```
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)

suppressMessages(library(gridExtra))

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

# Output grf, fs, and fs bootstrap
#outgrf<-c("output/gbsg_grf.Rdata")
#outfs<-c("output/gbsg_fs.Rdata")
# Boots=2000
outfsboot<-c("output/gbsg_fsboot_B=2000.Rdata")
# Set to null if not outputting
outgrf<-outfs<-NULL
outfsboot<-NULL
```

```
t.start.all <- proc.time()[3]
# GRF analysis To guide selection of binary cutpoints
df.analysis <- gbsg
df.analysis <- within(df.analysis, {
    id <- as.numeric(c(1:nrow(df.analysis)))
    # time to months
    time_months <- rfstime/30.4375
})
```

```r
confounders.name <- c("age", "meno", "size", "grade", "nodes", "pgr", "er")

outcome.name <- c("time_months")
event.name <- c("status")
id.name <- c("id")
treat.name <- c("hormon")

n.min <- 60
dmin.grf <- 12
frac.tau <- 0.6

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= 46.75811
##    leaf.node control.mean control.size control.se treated.mean treated.size
## 1          2     6.126909    82.000000   3.347323    -6.126909    82.000000
## 2          3    -4.186659   604.000000   1.053578     4.186659   604.000000
## 11         4    -8.029472   112.000000   2.791079     8.029472   112.000000
## 21         5     3.617442   177.000000   1.865505    -3.617442   177.000000
## 4          7    -5.918358   356.000000   1.330672     5.918358   356.000000
## 3         10   -11.514406    83.000000   3.149899    11.514406    83.000000
## 41        11     4.348818    97.000000   2.347739    -4.348818    97.000000
## 6         13    -5.931035   324.000000   1.323626     5.931035   324.000000
## 7         14    -7.828243    69.000000   3.723935     7.828243    69.000000
##    treated.se       diff depth
## 1    3.347323  12.253817     1
## 2    1.053578  -8.373318     1
## 11   2.791079 -16.058943     2
## 21   1.865505   7.234883     2
## 4    1.330672 -11.836716     2
## 3    3.149899 -23.028811     3
## 41   2.347739   8.697636     3
## 6    1.323626 -11.862069     3
## 7    3.723935 -15.656486     3
##   leaf.node control.mean control.size control.se treated.mean treated.size
## 1         2     6.126909    82.000000   3.347323    -6.126909    82.000000
##   treated.se     diff depth
## 1   3.347323 12.25382     1

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

## Truncation point for RMST: 46.75811

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: er <= 0

# action=1 --> recommend control

# plot(grf.est$tree) plot(grf.est$tree2) plot(grf.est$tree3)
```

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

## GRF variables in selected tree

print(grf.est$tree.names)

## [1] "er"

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

## GRF cuts wrt selected tree:

print(grf.est$tree.cuts)

## [1] "er <= 0"

# Tree 2
cat("GRF variables in selected tree 2", "\n")

## GRF variables in selected tree 2

print(grf.est$tree2.names)

## [1] "age" "er"

cat("GRF cuts wrt selected tree 2:", "\n")

## GRF cuts wrt selected tree 2:

print(grf.est$tree2.cuts)

## [1] "age <= 50" "age <= 43" "er <= 0"

# Tree 3
cat("GRF variables in selected tree 3", "\n")

## GRF variables in selected tree 3

print(grf.est$tree3.names)

## [1] "age"  "pgr"  "size" "er"

cat("GRF cuts wrt selected tree 3:", "\n")

## GRF cuts wrt selected tree 3:

print(grf.est$tree3.cuts)

## [1] "age <= 48"  "pgr <= 8"   "size <= 36" "age <= 33"  "age <= 43"
## [6] "er <= 0"    "er <= 107"

check <- subset(df.analysis, er <= 0)
print(dim(check))

## [1] 82 13

print(dim(df0.grf))

