Tutorial of R Package icRSF

Hui Xu, Xiangdong Gu, Mahlet G. Tadesse and Raji Balasubramanian

1 Introduction

In this tutorial, we illustrate how the R package *icRSF* can be used to obtain variable importance in high dimensional datasets. The package is appropriate for settings in which a silent event is observed through sequentially administered, error-prone self-reports or laboratory based diagnostic tests. This tutorial accompanies the paper Xu, H., et al. (2018) [1].

2 Setup

2.1 Install package

The R package **icRSF** can be downloaded from CRAN (https://cran.r-project.org/web/packages/icRSF/index.html).

```
install.packages("icRSF")
```

2.2 Load package

```
library(icRSF)
```

3 Variable Selection with icRSF

3.1 Variable Importance

Variable importance is calculated using the function **icrsf()**, which requires the following arguments:

- **simdata**: name of the data frame that includes the variables subject, testtimes, test result.
- **subject**: vector of subject IDs of length Mx1.
- **testtimes**: vector of visit or test times of length Mx1.
- **result**: vector of binary diagnostic test results (0 = negative for event of interest; 1 = positive for event of interest) of length Mx1.
- **sensitivity**: the sensitivity of the diagnostic test.
- **specificity**: the specificity of the diagnostic test.
- **Xmat**: a N x P matrix of covariates.
- **root.size**: minimum number of subjects in a terminal node.
- **ntree**: number of survival trees.

- **ns**: number of covariate selected at each node to split the tree.
- **node**: For parallel computation, specify the number of nodes.
- **pval**: P-value threshold from a Likelihood Ratio Test to determine when to stop growing the tree.

Let N and P denote the number of subjects and number of variables in the dataset, respectively. Let M denote the total number of visits, summed over all subjects in the study [i.e. M denotes the number of diagnostic test results available for all subjects in the study]. This algorithm builds a user-defined number of survival trees, using bootstrapped datasets. The subjects left out of each bootstrap sample is referred to as 'Out of Bag' (OOB) data. Using the OOB data in each tree, a permutation-based measure of variable importance for each of the P variables is obtained. For more details, please refer to the publication [1].

For example, in the illustration below, we assume that the test sensitivity is 0.61, the test specificity is 0.99. We select the square root of the number of columns in matrix Xmat at each node as the candidate covariates to find the best split. We stop growing the tree when the terminal node size is smaller than 10 or when there are no covariates among those randomly selected with likelihood ratio p values less than 0.1. We build 100 trees in the forest and parallelize the computation over 2 nodes.

```
library(parallel)
data(Xmat)
data(simdata)
vimp <- icrsf(data=simdata, subject=ID, testtimes=time, result=result, sensitivity=0.61,
             specificity=0.99, Xmat=Xmat, root.size=10, ntree=100, ns=sqrt(ncol(Xmat)), node=2,
             pval=0.1)
```

The output of the function is a p x 1 vector of the ensemble variable importance metric obtained from the proposed algorithm (icRSF) [1].

3.2 Output trees

Survival trees can be constructed using the **treebuilder()** function, which requires the following arguments:

- **simdata**: name of the data frame that includes the variables subject, testtimes, result.
- **subject**: vector of subject IDs of length Mx1.
- **testtimes**: vector of visit or test times of length Mx1.
- **result**: vector of binary diagnostic test results (0 = negative for event of interest; 1 = positive for event of interest) of length Mx1.
- **sensitivity**: the sensitivity of the diagnostic test.
- **specificity**: the specificity of the diagnostic test.
- **Xmat**: a N x P matrix of covariates.
- **root.size**: minimum number of subjects in a terminal node.
- **ns**: number of covariate selected at each node to split the tree.
- **pval**: P-value threshold from a Likelihood Ratio Test to determine when to stop growing the tree.

As denoted in the previous section, Let N and P denote the number of subjects and number of variables in the dataset, respectively. Let M denote the total number of visits, summed over all subjects in the study [i.e. M denotes the number of diagnostic test results available for all subjects in the study]. This function builds a single survival tree in a matrix format and indicated what subjects will be used as Out-of-bag sample for prediction.

For example, in the illustration below, we assume that the test sensitivity is 0.61, the test specificity is 0.99. The bootstrap sampling includes approximately 66% of the subjects to build the tree. We randomly select *ns* columns of the matrix Xmat at each node as the candidate covariates to find the best split, where *ns* is set to equal the square root of the number of columns in Xmat. We stop growing the tree when the terminal node size is smaller than 10 or when there are no covariates among those randomly selected with likelihood ratio p values less than 0.1.

```
data(Xmat)
data(simdata)
tree <- treebuilder(data=simdata, subject=ID, testtimes=time, result=result, sensitivity=0.61,
                    specificity=0.99, Xmat=Xmat, root.size=10, ns=sqrt(ncol(Xmat)), pval=0.1)
```

The output of the function is a matrix of the tree structure. Each row of this matrix corresponds to information regarding a node of the tree. The columns of this output matrix are:

1. Column 1: row number in the output matrix which includes information on the left daughter node.

2. Column 2: row number in the output matrix which includes information on the right daughter node.

3. Column 3: Index (column in data matrix) corresponding to variable selected as the best split.

4. Column 4: Value of the variable in Column 3 at which split is determined.

5. Column 5 - Column $(M + 5)$: estimates of parameters $\beta, S_2, \ldots, S_{j+1}$, where j = 1, …, M.

6. Last column: Number of subjects in the node.

3.3 Other uses

The function **simout()** is used to simulate error-prone test results for a user-defined vector of test times for each of the N subjects, given an user input NxP design matrix (Xmat). The function requires the following arguments:

- **Xmat**: a user-defined covariate matrix of N rows and P columns.

