```
opts_chunk$set (warning = FALSE, message = FALSE, tidy=TRUE, echo=TRUE)
options(warn = -1)

rm(list=ls())

library(survival)
library(knitr)
library(kableExtra)

library(glmnet)
```

```
## Loading required package:  Matrix
## Loaded glmnet 4.1-7
```

```
library(ggplot2)

# Following loaded in "forest_search_v0.R"
suppressMessages(library(randomForest))
#library(SPlit)

library(grf)
library(policytree)
library(DiagrammeR)

#library(cowplot)

library(data.table)
library(plyr)
library(aVirtualTwins)
# Not sure formatR is needed?
#library(formatR)
suppressMessages(library(gridExtra))

library(speff2trial)
```

```
## Loading required package:  leaps
```

```
# Location where code is stored
codepath<-c("/Users/larryleon/Documents/GitHub/forestSearch/R/")
source(paste0(codepath,"source_forestsearch_v0.R"))
source_fs_functions(file_loc=codepath)
```

```
t.start.all <- proc.time()[3]
# GRF analysis To guide selection of binary cutpoints
df.analysis <- subset(ACTG175, arms %in% c(2, 3))

df.analysis <- within(df.analysis, {
    id <- as.numeric(c(1:nrow(df.analysis)))
    time_days <- days
    treat <- ifelse(arms == 2, 1, 0)
})

# plot(survfit(Surv(time_days,cens)~treat,data=df.analysis))
coxph(Surv(time_days, cens) ~ treat, data = df.analysis)

## Call:
```

```
## coxph(formula = Surv(time_days, cens) ~ treat, data = df.analysis)
##
##          coef exp(coef) se(coef)      z     p
## treat -0.1102    0.8957   0.1303 -0.845 0.398
##
## Likelihood ratio test=0.72  on 1 df, p=0.3974
## n= 1085, number of events= 237
```

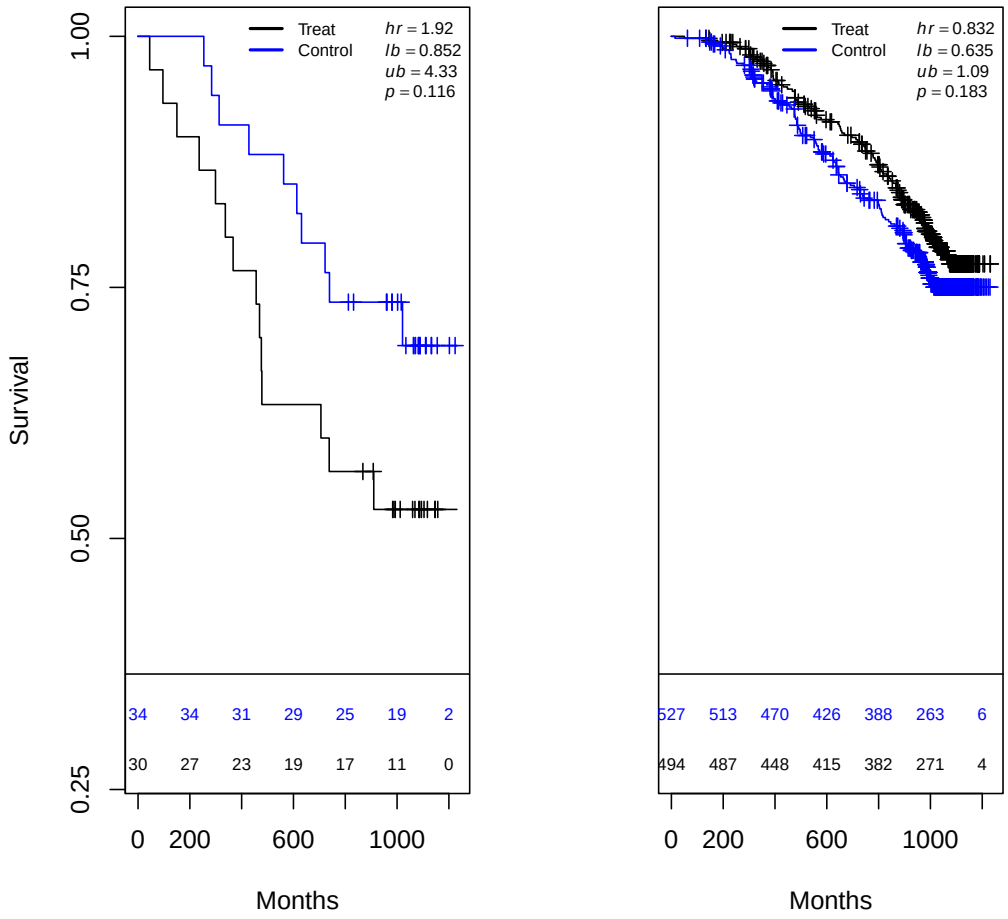
```
confounders.name <- c("age", "wtkg", "karnof", "cd40", "cd80", "hemo", "homo", "drugs",
    "race", "gender", "oprior", "symptom")
outcome.name <- c("time_days")
event.name <- c("cens")
id.name <- c("id")
treat.name <- c("treat")

n.min <- 60
dmin.grf <- 12
frac.tau <- 0.8

# Stored results load('output/grf_actg_Arms_2vs3_final.Rdata')

grf.est <- grf.subg.harm.survival(data = df.analysis, confounders.name = confounders.name,
    outcome.name = outcome.name, event.name = event.name, id.name = id.name, treat.name = treat.name,
    n.min = n.min, dmin.grf = dmin.grf, frac.tau = frac.tau, details = TRUE)
```

```
## tau= 816.8
##    leaf.node control.mean control.size control.se treated.mean treated.size
## 1          2   -22.143720   827.000000   9.309900    22.143720   827.000000
## 2          3    21.953370   258.000000  20.570947   -21.953370   258.000000
## 3          4    18.393717   172.000000  18.499444   -18.393717   172.000000
## 4          5   -81.119958   126.000000  28.085168    81.119958   126.000000
## 5          6   -30.221943   