```
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)

save_output<-TRUE
```

```
# NOTE: ZDV+DDI = arm 1, DDI = arm 3
t.start.all <- proc.time()[3]
# GRF analysis To guide selection of binary cutpoints
df.analysis <- subset(ACTG175, arms %in% c(1, 3))
df.analysis <- within(df.analysis, {
    id <- as.numeric(c(1:nrow(df.analysis)))
    time_days <- days
    treat <- ifelse(arms == 1, 1, 0)
})

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

```r
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 -0.1751    0.8394   0.1324 -1.323 0.186
##
## Likelihood ratio test=1.76  on 1 df, p=0.1849
## n= 1083, number of events= 231
```

```r
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
```

```r
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   -24.301541   854.000000   9.856152    24.301541   854.000000
## 2          3    17.567065   229.000000  15.188484   -17.567065   229.000000
## 3          4    35.305293   147.000000  22.117601   -35.305293   147.000000
## 4          5   -50.510935   381.000000  14.201781    50.510935   381.000000
## 5          6    63.911098   100.000000  27.079684   -63.911098   100.000000
## 6          7   -19.927428   455.000000  12.957305    19.927428   455.000000
## 21         9   -89.316987   117.000000  24.237157    89.316987   117.000000
## 31        10    72.668998    95.000000  25.744266   -72.668998    95.000000
## 41        11   -41.571025   301.000000  16.049637    41.571025   301.000000
## 51        12   -78.108162    69.000000  40.510692    78.108162    69.000000
## 61        13    39.003171   180.000000  24.678709   -39.003171   180.000000
## 7         14   -61.233827   151.000000  20.444702    61.233827   151.000000
## 8         15    31.866646   155.000000  13.454776   -31.866646   155.000000
##    treated.se       diff depth
## 1    9.856152  -48.60308     1
## 2   15.188484   35.13413     1
## 3   22.117601   70.61059     2
## 4   14.201781 -101.02187     2
## 5   27.079684  127.82220     2
## 6   12.957305  -39.85486     2
## 21  24.237157 -178.63397     3
## 31  25.744266  145.33800     3
## 41  16.049637  -83.14205     3
## 51  40.510692 -156.21632     3
## 61  24.678709   78.00634     3
## 7   20.444702 -122.46765     3
## 8   13.454776   63.73329     3
```

```
##    leaf.node control.mean control.size control.se treated.mean treated.size
## 31        10     72.66900     95.00000   25.74427    -72.66900     95.00000
##    treated.se    diff depth
## 31   25.74427 145.338     3
```

```
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: age <= 28
```

```
# action=1 --> recommend control

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

```
## [1] 95 34
```

```
check <- subset(df.analysis, cd40 <= 273 & karnof > 80 & age <= 29)
print(dim(check))
```

```
## [1] 70 29
```

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

if (save_output) save(grf.est, file = "output/grf_actg_Arms_1vs3_final.Rdata")
```

```
# Stored results load('output/grf_actg_Arms_1vs3_final.Rdata')

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

```
## GRF variables in selected tree
```

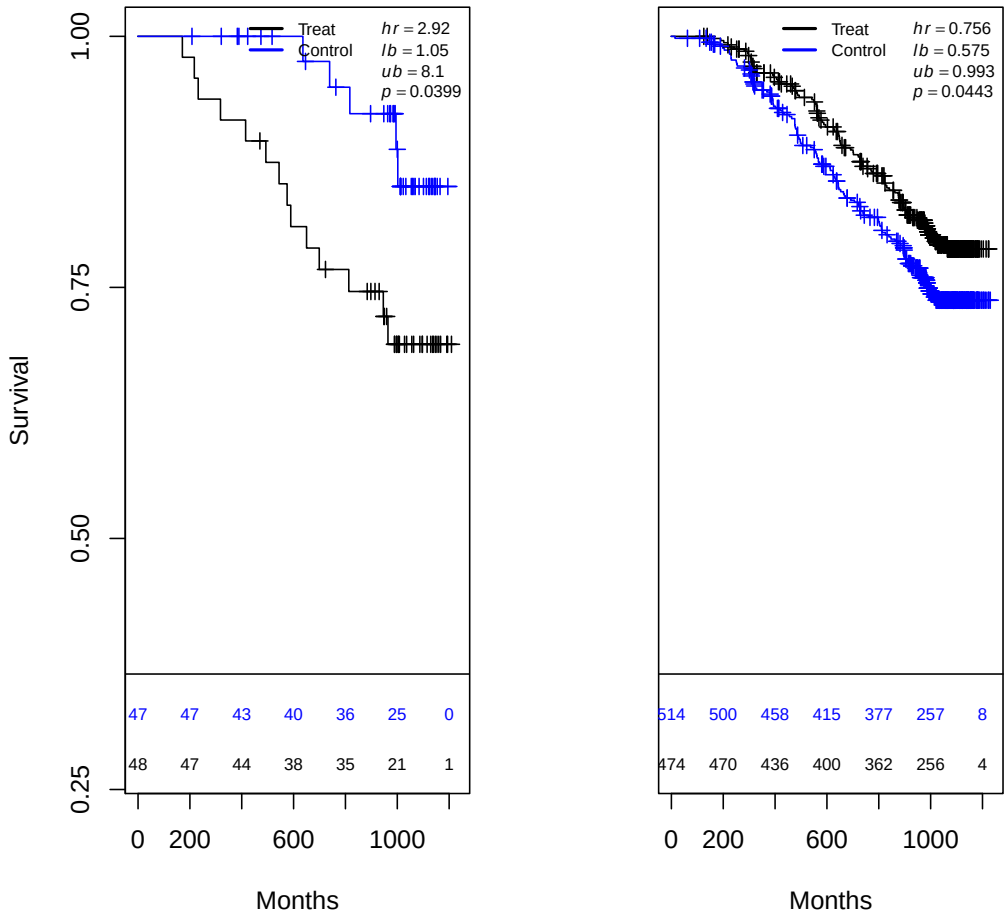
```
print(grf.est$tree.names)
```

```
## [1] "wtkg" "cd40" "age"  "homo"
```

