PHP2650 Final Project Code Appendix

Breanna Richards & Nancy Liu

2022-05-15

Predicting Survival and Controlling for Bias with Random Survival Forests and Conditional Inference Forests

PBC Data

Variable Selection

```r
pbc_use <- pbc %>%
  mutate(status = as.integer(ifelse(status == 2, 1, 0))) %>%
  select(-id)


pbc_use$trt <- as.factor(pbc_use$trt)

pbc_use$ascites <- as.factor(pbc_use$ascites)

pbc_use$hepato <- as.factor(pbc_use$hepato)

pbc_use$spiders <- as.factor(pbc_use$spiders)

pbc_use$copper <- as.numeric(pbc_use$copper)

pbc_use$stage <- as.numeric(pbc_use$stage)

pbc_use$platelet <- as.numeric(pbc_use$platelet)

pbc_use$trig <- as.numeric(pbc_use$trig)

pbc_use$time <- as.numeric(pbc_use$time)

pbc_use$chol <- as.numeric(pbc_use$chol)
```

```r
# test main effects
pbc.main <- coxph(Surv(time, status) ~ ., data =  pbc_use)
step(pbc.main, direction = "backward")
```

```r
pbc.cox2 <- coxph(Surv(time, status) ~ . - alk.phos, data = pbc_use)
pbc.cox3 <- coxph(Surv(time, status) ~ . - alk.phos -
                    hepato, data = pbc_use)
```

```r
pbc.cox4 <- coxph(Surv(time, status) ~ . - alk.phos -
                    hepato - ascites, data = pbc_use)
pbc.cox5 <- coxph(Surv(time, status) ~ . - alk.phos -
                    hepato - ascites - spiders, data = pbc_use)
pbc.cox6 <- coxph(Surv(time, status) ~ . - alk.phos -
                    hepato - ascites - spiders -
                    trt, data = pbc_use)
pbc.cox7 <- coxph(Surv(time, status) ~ . - alk.phos -
                    hepato - ascites - spiders -
                    trt - trig, data = pbc_use)
pbc.cox8 <- coxph(Surv(time, status) ~ . - alk.phos -
                    hepato - ascites - spiders -
                    trt - trig - platelet, data = pbc_use)
pbc.cox9 <- coxph(Surv(time, status) ~ . - alk.phos -
                    hepato - ascites - spiders -
                    trt - trig - platelet - sex, data = pbc_use)
pbc.cox10 <- coxph(Surv(time, status) ~ . - alk.phos -
                    hepato - ascites - spiders -
                    trt - trig - platelet - sex -
                     chol, data = pbc_use)
pbc_use <- pbc_use %>%
  select(-c(alk.phos, hepato, ascites, spiders,
                    trt, trig, platelet, sex, chol))
```

```r
# test two-way interactions
pbc.full <- coxph(Surv(time, status) ~ (.)^2, data =  pbc_use)
#step(pbc.full, direction = "backward")
```

```r
aareg_pbc <- aareg(Surv(time, status) ~ age + edema + bili + albumin +
  copper + ast + protime + stage + age:edema + age:copper +
  bili:ast,
  data = pbc_use)

pbc_use <- pbc_use %>%
  filter(time %in% aareg_pbc$times)


pbc_use <- imbalance::oversample(na.omit(pbc_use),
           classAttr = "status",
           ratio = 1) %>%
  mutate(status = as.integer(as.character(status)))
```

```r
pbc_cox <- coxph(Surv(time, status) ~ age + edema + bili + albumin +
  copper + ast + protime + stage + age:edema + age:copper +
  bili:ast,
  data = pbc_use)
```

```r
test.ph <- cox.zph(pbc_cox)
test.ph
```

```r
schoenplot1 <- ggcoxzph(test.ph)[1]
schoenplot2 <- ggcoxzph(test.ph)[3]
schoenplot3 <- ggcoxzph(test.ph)[7]
schoenplot4 <- ggcoxzph(test.ph)[8]
schoenplot5 <- ggcoxzph(test.ph)[11]

schoenplot1$`1`
schoenplot2$`3`
schoenplot3$`7`
schoenplot4$`8`
schoenplot5$`11`
```