518.000000  11.125830    30.221943   518.000000
## 6          7    37.410687   269.000000  20.541539   -37.410687   269.000000
## 31        10   -39.547450   456.000000  12.958713    39.547450   456.000000
## 41        11     8.160581   181.000000  10.440004    -8.160581   181.000000
## 51        12   -82.106514   112.000000  29.185500    82.106514   112.000000
## 61        13    74.151529   124.000000  25.889428   -74.151529   124.000000
## 7         14   104.261908    64.000000  56.120986  -104.261908    64.000000
## 8         15   -71.655269    81.000000  30.633092    71.655269    81.000000
##     treated.se       diff depth
## 1     9.309900  -44.28744     1
## 2    20.570947   43.90674     1
## 3    18.499444   36.78743     2
## 4    28.085168 -162.23992     2
## 5    11.125830  -60.44389     2
## 6    20.541539   74.82137     2
## 31   12.958713  -79.09490     3
## 41   10.440004   16.32116     3
## 51   29.185500 -164.21303     3
## 61   25.889428  148.30306     3
## 7    56.120986  208.52382     3
## 8    30.633092 -143.31054     3
##   leaf.node control.mean control.size control.se treated.mean treated.size
## 7        14    104.26191     64.00000   56.12099   -104.26191     64.00000
##   treated.se     diff depth
## 7   56.12099 208.5238     3
```

```r
cat("Truncation point for RMST:", c(grf.est$tau.rmst), "\n")

## Truncation point for RMST: 816.8

# Plot manually

# plot(grf.est£tree)

# plot(grf.est£tree1)

# plot(grf.est£tree2)

# plot(grf.est£tree3)

df0.grf <- subset(grf.est$data, treat.recommend == 0)
df1.grf <- subset(grf.est$data, treat.recommend == 1)

# Terminal leaf corresponding to selected SG
cat("Terminal leaf:", c(grf.est$sg.harm.id), "\n")

## Terminal leaf: karnof <= 90

# action=1 --> recommend control

# Manually identify the subgroup looking at tree and terminal leaf
print(dim(df0.grf))

## [1] 64 34

check <- subset(df.analysis, karnof <= 90 & cd80 > 1034 & age > 37)
print(dim(check))

## [1] 64 29

# plot(survfit(Surv(time_days,cens)~treat,data=df.analysis))
# coxph(Surv(time_days,cens)~treat,data=df.analysis)

# save(grf.est,file='output/grf_actg_Arms_2vs3_final.Rdata')
```

```r
cat("GRF variables in selected tree", "\n")

## GRF variables in selected tree

print(grf.est$tree.names)

## [1] "age"    "cd80"   "wtkg"   "cd40"   "karnof"

cat("GRF cuts wrt selected tree:", "\n")

## GRF cuts wrt selected tree:

print(grf.est$tree.cuts)

## [1] "age <= 37"     "cd80 <= 499"   "cd80 <= 1034"  "wtkg <= 64.64"
## [5] "cd40 <= 417"   "cd80 <= 680"   "karnof <= 90"
```

```r
par(mfrow = c(1, 2))
plot.subgroup(sub1 = df0.grf, sub1C = df1.grf, tte.name = "time_days", event.name = "cens",
