Build Survival Model: Random Survival Forest

Mingcheng Hu

Table of contents

```r
library(tidyverse)
library(survival)
library(randomForestSRC)
library(caret)
library(survcomp)
library(parallel)
library(doParallel)
library(mcprogress) # wrap mclapply with progress bar.
library(kableExtra) # include knitr automatically

source("/work/users/y/u/yuukias/BIOS-Material/BIOS992/utils/csv_utils.r")
# * Don't use setwd() for Quarto documents!
# setwd("/work/users/y/u/yuukias/BIOS-Material/BIOS992/data")

adjust_type <- ifelse(exists("params"), params$adjust_type, "partial") #
 ↪  options: "minimal", "partial", "full"
impute_type <- ifelse(exists("params"), params$impute_type, "unimputed") #
 ↪  options: "unimputed", "imputed"
include_statin <- ifelse(exists("params"), params$include_statin, "no") #
 ↪  options: "yes", "no"

n_folds <- 10
set.seed(1234)
```

```
# string of parameters
adjust_type_str <- switch(adjust_type,
    minimal = "minimal",
    partial = "partial",
    full = "full"
)
print(paste0("Model Adjustment Type: ", adjust_type_str))
```

```
[1] "Model Adjustment Type: partial"
```

```
impute_type_str <- switch(impute_type,
    unimputed = "unimputed",
    imputed = "imputed"
)
print(paste0("Data Imputation Type: ", impute_type_str))
```

```
[1] "Data Imputation Type: unimputed"
```

Load Data

```
if (include_statin == "yes") {
    data_train <-
 ↪  read.csv(paste0("/work/users/y/u/yuukias/BIOS-Material/BIOS992/data/train_data_",
 ↪  impute_type_str, "_statin.csv"),
        header = TRUE
    )
} else {
    data_train <-
 ↪  read.csv(paste0("/work/users/y/u/yuukias/BIOS-Material/BIOS992/data/train_data_",
 ↪  impute_type_str, ".csv"),
        header = TRUE
    )
}

data_train <- data_train[, -1] # the first column is the index generated by
 ↪  sklearn
(dim(data_train))
```

```
[1] 28127   100
```

```r
data <- select_subset(data_train, type = adjust_type)
(dim(data))
```

```
[1] 28127    75
```

```r
colnames(data)
```

```
 [1] "event"                          "time"
 [3] "HRV_MeanNN"                     "HRV_SDNN"
 [5] "HRV_RMSSD"                      "HRV_SDSD"
 [7] "HRV_CVNN"                       "HRV_CVSD"
 [9] "HRV_MedianNN"                   "HRV_MadNN"
[11] "HRV_MCVNN"                      "HRV_IQRNN"
[13] "HRV_SDRMSSD"                    "HRV_Prc20NN"
[15] "HRV_Prc80NN"                    "HRV_pNN50"
[17] "HRV_pNN20"                      "HRV_MinNN"
[19] "HRV_MaxNN"                      "HRV_HTI"
[21] "HRV_TINN"                       "HRV_LF"
[23] "HRV_HF"                         "HRV_VHF"
[25] "HRV_TP"                         "HRV_LFHF"
[27] "HRV_LFn"                        "HRV_HFn"
[29] "HRV_LnHF"                       "HRV_SD1"
[31] "HRV_SD2"                        "HRV_SD1SD2"
[33] "HRV_S"                          "HRV_CSI"
[35] "HRV_CVI"                        "HRV_CSI_Modified"
[37] "HRV_PIP"                        "HRV_IALS"
[39] "HRV_PSS"                        "HRV_PAS"
[41] "HRV_GI"                         "HRV_SI"
[43] "HRV_AI"                         "HRV_PI"
[45] "HRV_C1d"                        "HRV_C1a"
[47] "HRV_SD1d"                       "HRV_SD1a"
[49] "HRV_C2d"                        "HRV_C2a"
[51] "HRV_SD2d"                       "HRV_SD2a"
[53] "HRV_Cd"                         "HRV_Ca"
[55] "HRV_SDNNd"                      "HRV_SDNNa"
[57] "HRV_ApEn"                       "HRV_ShanEn"
[59] "HRV_FuzzyEn"                    "HRV_MSEn"
[61] "HRV_CMSEn"                      "HRV_RCMSEn"
```

```
[63] "HRV_CD"                          "HRV_HFD"
[65] "HRV_KFD"                         "HRV_LZC"
[67] "HRV_DFA_alpha1"                  "HRV_MFDFA_alpha1_Width"
[69] "HRV_MFDFA_alpha1_Peak"           "HRV_MFDFA_alpha1_Mean"
[71] "HRV_MFDFA_alpha1_Max"            "HRV_MFDFA_alpha1_Delta"
[73] "HRV_MFDFA_alpha1_Asymmetry"      "HRV_MFDFA_alpha1_Fluctuation"
[75] "HRV_MFDFA_alpha1_Increment"
```

```
data <- tibble::as_tibble(data)
```

```
# * It is very hard to compare the HR as different predictors are on
 ↪  different magnitudes, so we need to normalize them.
time_col <- data$time
event_col <- data$event
data <- data %>%
    select(-c(time, event)) %>%
    mutate(across(where(is.numeric), scale)) %>%
    mutate(
        time = time_col,
        event = event_col
    )
```