Random Survival Forest Implementation

```r
set.seed(1)
# We can find the optimal mtry and nodesize using OOB
tuning <- tune(formula = Surv(time, status) ~ age + edema + bili + albumin + copper +
       ast + protime + stage + age:edema + age:copper + bili:ast,
     data = pbc_use,
     mtryStart = ncol(pbc_use) / 2,
     nodesizeTry = c(c(1:9), seq(10, 100, by = 5)),
     ntreeTry =500,
     doBest = TRUE)

res <- as.data.frame(tuning$results)
# print optimal nodesize and mtry
tuning$optimal
node_size <- c(seq(5, 100, by = 5))
```

```r
ggplot(res) +
  geom_line(aes(x = nodesize, y = err, colour = factor(mtry))) +
  labs(col = "mtry")
```

```r
set.seed(1)
# random forest
pbc_rf <- rfsrc(Surv(time, status) ~ age + edema + bili + albumin +
  copper + ast + protime + stage + age:edema + age:copper +
  bili:ast,
  mtry = 1,
  nodesize = 5,
  data = pbc_use)
```

Conditional Inference Forest Implementation

```r
set.seed(1)
# run conditional inference forest

pbc_cif <- pecCforest(Surv(time, status) ~ age + edema + bili + albumin +
  copper + ast + protime + stage + age:edema + age:copper +
  bili:ast,
  data = pbc_use)
```

Comparing the Models

```r
set.seed(1)
rand_inds_pbc_event <- sample(nrow(pbc_use[pbc_use$status == 1,]), 2)
rand_inds_pbc_cens <- sample(nrow(pbc_use[pbc_use$status == 0,]), 2)
rand_data_pbc <- rbind(pbc_use[pbc_use$status == 1,][rand_inds_pbc_event, ],
                       pbc_use[pbc_use$status == 0,][rand_inds_pbc_cens, ])

cif.predict3_pbc <- treeresponse(pbc_cif$forest,
                                 newdata = rand_data_pbc)
```

```r
set.seed(1)
# Obtain predicted values for RSF
pbc_pred <- predict(pbc_rf, rand_data_pbc)

# obtain time values
Time <- pbc_rf$time.interest
```

```r
cif.predict3_pbc$`69`
cif.predict3_pbc$`40`
cif.predict3_pbc$`14`
cif.predict3_pbc$`153`
```

```r
##########
# Finding the Median Survival Probability
# obtain the index for which the survival probability = 0.5
t_id1 <- which.min(abs(round(pbc_pred$survival[1,], 2) -.5))
t_id2 <- which.min(abs(round(pbc_pred$survival[2,], 2) - .5))
t_id3 <- which.min(abs(round(pbc_pred$survival[3,], 2) - .5))
t_id4 <- which.min(abs(round(pbc_pred$survival[4,], 2) - .5))
# obtain the times at which survival probability = 0.5 occurs
Time[t_id1] # 1827
Time[t_id2] # 1487
Time[t_id3] # 3584
Time[t_id4] # 3584
```

```r
par(mfrow = c(2,2))
# Compare the two methods by plotting
plot(cif.predict3_pbc$`69`, col = "blue",
     main = "PBC Pred. Survival Individual 69",
     sub = "Event Experienced",
     xlab = "time",
     ylab = "Survival Probability")
lines(Time, pbc_pred$survival[1,], col = "green")
legend(4000, -0.46,           # Position
       legend = c("CIF", "RSF"),  # Legend texts
       title = "Method",
       col = c("blue", "green"),
       lwd = 2,
       xpd = "NA",
       cex = 0.8)

plot(cif.predict3_pbc$`40`, col = "blue",
     main = "PBC Pred. Survival Individual 40",
     sub = "Event Experienced",
     xlab = "time",
     ylab = "Survival Probability")
lines(Time, pbc_pred$survival[2,], col = "green")


plot(cif.predict3_pbc$`14`, col = "blue",
     main = "PBC Pred. Survival Individual 140",
     sub = "Censored",
     xlab = "time",
     ylab = "Survival Probability")
lines(Time, pbc_pred$survival[3,], col = "green")


plot(cif.predict3_pbc$`153`, col = "blue",
     main = "PBC Pred. Survival Individual 153",
     sub = "Censored",
     xlab = "time",
     ylab = "Survival Probability")
lines(Time, pbc_pred$survival[3,], col = "green")
```

Variable Importance

```
## VAR IMP CIF ##
pbc_cif_var <- party::cforest(Surv(time, status) ~ age + edema + bili + albumin +
  copper + ast + protime + stage + age:edema + age:copper +
  bili:ast,
  data = pbc_use)


pbc.var_imp_cif <- data.frame(importance = varImp(pbc_cif_var)) %>%
  arrange(desc(Overall))


pbc.var_imp_cif$vars <- rownames(pbc.var_imp_cif)

ggplot(pbc.var_imp_cif, aes(x = reorder(vars, -Overall, decreasing = TRUE),
                            y = Overall,
                            fill = vars)) +
  geom_col() +
  coord_flip() +
  labs(x = "variables",
       y = "CIF Variable Importance") + theme_bw() +
  theme(legend.position = "none")
```

```
## VAR IMP RSF ##
rf_imp_pbc <- data.frame(importance = vimp(pbc_rf)$importance %>% sort(decreasing = T))
rf_imp_pbc$variables <- rownames(rf_imp_pbc)

ggplot(rf_imp_pbc, aes(x = reorder(variables, -importance, decreasing = TRUE),
                            y = importance,
                            fill = variables)) +
```

```r
  geom_col() +
  coord_flip() +
  labs(x = "variables",
       y = "RSF Variable Importance") + theme_bw() +
  theme(legend.position = "none")

#plot.survival.rfsrc(pbc_rf)
```