    treat.name = "treat", fix.rows = FALSE, byrisk = 200, show.med = FALSE, ymin = 0.4)
```




```r
# Reduce dimension via Cox lasso

xx <- as.matrix(df.analysis[, confounders.name])
yy <- as.matrix(df.analysis[, c("time_days", "cens")])
colnames(yy) <- c("time", "status")

cvfit <- cv.glmnet(xx, yy, family = "cox")  #first do 10-fold cross-validation to select lambda

m <- glmnet(xx, yy, family = "cox", lambda = cvfit$lambda.min)  #plugin the optimal lambda

conflasso.name <- confounders.name[which(m$beta != 0)]

cat("Cox-LASSO selected:", c(conflasso.name), "\n")

## Cox-LASSO selected: age wtkg karnof cd40 cd80 drugs oprior symptom
```

```r
cat("GRF cuts wrt selected tree:", "\n")

## GRF cuts wrt selected tree:

print(grf.est$tree.cuts)

## [1] "age <= 37"     "cd80 <= 499"   "cd80 <= 1034"  "wtkg <= 64.64"
## [5] "cd40 <= 417"   "cd80 <= 680"   "karnof <= 90"

# Considering continuous factors per GRF cuts Only considering drugs and
# symptom per lasso
df.analysis <- within(df.analysis, {
    z1a <- ifelse(age <= 37, 1, 0)
    z1b <- ifelse(age <= median(age), 1, 0)
    z2 <- ifelse(wtkg <= 65, 1, 0)
    z3 <- ifelse(karnof <= 90, 1, 0)
    z4 <- ifelse(cd40 <= 417, 1, 0)
    z5a <- ifelse(cd80 <= 499, 1, 0)
    z5b <- ifelse(cd80 <= 680, 1, 0)
    z5c <- ifelse(cd80 <= 1034, 1, 0)
    # z6<-hemo z7<-homo
    z8 <- drugs
    # z9<-race z10<-gender z11<-oprior
    z12 <- symptom
    # Convert to factors
    v1a <- as.factor(z1a)
    v1b <- as.factor(z1b)
    v2 <- as.factor(z2)
    v3 <- as.factor(z3)
    v4 <- as.factor(z4)
    v5a <- as.factor(z5a)
    v5b <- as.factor(z5b)
    v5c <- as.factor(z5c)
    v6 <- as.factor(z8)
    v7 <- as.factor(z12)
})

FSconfounders.name <- c("v1a", "v1b", "v2", "v3", "v4", "v5a", "v5b", "v5c", "v6",
    "v7")

outcome.name <- c("time_days")
event.name <- c("cens")
id.name <- c("id")
treat.name <- c("treat")

df.confounders <- df.analysis[, FSconfounders.name]
df.confounders <- dummy(df.confounders)

hr.threshold <- 1.5  # Inititial candidates
hr.consistency <- 1.25  # Candidates for many splits

pconsistency.threshold <- 0.9
maxk <- 4
# maxk is max # of covariates in combination Since we want to allow generation
# of intervals for single covariate allowing for 4 can yield v1, v2 (say), and
# v3,v4 with v3 and v4 generating intervals for a single covariate
```

```r
# Limit timing for forestsearch
max.minutes <- 60
nmin.fs <- 60
# stop.threshold<-0.60 # If any sg meets this, then choose this (stop here);
m1.threshold <- Inf  # Turning this off (Default)
stop.threshold <- 1