Note now the interpretation of HR is different! For example, if HR=1.16 for the predictor in the univariate model fitted using scaled data, it means that each standard deviation increase is associated with 16% higher risk of event.

```
# For RSF model, we don't need to exclude the missing values
```

Random Survival Forest (RSF)

Variable Selection

The `method` argument can be set to `vh` instead for variable hunting, which should be used for problems where the number of variables is substantially larger than the sample size.

```
n_cores <- min(parallel::detectCores() - 1, 32)
cl <- makeCluster(n_cores)
registerDoParallel(cl)
```

```r
rsf_var_select <- var.select.rfsrc(Surv(time, event) ~ .,
    data = data,
    method = "md",
    seed = 1234,
    ntree = 200,
    parallel = TRUE
) # minimal depth variable selection
```

```
running forests ...
minimal depth variable selection ...


-----------------------------------------------------------
family             : surv
var. selection     : Minimal Depth
conservativeness   : medium
x-weighting used?  : TRUE
dimension          : 73
sample size        : 26782
ntree              : 200
nsplit             : 10
mtry               : 25
nodesize           : 2
refitted forest    : FALSE
model size         : 73
depth threshold    : 15.4073
PE (true OOB)      : 44.7123


Top variables:
                             depth   vimp
HRV_Prc20NN                  2.005  0.000
HRV_CD                       3.375  0.004
HRV_MaxNN                    3.480  0.016
HRV_KFD                      3.760  0.004
HRV_SI                       4.045  0.002
HRV_ApEn                     4.420  0.002
HRV_MedianNN                 4.600 -0.001
HRV_LZC                      4.695 -0.004
HRV_PIP                      4.705  0.000
HRV_IALS                     4.810  0.000
```

```
HRV_HTI                      4.850 -0.001
HRV_PI                       4.890  0.001
HRV_pNN50                    4.905  0.002
HRV_MadNN                    4.915  0.003
HRV_MeanNN                   4.950  0.008
HRV_ShanEn                   5.020 -0.001
HRV_MinNN                    5.115  0.007
HRV_TINN                     5.165  0.004
HRV_PAS                      5.175  0.003
HRV_SD2a                     5.175  0.009
HRV_IQRNN                    5.185 -0.003
HRV_GI                       5.240  0.001
HRV_C2d                      5.370  0.002
HRV_CMSEn                    5.375  0.003
HRV_RCMSEn                   5.415 -0.001
HRV_MFDFA_alpha1_Peak        5.420  0.002
HRV_MFDFA_alpha1_Asymmetry   5.460 -0.001
HRV_Prc80NN                  5.475  0.002
HRV_MCVNN                    5.505  0.000
HRV_pNN20                    5.570  0.003
HRV_PSS                      5.600  0.000
HRV_HFn                      5.635  0.002
HRV_SDRMSSD                  5.685  0.002
HRV_VHF                      5.700  0.003
HRV_AI                       5.750  0.003
HRV_LF                       5.755  0.002
HRV_MFDFA_alpha1_Max         5.755  0.002
HRV_C1a                      5.775  0.001
HRV_FuzzyEn                  5.800  0.000
HRV_MFDFA_alpha1_Fluctuation 5.805  0.001
HRV_CSI_Modified             5.850  0.005
HRV_C2a                      5.865  0.002
HRV_MFDFA_alpha1_Width       5.945  0.001
HRV_C1d                      5.965 -0.002
HRV_Ca                       5.965  0.001
HRV_MFDFA_alpha1_Delta       5.975  0.001
HRV_MFDFA_alpha1_Mean        6.000 -0.002
HRV_SD2                      6.020  0.004
HRV_LFHF                     6.025  0.001
HRV_MFDFA_alpha1_Increment   6.100  0.001
HRV_Cd                       6.105  0.001
HRV_DFA_alpha1               6.125 -0.001
HRV_HFD                      6.135  0.000
```

```
HRV_LFn                      6.160  0.001
HRV_MSEn                     6.190  0.000
HRV_CVSD                     6.240  0.000
HRV_TP                       6.295  0.001
HRV_SD1a                     6.320 -0.001
HRV_SDNN                     6.390  0.005
HRV_SD1SD2                   6.395  0.002
HRV_HF                       6.425  0.001
HRV_CVNN                     6.430  0.001
HRV_LnHF                     6.445  0.001
HRV_SDNNa                    6.535  0.004
HRV_S                        6.635  0.001
HRV_CSI                      6.670  0.001
HRV_CVI                      6.760  0.003
HRV_SDNNd                    6.880  0.002
HRV_SD2d                     6.920  0.001
HRV_SDSD                     6.985  0.000
HRV_SD1d                     7.010  0.001
HRV_SD1                      7.020  0.000
HRV_RMSSD                    7.195  0.000
-----------------------------------------------------------
```

```
stopCluster(cl)
```

```
vars_ranked <- rsf_var_select$topvars
```

