltime ~ ., data = GBSG2learn, weights = GBSG2w,
                          family = Huber(d = log(2)))

Compute fitted values:

R> GBSG2Hp <- predict(L2BHubermod, newdata = GBSG2learn)
R> L2Berisk <- sum((GBSG2learn$ltime - predict(L2Bmod, newdata = GBSG2learn))^2*GBSG2w) / n
R> RFerisk <- sum((GBSG2learn$ltime - predict(RFmod, newdata = GBSG2learn))^2*GBSG2w) / n
R> plot(aic <- AIC(L2Bmod))

Figure 3: AIC criterion for GBSG2 data.
Figure 4: GBSG-2 data: Reproduction of Figure 3.
Figure 5: GBSG-2 data: Reproduction of Figure 5.
Figure 6: GBSG-2 data: Reproduction of Figure 6.
Figure 7: GBSG-2 data: Reproduction of Figure 7.
References