# =1 will run through all sg's meeting HR criteria pconsistency.threshold<-0.70
# # Minimum threshold (will choose max among subgroups satisfying)
fs.splits <- 1000  # How many times to split for consistency
# vi is % factor is selected in cross-validation --> higher more important Set
# as liberal since LASSO used for dimension reduction Primarily for ordering
# factors
vi.grf.min <- 0.1
# Null, turns off grf screening Set to 5 for this heavily censored data
d.min <- 5  # Min number of events for both arms (d0.min=d1.min=d.min)
# default=5

sg_focus <- "Nsg"
split_method <- "Random"
pstop_futile <- 0.3
# Stops the consistency evaluation after first subgroup with consistency below
# pstop_futile With idea that since SG's are sorted by hazard ratio estimates,
# once consistency is below pstop_futile it seems unlikely that SG's with lower
# hr's will reach the required consistency criterion

# load('output/fs_actg_Arms_2vs3_final.Rdata') fs.est<-fs_actg_final

fs.est <- forestsearch(df = df.analysis, confounders.name = FSconfounders.name, df.predict = df.analysis
    details = TRUE, sg_focus = sg_focus, split_method = split_method, pstop_futile = pstop_futile,
    outcome.name = outcome.name, treat.name = treat.name, event.name = event.name,
    id.name = id.name, n.min = nmin.fs, hr.threshold = hr.threshold, hr.consistency = hr.consistency,
    fs.splits = fs.splits, stop.threshold = stop.threshold, d0.min = d.min, d1.min = d.min,
    pconsistency.threshold = pconsistency.threshold, max.minutes = max.minutes, maxk = maxk,
    plot.sg = FALSE, vi.grf.min = vi.grf.min)

## Confounders per grf screening v2 v7 v1a v3 v5b v6 v5c v5a v1b v4
## Number of possible subgroups= 1048575
## Number of possible subgroups (in millions)= 1.048575
## # of subgroups based on # variables > k.max and excluded 1042380
## k.max= 4
## Events criteria for control,exp= 5 5
## # of subgroups with events less than criteria: control, experimental 3623 3922
## # of subgroups meeting all criteria = 1442
## # of subgroups fitted (Cox model estimable) = 1442
## Minutes= 0.11965
## Number of criteria not met for subgroup evaluation
## crit.failure
##       0       1       2       3       4
## 1043822     323    3201     481     748
## Number of subgroups meeting HR threshold 91
## Subgroups (1st 10) meeting overall screening thresholds (HR, m1) sorted by focus: (m1,sg_focus)= Inf
##     K   n  E d1  m1  m0   HR L(HR) U(HR) v2.0 v2.1 v7.0 v7.1 v1a.0 v1a.1 v3.0
##  1: 3 225 25 16 Inf Inf 1.61  0.71  3.63    1    0    0    0     0     0    0
##  2: 3 222 51 31 Inf Inf 1.64  0.93  2.87    1    0    0    0     1     0    0
##  3: 4 213 55 28 Inf Inf 1.50  0.88  2.55    1    0    0    0     0     0    0
```

```