Prediction Error Curves

```r
set.seed(1)
fit_comp_pbc <- pec(list("rsf" = pbc_rf, "cforest" = pbc_cif),
                    data = pbc_use,
                    formula = Surv(time, status) ~ age + edema +
                      bili + albumin + copper + ast + protime + stage +
                      age:edema + age:copper + bili:ast,
                    splitMethod = "BootCv",
                    reference = FALSE,
                    maxtime = 3445,
                    B = 50,
                    exact = TRUE
              )
```

```r
crps(fit_comp_pbc)


plot(fit_comp_pbc, predErr = "BootCvErr")
```

Employee Turnover Data

```r
turnover <- read_csv("data/turnover.csv")
```

Variable Selection

```r
turn_use <- turnover
turn_use$event = as.integer(turn_use$event)
turn_use$gender = as.factor(turn_use$gender)
turn_use$industry = as.factor(turn_use$industry)
turn_use$profession = as.factor(turn_use$profession)
turn_use$traffic = as.factor(turn_use$traffic)
turn_use$coach = as.factor(turn_use$coach)
turn_use$head_gender = as.factor(turn_use$head_gender)
turn_use$greywage = as.factor(turn_use$greywage)
turn_use$way = as.factor(turn_use$way)
```

```r
# test main effects
turn.main <- coxph(Surv(stag, event) ~ ., data =  turn_use)
step(turn.main, direction = "backward")
```

```r
turn_use <- turn_use %>%
  select(c(stag, event, age, industry, profession, traffic, greywage, way, selfcontrol, anxiety))
```

```r
# test two-way interactions
turn.full <- coxph(Surv(stag, event) ~ (.)^2, data =  turn_use)
step(turn.full, direction = "backward")
```

```r
turn_cox <- coxph(Surv(stag, event) ~ age + industry + profession + traffic +
        greywage + way + selfcontrol + anxiety + age:way + way:selfcontrol,
       data =  turn_use)
```

```r
test.ph_turn <- cox.zph(turn_cox)
test.ph_turn
```

```r
schoenplot1_turn <- ggcoxzph(test.ph_turn)[3]

schoenplot1_turn$`3`
```

Random Survival Forest Implementation

```r
turn_use$age.way <- interaction(turn_use$age, turn_use$way)
turn_use$way.selfcontrol <- interaction(turn_use$way,
                                        turn_use$selfcontrol)
```

```r
set.seed(1)
# random survival forest
turn_rsf <- ranger(Surv(stag, event) ~ age + industry + profession + traffic +
        greywage + way + selfcontrol + anxiety +
          age.way + way.selfcontrol,
        mtry = 2,
       data =  turn_use)
```

Conditional Inference Forest Implementation

```r
set.seed(1)
turn_cif <- pecCforest(Surv(stag, event) ~ age + industry + profession + traffic +
        greywage + way + selfcontrol + anxiety +
          age.way + way.selfcontrol,
       data =  turn_use)
```