Cross Validation to Select the Best Number of Features

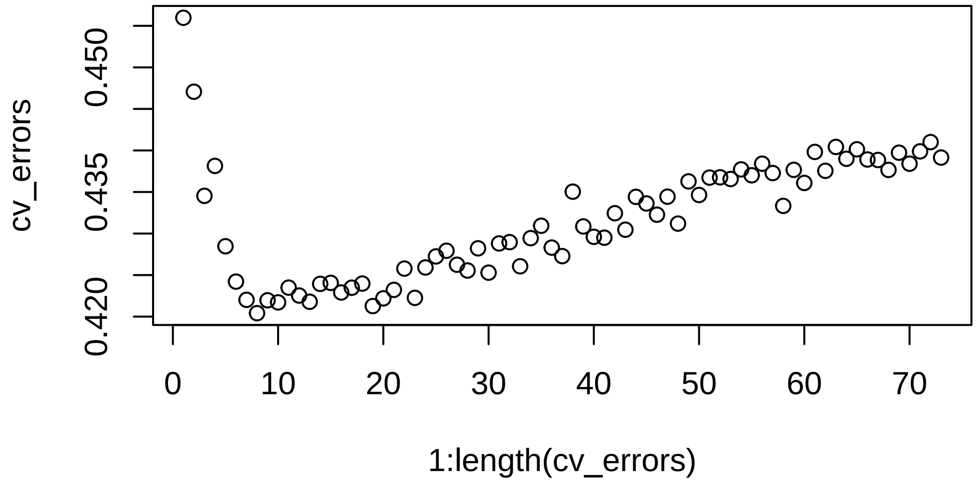
We will use 10-fold cross validation to select the best number of features used in the model.

```
set.seed(1234)
folds <- createFolds(data$event, k = n_folds) # return indices of folds

cv_errors <- pmclapply(seq(1, length(vars_ranked), by = 1),
 ↪  function(num_vars) {
    selected_vars <- vars_ranked[1:num_vars]

    fold_errors <- sapply(folds, function(fold_idx) {
        train_data_fold <- data[-fold_idx, c("time", "event", selected_vars)]
        val_data_fold <- data[fold_idx, c("time", "event", selected_vars)]
        model <- rfsrc.fast(Surv(time, event) ~ .,
            data = train_data_fold,
            ntree = 200,
            forest = TRUE
        )
        pred <- predict(model,
            newdata = val_data_fold,
            na.action = "na.impute"  # * There may be missing values in the
             ↪  dataset
        )$predicted  # define pred has attributes survival(sample_size*time)
 ↪  and predicted(sample_size) for risk
        # Use C-index to measure the performance of the model
        1 - concordance.index(
            pred,  # pass risk prediction for first argument
            val_data_fold$time,
            val_data_fold$event
        )$c.index
    })
    mean(fold_errors)
}, title = "Cross Validation to Select the Best Number of Features")
```

```
cv_errors <- as.numeric(cv_errors)
plot(1:length(cv_errors), cv_errors)
```



```
best_num_vars <- which.min(cv_errors)
vars_selected <- vars_ranked[1:best_num_vars]
```

```
print(paste0("The best number of features to retain is ", best_num_vars))
```

```
[1] "The best number of features to retain is 8"
```

Model Fitting

```
data_selected <- data[, c("time", "event", vars_selected)]

# Before formally fitting the model, we can tune the hyperparameters to find:
# 1. optimal mtry (possible split at each node)
# 2. optimal nodesize (minimum size of terminal nodes)
rsf_tuned <- tune.rfsrc(
    Surv(time, event) ~ .,
    data = data_selected,
)
```

```r
rsf_model <- rfsrc(Surv(time, event) ~ .,
    data = data_selected,
    ntree = 500,
    mtry = rsf_tuned$best.mtry,
    nodesize = rsf_tuned$best.nodesize
)
```

```r
# We also fit the full model
rsf_tuned_full <- tune.rfsrc(
    Surv(time, event) ~ .,
    data = data,
)

rsf_model_full <- rfsrc(Surv(time, event) ~ .,
    data = data,
    ntree = 1000,
    mtry = rsf_tuned_full$best.mtry,
    nodesize = rsf_tuned_full$best.nodesize
)
```

```r
# plot.rfsrc?
# plot.variable.rfsrc?
# print.rfsrc?
```