##  4: 2 212 50 26 Inf Inf 1.58  0.90  2.75    0    0    0    0     0     0    0
##  5: 3 212 50 26 Inf Inf 1.58  0.90  2.75    0    0    0    0     0     1    0
##  6: 3 201 21 13 Inf Inf 1.58  0.65  3.81    1    0    0    0     0     0    0
##  7: 3 195 22 14 Inf Inf 1.55  0.65  3.68    1    0    0    0     0     0    0
##  8: 4 192 20 13 Inf Inf 1.82  0.72  4.56    1    0    0    0     0     0    0
##  9: 3 191 47 26 Inf Inf 1.97  1.11  3.50    0    0    0    0     0     0    0
## 10: 4 191 47 26 Inf Inf 1.97  1.11  3.50    0    0    0    0     0     1    0
##     v3.1 v5b.0 v5b.1 v6.0 v6.1 v5c.0 v5c.1 v5a.0 v5a.1 v1b.0 v1b.1 v4.0 v4.1
##  1:    0     0     0    0    0     0     0     1     0     0     0    1    0
##  2:    0     1     0    0    0     0     0     0     0     0     0    0    0
##  3:    1     1     0    1    0     0     0     0     0     0     0    0    0
##  4:    1     0     0    0    0     0     0     0     0     0     1    0    0
##  5:    1     0     0    0    0     0     0     0     0     0     1    0    0
##  6:    0     0     0    1    0     0     0     0     0     0     0    1    0
##  7:    0     1     0    0    0     0     0     0     0     0     0    1    0
##  8:    0     0     0    1    0     0     0     1     0     0     0    1    0
##  9:    1     0     0    1    0     0     0     0     0     0     1    0    0
## 10:    1     0     0    1    0     0     0     0     0     0     1    0    0
## Consistency 0.075
## Consistency 0.217
## Consistency 0.079
## Consistency 0.141
## Consistency 0.141
## Consistency 0.06
## Consistency 0.058
## Consistency 0.142
## Consistency 0.835
## Consistency 0.835
## Consistency 0.07
## Consistency 0.07
## Consistency 0.281
## Consistency 0.077
## Consistency 0.077
## Consistency 0.501
## Consistency 0.109
## Consistency 0.17
## Consistency 0.106
## Consistency 0.182
## Consistency 0.182
## Consistency 0.048
## Consistency 0.278
## Consistency 0.404
## Consistency 0.172
## Consistency 0.059
## Consistency 0.06
## Consistency 0.542
## Consistency 0.142
## Consistency 0.076
## Consistency 0.178
## Consistency 0.116
## Consistency 0.174
## Consistency 0.069
## Consistency 0.069
## Consistency 0.149
## Consistency 0.149
```

```
## Consistency 0.118
## Consistency 0.913
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v1a.0 v5b.0 v5c.1 0.913
## Number of subgroups meeting consistency criteria= 1
##    p.consistency Nsg group.id m.index K   M.1   M.2   M.3 M.4
## 1:         0.913 124       23      39 3 v1a.0 v5b.0 v5c.1
##    p.consistency Nsg group.id m.index K   M.1   M.2   M.3 M.4
## 1:         0.913 124       23      39 3 v1a.0 v5b.0 v5c.1
```