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

```
## GRF cuts wrt selected tree:
```

```
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)
```




```
print(grf.est$tree.cuts)
```

```
## [1] "wtkg <= 73.71" "wtkg <= 60.24" "cd40 <= 336"   "cd40 <= 199"
## [5] "age <= 28"     "homo <= 0"     "wtkg <= 81.7"
```

```
# 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)]
```

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

## Cox-LASSO selected: wtkg karnof cd40 cd80 hemo homo drugs race oprior symptom

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

## GRF cuts wrt selected tree:

print(grf.est$tree.cuts)

## [1] "wtkg <= 73.71" "wtkg <= 60.24" "cd40 <= 336"   "cd40 <= 199"
## [5] "age <= 28"     "homo <= 0"     "wtkg <= 81.7"

#'cd40 <= 273'  'karnof <= 80' 'wtkg <= 81.7' 'age <= 34'    'age <= 29'    'cd40 <= 311'  'cd40 <= 321

# Cox-LASSO selected: wtkg karnof cd40 cd80 hemo homo race symptom

# Considering continuous factors per GRF cuts Only considering hemo, homo,
# race, and symptom per lasso
df.analysis <- within(df.analysis, {
    # Age at 29 and 34
    z1a <- ifelse(age <= 29, 1, 0)
    z1b <- ifelse(age <= 34, 1, 0)
    # Wtkg 81.7
    z2 <- ifelse(wtkg <= 82, 1, 0)
    # Karnof 80
    z3 <- ifelse(karnof <= 80, 1, 0)
    # cd80 ---> median
    z4 <- ifelse(cd80 <= median(cd80), 1, 0)
    # cd40 273, 311, 321
    z5a <- ifelse(cd40 <= 273, 1, 0)
    z5b <- ifelse(cd40 <= 311, 1, 0)
    z5c <- ifelse(cd40 <= 321, 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(z6)
    v7 <- as.factor(z7)
    v8 <- as.factor(z9)
    v9 <- as.factor(z12)
})

FSconfounders.name <- c("v1a", "v1b", "v2", "v3", "v4", "v5a", "v5b", "v5c", "v6",
    "v7", "v8", "v9")
```

```r
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

# 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
# liberal here, since LASSO used for selection; VI (variable importance per
# GRF) used for sorting
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 <- "hr"
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_1vs3_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 v9 v1b v1a v5a v2 v4 v7 v5b v8 v6 v5c v3
## Number of possible subgroups= 16777215
## Number of possible subgroups (in millions)= 16.77722
```