## [1] 82 18
```

```r
check <- subset(df.analysis, er > 0)
print(dim(check))
```

```
## [1] 604  13
```

```r
print(dim(df1.grf))
```

```
## [1] 604  18
```

```r
# Second candidate with delta=4.6 2nd node for tree=3
check <- subset(df.analysis, age <= 48 & pgr > 8 & age > 43)
print(dim(check))
```

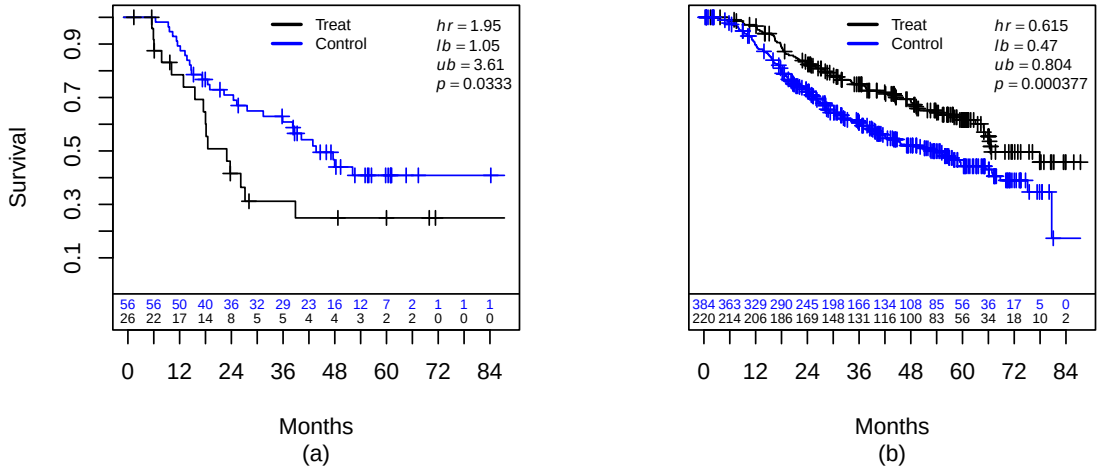
```
## [1] 97 13
```

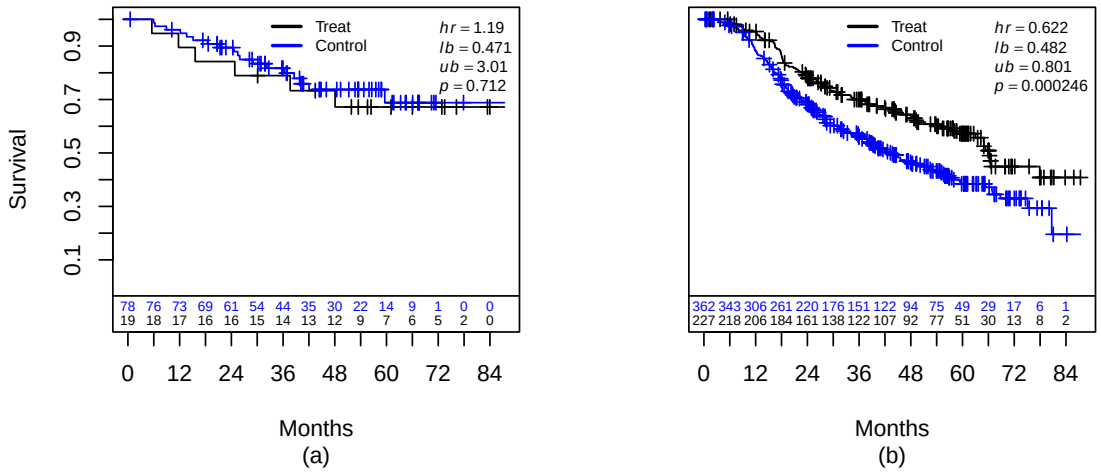
```r
# Examining Tree3, the next sg in favor of control Split: Age<=48, Pgr>8,
# Age>43 (43 < Age <=48) & (Pgr>8)

# Second candidate Not quite 6-month?

df0.grfB <- subset(df.analysis, age <= 48 & pgr > 8 & age > 43)
df1.grfB <- subset(grf.est$data, age > 48 | pgr <= 8 | age <= 43)

layout(matrix(c(1, 2, 3, 4), 2, 2, byrow = TRUE))
plot.subgroup(sub1 = df0.grf, sub1C = df1.grf, tte.name = "time_months", event.name = "status",
    treat.name = "hormon", fix.rows = FALSE, byrisk = 6, show.med = FALSE)
plot.subgroup(sub1 = df0.grfB, sub1C = df1.grfB, tte.name = "time_months", event.name = "status",
    treat.name = "hormon", fix.rows = FALSE, byrisk = 6, show.med = FALSE)
```






```
# plot.subgroup(sub1=check0,sub1C=check1,tte.name='time_months',event.name='status',treat.name='hormon'
# plot.subgroup(sub1=df0.loh,sub1C=df1.loh,tte.name='time_months',event.name='status',treat.name='hormo

if (!is.null(outgrf)) save(grf.est, file = outgrf)
```

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

## Minutes and hours for GRF estimation 0.1698667 0.002831111
```

```
# Recall, GRF splits

t.start <- proc.time()[3]

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