Comparing the Models

```r
set.seed(1)
rand_inds_turn_event <- sample(nrow(turn_use[turn_use$event == 1,]), 2)
rand_inds_turn_cens <- sample(nrow(turn_use[turn_use$event == 0,]), 2)
rand_data_turn <- rbind(turn_use[turn_use$event == 1,][rand_inds_turn_event, ],
                       turn_use[turn_use$event == 0,][rand_inds_turn_cens, ])

cif.predict3_turn <- treeresponse(turn_cif$forest,
                                  newdata = rand_data_turn)

turn_rsf_pred <- predict(turn_rsf, rand_data_turn, type='response')
time <- turn_rsf_pred$unique.death.times
```

```
cif.predict3_turn$`1`
cif.predict3_turn$`2`
cif.predict3_turn$`3`
cif.predict3_turn$`4`


#fit <- survfit(Surv(stag, event) ~ age + industry + profession + traffic +
 #       greywage + way + selfcontrol + anxiety,
#        data = turn_use)

# find median survival
t1 <- max(which(abs(turn_rsf_pred$survival[1,] -.5) == min(abs(turn_rsf_pred$survival[1,] -.5))))
t2 <- max(which(abs(turn_rsf_pred$survival[2,] -.5) == min(abs(turn_rsf_pred$survival[2,] -.5))))
t3 <- max(which(abs(turn_rsf_pred$survival[3,] -.5) == min(abs(turn_rsf_pred$survival[3,] -.5))))
t4 <- max(which(abs(turn_rsf_pred$survival[4,] -.5) == min(abs(turn_rsf_pred$survival[4,] -.5))))

time[c(t1, t2, t3, t4)] #  49.34702  74.05339 164.56674 179.44969
```

```
par(mfrow = c(2,2))
# Compare the two methods by plotting
plot(cif.predict3_turn$`1`, col = "blue",
     main = "Turn Pred. Survival Individual 129",
     sub = "Event Experienced",
     xlab = "time",
     ylab = "Survival Probability")
lines(time, turn_rsf_pred$survival[1,], col = "green")
legend(180, -0.46,          # Position
  legend = c("CIF", "RSF"),  # Legend texts
    title = "Method",
     col = c("blue", "green"),
     lwd = 2,
     xpd = "NA",
     cex = 0.8)

plot(cif.predict3_turn$`2`, col = "blue",
     main = "Turn Pred. Survival Individual 509",
     sub = "Event Experienced",
     xlab = "time",
     ylab = "Survival Probability")
lines(time, turn_rsf_pred$survival[2,], col = "green")


plot(cif.predict3_turn$`3`, col = "blue",
     main = "PBC Pred. Survival Individual 471",
     sub = "Censored",
     xlab = "time",
     ylab = "Survival Probability")
lines(time, turn_rsf_pred$survival[3,], col = "green")


plot(cif.predict3_turn$`4`, col = "blue",
     main = "PBC Pred. Survival Individual 299",
     sub = "Censored",
```

```r
     xlab = "time",
     ylab = "Survival Probability")
lines(time, turn_rsf_pred$survival[4,], col = "green")
```

Variable Importance

```r
set.seed(1)
turn_cif_var <- party::cforest(Surv(stag, event) ~ age + industry + profession + traffic +
        greywage + way + selfcontrol + anxiety +
          age.way + way.selfcontrol,
       data =  turn_use)


turn.var_imp_cif <- data.frame(importance = varImp(turn_cif_var)) %>%
  arrange(desc(Overall))


turn.var_imp_cif$vars <- rownames(turn.var_imp_cif)

ggplot(turn.var_imp_cif, aes(x = reorder(vars, -Overall, decreasing = TRUE),
                       y = Overall,
                       fill = vars)) +
  geom_col() +
  coord_flip() +
  labs(x = "variables",
       y = "CIF Variable Importance") + theme_bw() +
  theme(legend.position = "none")
```

```r
turn.rf.imp <- ranger(Surv(stag, event) ~ age + industry + profession + traffic +
        greywage + way + selfcontrol + anxiety +
          age.way + way.selfcontrol,
       data =  turn_use,
       importance = 'permutation')

turn.rf.imp$variable.importance
turn_rf_imp <- data.frame(importance = turn.rf.imp$variable.importance) %>%
  arrange(desc(importance))
turn_rf_imp$vars <- rownames(turn_rf_imp)

ggplot(turn_rf_imp, aes(x = reorder(vars, importance, decreasing = TRUE),
                       y = importance,
                       fill = vars)) +
  geom_col() +
  coord_flip() +
  labs(x = "variables",
       y = "RSF Variable Importance") + theme_bw() +
  theme(legend.position = "none")
```

Prediction Error Curves

```r
predictSurvProb.ranger <- function (object, newdata, times, ...) {
    ptemp <- ranger:::predict.ranger(object, data = newdata, importance = "none")$survival
    pos <- prodlim::sindex(jump.times = object$unique.death.times,
        eval.times = times)
    p <- cbind(1, ptemp)[, pos + 1, drop = FALSE]
    if (NROW(p) != NROW(newdata) || NCOL(p) != length(times))
        stop(paste("\nPrediction matrix has wrong dimensions:\nRequested newdata x times: ",
            NROW(newdata), " x ", length(times), "\nProvided prediction matrix: ",
            NROW(p), " x ", NCOL(p), "\n\n", sep = ""))
    p
}
```

```
form <- as.formula(paste("Surv(stag, event)~",
       paste(turn_rsf$forest$independent.variable.names, collapse="+")))

#?pec::predictSurvProb
fit_comp_turn <- pec(list("rsfc" = turn_rsf, "cforest" = turn_cif),
                     data = turn_use,
              formula= form,
              splitMethod = "BootCv",
        B = 5,
        reference = FALSE
              )
```

```
crps(fit_comp_turn)
```

```
plot(fit_comp_turn, predErr = "BootCvErr")
```