```
## # of subgroups based on # variables > k.max and excluded 16764265
## k.max= 4
## Events criteria for control,exp= 5 5
## # of subgroups with events less than criteria: control, experimental 7654 7982
## # of subgroups meeting all criteria = 2837
## # of subgroups fitted (Cox model estimable) = 2837
## Minutes= 1.1266
## Number of criteria not met for subgroup evaluation
## crit.failure
##        0        1        2        3        4
## 16767102      802     6560      937     1814
## Number of subgroups meeting HR threshold 153
## 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) v9.0 v9.1 v1b.0 v1b.1 v1a.0 v1a.1
##  1: 4  91 21 16 Inf Inf 3.15  1.15  8.61    0    0     0     0     0     1
##  2: 3 105 25 17 Inf Inf 2.91  1.25  6.75    0    0     0     0     0     1
##  3: 4 105 25 17 Inf Inf 2.91  1.25  6.75    0    0     0     1     0     1
##  4: 4  79 14  9 Inf Inf 2.89  0.97  8.64    1    0     0     1     0     0
##  5: 3  90 16 10 Inf Inf 2.79  1.01  7.67    0    0     0     1     0     0
##  6: 3  64 14  9 Inf Inf 2.79  0.93  8.32    0    0     0     1     1     0
##  7: 3  72 16 11 Inf Inf 2.72  0.94  7.83    0    0     0     0     0     1
##  8: 4  72 16 11 Inf Inf 2.72  0.94  7.83    0    0     0     1     0     1
##  9: 4  66 13  8 Inf Inf 2.64  0.86  8.09    1    0     0     1     0     0
## 10: 3  70 18 12 Inf Inf 2.58  0.97  6.88    0    0     0     0     0     1
##     v5a.0 v5a.1 v2.0 v2.1 v4.0 v4.1 v7.0 v7.1 v5b.0 v5b.1 v8.0 v8.1 v6.0 v6.1
##  1:     0     0    0    1    0    0    0    0     0     0    0    0    0    0
##  2:     0     0    0    0    0    0    0    0     0     1    0    0    0    0
##  3:     0     0    0    0    0    0    0    0     0     1    0    0    0    0
##  4:     1     0    1    0    0    0    0    0     0     0    0    0    0    0
##  5:     1     0    1    0    0    0    0    0     0     0    0    0    0    0
##  6:     0     0    1    0    0    0    0    0     0     0    0    0    0    0
##  7:     0     0    0    0    0    0    0    0     0     1    1    0    0    0
##  8:     0     0    0    0    0    0    0    0     0     1    1    0    0    0
##  9:     0     0    1    0    0    0    0    0     0     0    1    0    0    0
## 10:     0     1    0    0    0    0    0    0     0     0    0    0    0    0
##     v5c.0 v5c.1 v3.0 v3.1
##  1:     0     1    1    0
##  2:     0     0    1    0
##  3:     0     0    1    0
##  4:     0     0    0    0
##  5:     0     0    0    0
##  6:     0     0    0    0
##  7:     0     0    0    0
##  8:     0     0    0    0
##  9:     0     0    0    0
## 10:     0     0    1    0
## Consistency 0.962
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v1a.1 v2.1 v5c.1 v3.0 0.962
## Consistency 0.966
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v1a.1 v5b.1 v3.0 0.966
## Consistency 0.966
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v1b.1 v1a.1 v5b.1 v3.0 0.966
```

```
## Consistency 0.665
## Consistency 0.683
## Consistency 0.668
## Consistency 0.662
## Consistency 0.662
## Consistency 0.428
## Consistency 0.544
## Consistency 0.544
## Consistency 0.624
## Consistency 0.48
## Consistency 0.488
## Consistency 0.514
## Consistency 0.514
## Consistency 0.514
## Consistency 0.514
## Consistency 0.607
## Consistency 0.384
## Consistency 0.384
## Consistency 0.384
## Consistency 0.92
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v1a.1 v5c.1 v3.0 0.92
## Consistency 0.92
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v1b.1 v1a.1 v5c.1 v3.0 0.92
## Consistency 0.521
## Consistency 0.697
## Consistency 0.28
## Number of subgroups meeting consistency criteria= 5
##    p.consistency Nsg group.id m.index K   M.1   M.2   M.3  M.4
## 1:         0.962  91      133       1 4 v1a.1  v2.1 v5c.1 v3.0
## 2:         0.966 105      143       2 3 v1a.1 v5b.1  v3.0
## 3:         0.966 105      142       3 4 v1b.1 v1a.1 v5b.1 v3.0
## 4:          0.92 113      132      23 3 v1a.1 v5c.1  v3.0
## 5:          0.92 113      130      24 4 v1b.1 v1a.1 v5c.1 v3.0
##    p.consistency Nsg group.id m.index K   M.1   M.2   M.3  M.4
## 1:         0.966 105      143       2 3 v1a.1 v5b.1  v3.0
## 2:         0.966 105      142       3 4 v1b.1 v1a.1 v5b.1 v3.0
## 3:         0.962  91      133       1 4 v1a.1  v2.1 v5c.1 v3.0
## 4:         0.920 113      132      23 3 v1a.1 v5c.1  v3.0
## 5:         0.920 113      130      24 4 v1b.1 v1a.1 v5c.1 v3.0

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)

##  v9.0  v9.1 v1b.1 v1a.0 v1a.1 v5a.0 v5a.1  v2.0  v2.1  v4.0  v4.1  v7.0  v7.1
##    27     3    86    11    50    39     9    34    28    35    22    10    11
## v5b.0 v5b.1  v8.0  v6.0 v5c.0 v5c.1  v3.0
##    18    40    32    21    14    32    24