```r
## GRF variables in selected tree

print(grf.est$tree.names)

## [1] "er"

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

## GRF cuts wrt selected tree:

print(grf.est$tree.cuts)

## [1] "er <= 0"

# Tree 2
cat("GRF variables in selected tree 2", "\n")

## GRF variables in selected tree 2

print(grf.est$tree2.names)

## [1] "age" "er"

cat("GRF cuts wrt selected tree 2:", "\n")

## GRF cuts wrt selected tree 2:

print(grf.est$tree2.cuts)

## [1] "age <= 50" "age <= 43" "er <= 0"

# Tree 3
cat("GRF variables in selected tree 3", "\n")

## GRF variables in selected tree 3

print(grf.est$tree3.names)

## [1] "age"  "pgr"  "size" "er"

cat("GRF cuts wrt selected tree 3:", "\n")

## GRF cuts wrt selected tree 3:

print(grf.est$tree3.cuts)

## [1] "age <= 48"  "pgr <= 8"   "size <= 36" "age <= 33"  "age <= 43"
## [6] "er <= 0"    "er <= 107"

df.analysis <- gbsg
df.analysis <- within(df.analysis, {
    id <- as.numeric(c(1:nrow(df.analysis)))
    # time to months
    time_months <- rfstime/30.4375
    z1a <- ifelse(er <= 0, 1, 0)
    z1b <- ifelse(er <= 107, 1, 0)

    z2a <- ifelse(pgr <= 8, 1, 0)
    z2b <- ifelse(pgr <= 74, 1, 0)
```

```
    z3a <- ifelse(age <= 33, 1, 0)
    z3b <- ifelse(age <= 43, 1, 0)
    z3c <- ifelse(age <= 48, 1, 0)
    z3d <- ifelse(age <= 50, 1, 0)  # Close to median=53

    z4 <- ifelse(meno == 0, 1, 0)

    z5 <- ifelse(nodes <= quantile(nodes, c(0.5)), 1, 0)

    z6 <- ifelse(size <= 36, 1, 0)

    z7a <- ifelse(grade == 1, 1, 0)
    z7b <- ifelse(grade == 3, 1, 0)

    # As factors
    v1a <- as.factor(z1a)
    v1b <- as.factor(z1b)

    v2a <- as.factor(z2a)
    v2b <- as.factor(z2b)

    v3a <- as.factor(z3a)
    v3b <- as.factor(z3b)
    v3c <- as.factor(z3c)
    v3d <- as.factor(z3d)

    v4 <- as.factor(z4)
    v5 <- as.factor(z5)
    v6 <- as.factor(z6)

    v7a <- as.factor(z7a)
    v7b <- as.factor(z7b)
})


confounders.name <- c("v1a", "v1b", "v2a", "v2b", "v3a", "v3b", "v3c", "v3d", "v4",
    "v5", "v6", "v7a", "v7b")

# Note, can try smaller subset to check initial code run
# confounders.name<-c('v1a','v1b','v1c',
#'v2a','v2b','v2c',
#'v3a','v3b','v3c')

outcome.name <- c("time_months")
event.name <- c("status")
id.name <- c("id")
treat.name <- c("hormon")

df.confounders <- df.analysis[, confounders.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
```

```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
fs.splits <- 1000  # How many times to split for consistency
# vi is % factor is selected in cross-validation --> higher more important
vi.grf.min <- 0.2
# Null, turns off grf screening

d.min <- 10  # Min number of events for both arms (d0.min=d1.min=d.min)
# default=5

sg_focus <- "hr"


# Default FS implementation (Max consistency with harm) sg_focus<-'Nsg' largest
# SG with at least pconsistency.threshold