```
xx <- fs.est$find.grps$out.found$hr.subgroups
covs.found <- xx[, -c(1:10)]
covs.most <- apply(covs.found, 2, sum)
covs.most <- covs.most[covs.most > 0]
print(covs.most)
```

```
##  v2.0  v7.0  v7.1 v1a.0 v1a.1  v3.0  v3.1 v5b.0 v5b.1  v6.0 v5c.0 v5c.1 v5a.0
##    37    17     1    22    16     7    50    23    17    31    19    15    12
## v5a.1 v1b.0 v1b.1  v4.0  v4.1
##     3     5    33    11    16
```

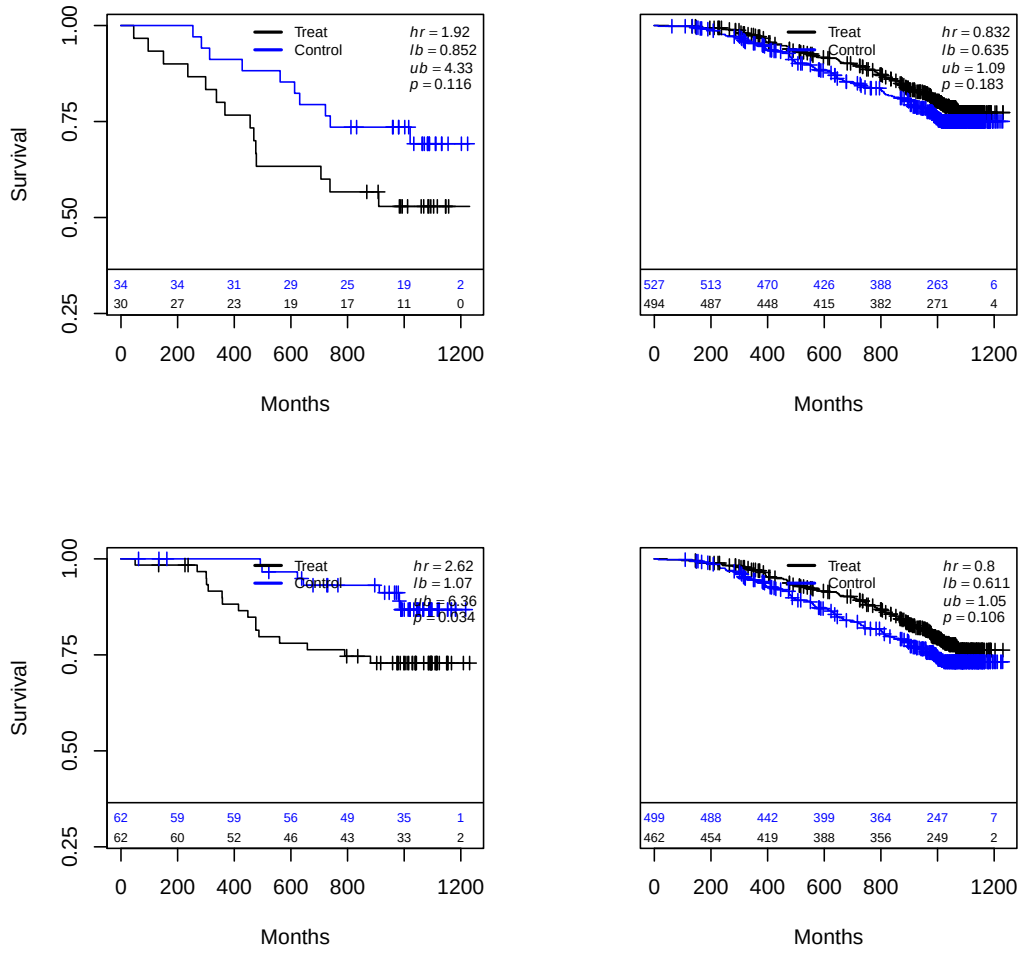
```
print(fs.est$grp.consistency$result)
```

```
##    p.consistency Nsg group.id m.index K   M.1   M.2   M.3 M.4
## 1:         0.913 124       23      39 3 v1a.0 v5b.0 v5c.1
```

```
fs_actg_final <- fs.est

df0.fs <- subset(fs.est$df.pred, treat.recommend == 0)
df1.fs <- subset(fs.est$df.pred, treat.recommend == 1)

# save(fs_actg_final,df.analysis,FSconfounders.name,file='output/fs_actg_Arms_2vs3_final.Rdata')
```

```
# Note, the elements above will need to be re-initiated if running separate
# from above E.g., outcome.names, event.name, ... hr.threshold, etc.

# load('output/fs_actg_Arms_2vs3_final.Rdata') fs.est<-fs_actg_final

library(doParallel)
registerDoParallel(parallel::detectCores(logical = FALSE))

cox.formula.boot <- as.formula(paste("Surv(time_days,cens)~treat"))
split_method <- "Random"
est.loghr <- TRUE

confounders.name <- FSconfounders.name
stop.threshold <- 0.99
max.minutes <- 6

# Suggest running 50, first ... to get timing estimate
NB <- 2000

df_temp <- fs.est$df.pred[, c("id", "treat.recommend")]
dfa <- merge(df.analysis, df_temp, by = "id")
df_boot_analysis <- dfa
```

```r
# Compare with GRF
layout(matrix(c(1, 2, 3, 4), 2, 2, byrow = TRUE))
plot.subgroup(sub1 = df0.grf, sub1C = df1.grf, tte.name = "time_days", event.name = "cens",
    treat.name = "treat", fix.rows = FALSE, byrisk = 200, show.med = FALSE, ymin = 0.4)
plot.subgroup(sub1 = df0.fs, sub1C = df1.fs, tte.name = "time_days", event.name = "cens",
    treat.name = "treat", fix.rows = FALSE, byrisk = 200, show.med = FALSE, ymin = 0.4)
```




```r
fitH <- get_Cox_sg(df_sg = subset(df_boot_analysis, treat.recommend == 0), cox.formula = cox.formula.boo
    est.loghr = est.loghr)
H_obs <- fitH$est_obs  # log(hr) scale
seH_obs <- fitH$se_obs
# Hc observed estimates
fitHc <- get_Cox_sg(df_sg = subset(df_boot_analysis, treat.recommend == 1), cox.formula = cox.formula.bo
    est.loghr = est.loghr)
Hc_obs <- fitHc$est_obs
seHc_obs <- fitHc$se_obs
rm("fitH", "fitHc")
```

```
Ystar_mat <- bootYstar({
    ystar <- get_Ystar(boot)
}, boots = NB, seed = 8316951, counter = "boot", export = fun_arg_list_boot)
# Check dimension
if (dim(Ystar_mat)[1] != NB | dim(Ystar_mat)[2] != nrow(df_boot_analysis)) stop("Dimension of Ystar_mat