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

##    p.consistency Nsg group.id m.index K   M.1   M.2   M.3  M.4
```

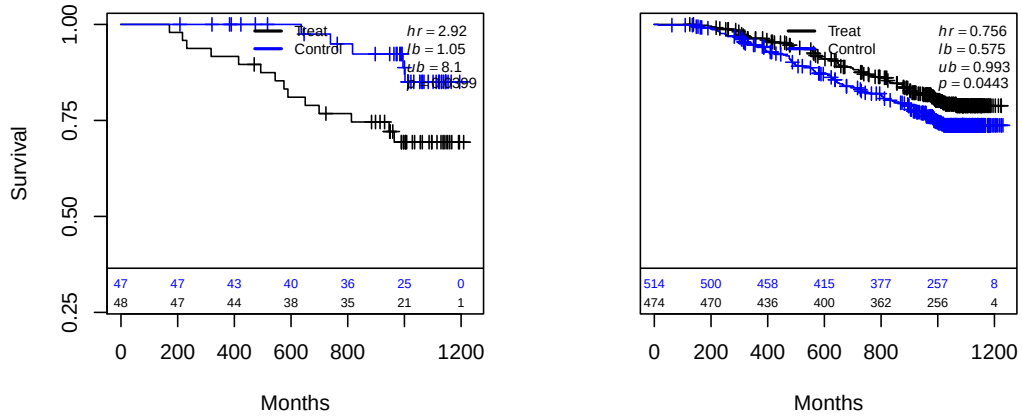
```
## 1:         0.966 105      143       2 3 v1a.1 v5b.1  v3.0
## 2:         0.966 105      142       3 4 v1b.1 v1a.1 v5b.1 v3.0
## 3:         0.962  91      133       1 4 v1a.1  v2.1 v5c.1 v3.0
## 4:         0.920 113      132      23 3 v1a.1 v5c.1  v3.0
## 5:         0.920 113      130      24 4 v1b.1 v1a.1 v5c.1 v3.0
```

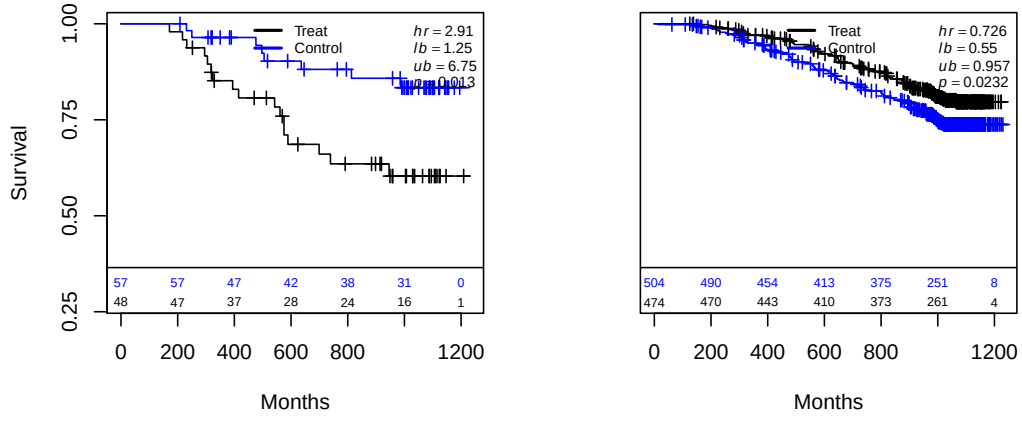
```r
fs_actg_final <- fs.est

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

if (save_output) save(fs_actg_final, df.analysis, FSconfounders.name, file = "output/fs_actg_Arms_1vs3_
```

```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
# 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_1vs3_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"))
est.loghr <- TRUE

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

# Suggest running 50, first ... to get timing estimate
NB <- 1000

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

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 1000 65.45303

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

## Projection per 100 6.545303
```

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

## Propn bootstrap subgroups found = 0.985

# 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: 2.908824 1.2502 1.252779 6.753987 1.430647 0.2714719 0.986309 2.075162

print(Hc_estimates)

##           H0      sdH0  H0_lower  H0_upper        H1      sdH1  H1_lower
## 1: 0.7255405 0.1025153 0.5500359 0.9570448 0.7794267 0.1134905 0.5859138
##    H1_upper
## 1: 1.036852

save(fs.est, Ystar_mat, resB, H_estimates, Hc_estimates, df_boot_analysis, file = "output/fsBoot_actg_A
```

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	0.840	0.650	1.090	1083
H subgroup estimates				
$\hat{H}$	2.909	1.253	6.754	105
$\hat{H}_{bc}$	1.431	0.986	2.075	105
H-complement subgroup estimates				
$\hat{H}^c$	0.726	0.550	0.957	978
$\hat{H}^c_{bc}$	0.779	0.586	1.037	978

```
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 70.0857 1.168095

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

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

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

```
## Number of cores= 20
```