# The FS algorithm orders subgroups by largest hazard ratios and then cycles
# through each SG candidate (HR>1.5) to calculate consistency.  Note: there is
# a pstop_futile input which is default at 0.5, meaning that once a subgroup
# with consistency less than 50% the algorithm will stop searching: Since
# meeting 90% consistency for SG's with even lower HR's seems unlikely Setting
# pstop_futile=0 will cycle through all candidates (HR>1.5)

fs.est <- forestsearch(df = df.analysis, confounders.name = confounders.name, df.predict = df.analysis,
    details = TRUE, sg_focus = sg_focus, 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 v1a v2b v1b v5 v7b v3d v2a v6 v3b v4 v3c
## Number of possible subgroups= 4194303
## Number of possible subgroups (in millions)= 4.194303
## # of subgroups based on # variables > k.max and excluded 4185195
## k.max= 4
## Events criteria for control,exp= 10 10
## # of subgroups with events less than criteria: control, experimental 5569 6975
## # of subgroups meeting all criteria = 1271
## # of subgroups fitted (Cox model estimable) = 1271
## Minutes= 0.3021167
## Number of criteria not met for subgroup evaluation
## crit.failure
##       0       1       2       3       4
## 4186466     958    4725     932    1222
## Number of subgroups meeting HR threshold 65
## 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) v1a.0 v1a.1 v2b.0 v2b.1 v1b.0
##  1: 3  78 29 10 17.74   Inf 3.73  1.72  8.08     0     0     0     1     0
##  2: 4  78 29 10 17.74   Inf 3.73  1.72  8.08     0     0     0     1     0
##  3: 4  72 28 10 17.74   Inf 3.58  1.64  7.81     0     0     0     1     0
```

```
##  4: 4  77 28 10 17.74   Inf 3.27  1.51  7.11     0     0     0     1     0
##  5: 4  76 26 10 17.74   Inf 2.95  1.33  6.54     1     0     0     1     0
##  6: 3 102 39 13 17.74   Inf 2.90  1.48  5.66     0     0     0     1     0
##  7: 4  85 33 10 27.17   Inf 2.64  1.24  5.62     0     0     0     1     0
##  8: 4  94 37 12 27.17   Inf 2.61  1.31  5.22     0     0     0     1     0
##  9: 2  61 34 15 18.53 47.97 2.54  1.28  5.04     0     1     0     0     0
## 10: 4  85 30 11 37.65   Inf 2.42  1.15  5.10     0     0     0     0     0
##     v1b.1 v5.0 v5.1 v7b.0 v7b.1 v3d.0 v3d.1 v2a.0 v2a.1 v6.0 v6.1 v3b.0 v3b.1
##  1:     0    0    0     0     0     0     0     0     0    0    0     1     0
##  2:     0    0    0     0     0     0     1     0     0    0    0     1     0
##  3:     1    0    0     0     0     0     0     0     0    0    0     1     0
##  4:     0    0    0     0     0     0     1     0     0    0    1     1     0
##  5:     0    0    0     0     0     0     1     0     0    0    0     1     0
##  6:     0    0    0     0     0     0     1     0     0    0    0     1     0
##  7:     0    0    0     0     0     0     1     0     0    0    0     1     0
##  8:     1    0    0     0     0     0     1     0     0    0    0     1     0
##  9:     0    0    0     0     0     0     0     0     0    0    1     0     0
## 10:     1    0    0     0     0     0     0     0     0    0    1     1     0
##     v4.0 v4.1 v3c.0 v3c.1
##  1:    0    0     0     1
##  2:    0    0     0     1
##  3:    0    0     0     1
##  4:    0    0     0     0
##  5:    0    0     0     0
##  6:    0    0     0     0
##  7:    0    1     0     0
##  8:    0    0     0     0
##  9:    0    0     0     0
## 10:    0    0     0     1
## Consistency 0.989
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v2b.1 v3b.0 v3c.1 0.989
## Consistency 0.989
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v2b.1 v3d.1 v3b.0 v3c.1 0.989
## Consistency 0.988
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v2b.1 v1b.1 v3b.0 v3c.1 0.988
## Consistency 0.971
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v2b.1 v3d.1 v6.1 v3b.0 0.971
## Consistency 0.952
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v1a.0 v2b.1 v3d.1 v3b.0 0.952
## Consistency 0.976
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v2b.1 v3d.1 v3b.0 0.976
## Consistency 0.921
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v2b.1 v3d.1 v3b.0 v4.1 0.921
## Consistency 0.95
## Splitting method, # of splits= Random 1000
## Model, % Consistency Met= v2b.1 v1b.1 v3d.1 v3b.0 0.95
## Consistency 0.959
## Splitting method, # of splits= Random 1000
```