tB.start <- proc.time()[3]
# Bootstraps
resB <- bootPar({
    ans <- fsboot_forparallel(boot)
}, boots = NB, seed = 8316951, counter = "boot", export = fun_arg_list_boot)
tB.now <- proc.time()[3]
tB.min <- (tB.now - tB.start)/60

doParallel::stopImplicitCluster()

cat("Minutes for Boots", c(NB, tB.min), "\n")

## Minutes for Boots 2000 48.0389

cat("Projection per 100", c(tB.min * (100/NB)), "\n")

## Projection per 100 2.401945

cat("Propn bootstrap subgroups found =", c(sum(!is.na(resB$H_biasadj_1))/NB), "\n")

## Propn bootstrap subgroups found = 0.9835

# How many timmed out
cat("Number timmed out=", c(sum(is.na(resB$H_biasadj_1) & resB$tmins_search > max.minutes)),
    "\n")

## Number timmed out= 0

H_estimates <- get_dfRes(Hobs = H_obs, seHobs = seH_obs, H1_adj = resB$H_biasadj_1,
    ystar = Ystar_mat, cov_method = "standard", cov_trim = 0.05)

Hc_estimates <- get_dfRes(Hobs = Hc_obs, seHobs = seHc_obs, H1_adj = resB$Hc_biasadj_1,
    ystar = Ystar_mat, cov_method = "standard", cov_trim = 0.05)

print(H_estimates)

##          H0     sdH0 H0_lower H0_upper       H1      sdH1 H1_lower H1_upper
## 1: 2.615041 1.185542 1.075429 6.358799 1.661497 0.1739808 1.353217 2.040007

print(Hc_estimates)

##           H0     sdH0  H0_lower H0_upper        H1      sdH1  H1_lower H1_upper
## 1: 0.8000619 0.110339 0.6105649 1.048372 0.9080085 0.1128311 0.7117342 1.158409

# save(fs.est,Ystar_mat,resB,H_estimates,Hc_estimates,df_boot_analysis,file='output/fsBoot_actg_Arms_2v
```

```
t.done <- proc.time()[3]
t.min <- (t.done - t.start.all)/60
cat("Minutes and hours to finish", c(t.min, t.min/60), "\n")

## Minutes and hours to finish 51.4478 0.8574633
```

Table 1: ACTG-175 FS Analysis: Cox hazard ratio (HR) estimates for the ITT population and subgroups H and H^c. Cox model estimates are based on subgroups: H true (knowing the actual subgroup, a-priori); the estimated subgroup $\hat{H}$; and the bootstrap ($B = 2,000$) bias-correction to $\hat{H}$ estimates, denoted $\hat{H}_{bc}$. Estimates for the complement H^c are defined analogously. The number of subjects in each population (# Subjects) are listed.

	HR Estimate	Lower	Upper	# Subjects
ITT				
ITT	0.900	0.690	1.160	1085
H subgroup estimates				
$\hat{H}$	2.615	1.075	6.359	124
$\hat{H}_{bc}$	1.661	1.353	2.040	124
H-complement subgroup estimates				
$\hat{H}^c$	0.800	0.611	1.048	961
$\hat{H}^c_{bc}$	0.908	0.712	1.158	961

```
cat("Machine=", c(Sys.info()[[4]]), "\n")

## Machine= Mac-Studio-M1-Ultra-2022.local

cat("Number of cores=", c(detectCores(logical = FALSE)), "\n")

## Number of cores= 20
```