```
## Model, % Consistency Met= v1a.1 v6.1 0.959
## Consistency 0.72
## Consistency 0.677
## Consistency 0.554
## Consistency 0.578
## Consistency 0.531
## Consistency 0.619
## Consistency 0.43
## Number of subgroups meeting consistency criteria= 9
##    p.consistency Nsg group.id m.index K   M.1   M.2   M.3   M.4
## 1:         0.989  78       46       1 3 v2b.1 v3b.0 v3c.1
## 2:         0.989  78       51       2 4 v2b.1 v3d.1 v3b.0 v3c.1
## 3:         0.988  72       47       3 4 v2b.1 v1b.1 v3b.0 v3c.1
## 4:         0.971  77       13       4 4 v2b.1 v3d.1  v6.1 v3b.0
## 5:         0.952  76       23       5 4 v1a.0 v2b.1 v3d.1 v3b.0
## 6:         0.976 102       24       6 3 v2b.1 v3d.1 v3b.0
## 7:         0.921  85       35       7 4 v2b.1 v3d.1 v3b.0  v4.1
## 8:          0.95  94       25       8 4 v2b.1 v1b.1 v3d.1 v3b.0
## 9:         0.959  61       11       9 2 v1a.1  v6.1
##    p.consistency Nsg group.id m.index K   M.1   M.2   M.3   M.4
## 1:         0.989  78       46       1 3 v2b.1 v3b.0 v3c.1
## 2:         0.989  78       51       2 4 v2b.1 v3d.1 v3b.0 v3c.1
## 3:         0.988  72       47       3 4 v2b.1 v1b.1 v3b.0 v3c.1
## 4:         0.976 102       24       6 3 v2b.1 v3d.1 v3b.0
## 5:         0.971  77       13       4 4 v2b.1 v3d.1  v6.1 v3b.0
## 6:         0.959  61       11       9 2 v1a.1  v6.1
## 7:         0.952  76       23       5 4 v1a.0 v2b.1 v3d.1 v3b.0
## 8:         0.950  94       25       8 4 v2b.1 v1b.1 v3d.1 v3b.0
## 9:         0.921  85       35       7 4 v2b.1 v3d.1 v3b.0  v4.1
```

```
# plot.sg=TRUE will plot the estimated subgroups but this is plotted below

# These are the frequency each factor appears in a SG combination

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)
```

```
## v1a.0 v1a.1 v2b.1 v1b.1  v5.0 v7b.0 v7b.1 v3d.1 v2a.0 v2a.1  v6.0  v6.1 v3b.0
##    10     4    28    21     1     7     7    31     1    16     1    10    42
##  v4.1 v3c.1
##    13    26
```

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

```
##    p.consistency Nsg group.id m.index K   M.1   M.2   M.3   M.4
## 1:         0.989  78       46       1 3 v2b.1 v3b.0 v3c.1
## 2:         0.989  78       51       2 4 v2b.1 v3d.1 v3b.0 v3c.1
## 3:         0.988  72       47       3 4 v2b.1 v1b.1 v3b.0 v3c.1
## 4:         0.976 102       24       6 3 v2b.1 v3d.1 v3b.0
## 5:         0.971  77       13       4 4 v2b.1 v3d.1  v6.1 v3b.0
## 6:         0.959  61       11       9 2 v1a.1  v6.1
## 7:         0.952  76       23       5 4 v1a.0 v2b.1 v3d.1 v3b.0
## 8:         0.950  94       25       8 4 v2b.1 v1b.1 v3d.1 v3b.0
## 9:         0.921  85       35       7 4 v2b.1 v3d.1 v3b.0  v4.1
```

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

if (!is.null(outfs)) save(fs.est, df.analysis, confounders.name, file = outfs)
```

```r
t.done <- proc.time()[3]
t.min <- (t.done - t.start)/60
cat("Minutes and hours for FS estimation", c(t.min, t.min/60), "\n")

## Minutes and hours for FS estimation 0.6980167 0.01163361
```

```r
t.start <- proc.time()[3]

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

cox.formula.boot <- as.formula(paste("Surv(time_months,status)~hormon"))
split_method <- "Random"
est.loghr <- TRUE

stop.threshold <- 1
# Can probably set to 0.95 or 0.99, but we set the same as above to mimic the
# estimation algorithm
fs.splits <- 1000
max.minutes <- 6

NB <- 2000

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)
```

```r
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 116.6073

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

## Projection per 100 5.830363

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

## Propn bootstrap subgroups found = 0.9515

# 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: 3.729046 1.470716 1.721426 8.078059 2.260805 0.4491633 1.531621 3.337142

print(Hc_estimates)

##           H0       sdH0  H0_lower  H0_upper        H1       sdH1  H1_lower
## 1: 0.5950175 0.07852577 0.4594041 0.7706634 0.6295678 0.08331257 0.4857362
##     H1_upper
## 1: 0.8159894

if (!is.null(outfsboot)) save(fs.est, Ystar_mat, resB, H_estimates, Hc_estimates,
    df_boot_analysis, file = outfsboot)
```

```r
t.done <- proc.time()[3]
t.min <- (t.done - t.start)/60
cat("Minutes and hours for FS bootstrap", c(t.min, t.min/60), "\n")

## Minutes and hours for FS bootstrap 116.6216 1.943693
```

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

# ITT analysis
cox_itt <- summary(coxph(Surv(time_months, status) ~ hormon, data = fs.est$df.pred))$conf.int
```

```r
# ITT estimates
resITT <- c(round(cox_itt[c(1, 3, 4)], 2), nrow(fs.est$df.pred))

# Forest Search Un-adjusted
Hstat <- c(unlist(H_estimates))[c(1, 3, 4)]
resH_obs <- c(c(Hstat), nrow(df0.fs))
# Bias-corrected
Hstat <- c(unlist(H_estimates))[c(5, 7, 8)]
resH_bc <- c(c(Hstat), nrow(df0.fs))

Hstat2 <- c(unlist(H_estimates))[c(5, 7, 8)]
Hstat2 <- round(Hstat2, 2)
a <- paste0(Hstat2[1], " [")
a <- paste0(a, Hstat2[2])
a <- paste0(a, ",")
a <- paste0(a, Hstat2[3])
a <- paste0(a, "]")
H_bc2 <- c(a)

# Un-adjusted
Hcstat <- c(unlist(Hc_estimates))[c(1, 3, 4)]
resHc_obs <- c(c(Hcstat), nrow(df1.fs))
# Bias-corrected
Hcstat <- c(unlist(Hc_estimates))[c(5, 7, 8)]
resHc_bc <- c(c(Hcstat), nrow(df1.fs))


Hcstat2 <- c(unlist(Hc_estimates))[c(5, 7, 8)]
Hcstat2 <- round(Hcstat2, 2)
a <- paste0(Hcstat2[1], " [")
a <- paste0(a, Hcstat2[2])
a <- paste0(a, ",")
a <- paste0(a, Hcstat2[3])
a <- paste0(a, "]")
Hc_bc2 <- c(a)

res <- rbind(resITT, resH_obs, resH_bc, resHc_obs, resHc_bc)

resf <- as.data.frame(res)

colnames(resf) <- c("HR Estimate", "Lower", "Upper", "$\\#$ Subjects")

rnH <- c("$\\hat{H}$", "$\\hat{H}_{bc}$")
rnHc <- c("$\\hat{H}^{c}$", "$\\hat{H}^{c}_{bc}$")
rnItt <- c("ITT")
rownames(resf) <- c(rnItt, rnH, rnHc)
```

```r
# Resolve conflict with dplyr
library(conflicted)
group_rows <- kableExtra::group_rows

options(knitr.kable.NA = ".", format = "latex")
tab_gbsg <- kbl(resf, longtable = FALSE, align = "c", format = "latex", booktabs = TRUE,
    escape = F, digits = 3, caption = "\\label{tab:gbsg} GBSG FS Analysis: Cox hazard ratio (HR) estimat
```

```
Cox model estimates are based on subgroups: The estimated subgroup $\\hat{H}$; and
the bootstrap ($B=2000$) bias-correction to $\\hat{H}$ estimates, denoted $\\hat{H}_{bc}$.  Estimates fo
The number of subjects in each population ($\\#$ Subjects) are listed.") %>%
    kable_styling(full_width = FALSE, font_size = 9, latex_options = "hold_position") %>%
    group_rows("ITT", 1, 1) %>%
    group_rows("H subgroup estimates", 2, 3) %>%
    group_rows("H-complement subgroup estimates", 4, 5)
```

Table 1: GBSG FS Analysis: Cox hazard ratio (HR) estimates for the ITT population and subgroups H and H^c. Cox model estimates are based on subgroups: The estimated subgroup $\hat{H}$; and the bootstrap ($B = 2000$) 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	0.690	0.540	0.890	686
H subgroup estimates				
$\hat{H}$	3.729	1.721	8.078	78
$\hat{H}_{bc}$	2.261	1.532	3.337	78
H-complement subgroup estimates				
$\hat{H}^c$	0.595	0.459	0.771	608
$\hat{H}^c_{bc}$	0.630	0.486	0.816	608

```
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 117.4924 1.958207

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
```