- **testtimes**: a vector of times at which self-reported outcomes are collected for all subjects.

- **sensitivity**: the sensitivity of the self-report.

- **specificity**: the specificity of the self-report.

- **noevent**: denotes the probability of remaining event free by study end (or the complement of cumulative incidence the event during the study period).

- **betas**: denotes the vector of regression coefficients associated with the set of biomarkers in the Cox proportional hazards model.

- **design**: denotes whether test results are missing after the first positive result. 'NMISS' denotes no missing tests after first positive and 'NTFP' denotes all tests are missing after first positive. (The default is 'NMISS').

For example, assume **Xmat** is the user-defined covariate matrix and there are 4 visits (1-4). The sensitivity and specificity of the test are 0.61 and 0.99, respectively. The cumulative event rate is set to equal 0.3 (noevent = 1 - cumulative rate). Assume the first five covariates in **Xmat** are associated with the outcome with regression coefficients all equal to $\beta = 0.81$ and that there are no test results available after the first positive test result.

```
data(Xmat)
sim <- simout(Xmat, testtimes=1:4, sensitivity=0.61, specificity=0.99, noevent=0.7,
              betas=c(rep(0.81, 5), rep(0, ncol(Xmat)-5)), design="NTFP")
```

The output of the function is a data frame with simulated longitudinal form of observed test results [1 row per subject per test time]. The dimensions of this dataframe are M x 3, where first column is the subject ID, second column is the test time and the third column is the binary test result [1 = positive, indicating event of interest has occurred; 0=negative]. For more details, please refer to the paper [1] and R package manual (https://cran.r-project.org/web/packages/icRSF/index.html).

4 R code for the paper and supplementary material

In this Section, we provide the R code to recreate all simulation results reported in the main paper [1]. We note that the results from Table 1 and Figure 1(a) can be regenerated using the code below. However, the code provided below for Tables 2 and 3 and Figure 1(b)-(c) depend on the GWAS and 'omics' datasets that we are not able to provide due to restrictions.

4.1 Simulation studies assuming continuous covariates

Here we present the sample R code to obtain **Table 1** of the paper [1]. This code is standalone and does not require access to datasets.

```
simul1 <- function(sens, spec, noeve, des){
Xmat <- matrix(rnorm(10000, 0, 1), nrow=100, ncol=100)
simdata <- simout(Xmat, testtimes=1:4, sensitivity=sens, specificity=spec, noevent=noeve,
              betas=c(rep(2, 5), rep(0, ncol(Xmat)-5)), design=des)

# Calculate variable importance using icRSF
vimp <- icrsf(data=simdata, subject=ID, testtimes=time, result=result, sensitivity=sens,
             specificity=spec, Xmat=Xmat, root.size=10, ntree=1000, ns=sqrt(ncol(Xmat)), node=2)
o.r <- rev(order(vimp))
ord <- o.r[1:5]
d <- 1 * (is.element(ord, 1:5))
return(mean(d))
}

### Generate the results reported in Table 1
simul1(sens=1, spec=1, noeve=0.9, des="NMISS")
simul1(sens=0.75, spec=1, noeve=0.9, des="NMISS")
simul1(sens=0.61, spec=0.995, noeve=0.9, des="NMISS")
simul1(sens=1, spec=0.9, noeve=0.9, des="NMISS")
simul1(sens=1, spec=1, noeve=0.7, des="NMISS")
simul1(sens=0.75, spec=1, noeve=0.7, des="NMISS")
simul1(sens=0.61, spec=0.995, noeve=0.7, des="NMISS")
simul1(sens=1, spec=0.9, noeve=0.7, des="NMISS")
simul1(sens=1, spec=1, noeve=0.9, des="NTFP")
simul1(sens=0.75, spec=1, noeve=0.9, des="NTFP")
simul1(sens=0.61, spec=0.995, noeve=0.9, des="NTFP")
simul1(sens=1, spec=0.9, noeve=0.9, des="NTFP")
simul1(sens=1, spec=1, noeve=0.7, des="NTFP")
simul1(sens=0.75, spec=1, noeve=0.7, des="NTFP")
simul1(sens=0.61, spec=0.995, noeve=0.7, des="NTFP")
simul1(sens=1, spec=0.9, noeve=0.7, des="NTFP")
```

4.2 Simulation studies incorporating GWAS data

Here we present the code to obtain **Table 2** and **Table 3** discussed in **Section 3** in the paper [1]. This code requires the data matrix $Xmat$ below.

Main functions:

```
# read in GWAS binary covariates
data(Xmat)
Xmat <- Xmat[1:100, 1:100]
source("maf.rda")
# Simulation function
simul2 <- function(sens, spec, noeve, des){
# Randomly select biomarker
idx <- sample(100, 5, replace=F)
beta <- rep(0, 100)
beta[idx] <- 0.81
simdata <- simout(Xmat, testtimes=1:8, sensitivity=sens, specificity=spec, noevent=noeve,
              betas=beta, design=des)
vimp <- icrsf(data=simdata, subject=ID, testtimes=time, result=result, sensitivity=sens,
             specificity=spec, Xmat=Xmat, root.size=3, ntree=1000, ns=sqrt(ncol(Xmat)), node=2)
o.r <- rev(order(vimp))
d <- 1 * (is.element(1:100, ord))
d1 <- d[idx]
f <- maf[idx]
return(list(vimp=vimp, id=id, d1=d1, f=f, d=d))
}
result.analysis <- function(sens, spec, noeve, des){
  result <- mclapply(1:100, simul2(sens=1, spec=1, noeve=0.9, des="NMISS"))
  l1 <- lapply(result, function(x) rbind(x$id, x$d1, x$f))
  ll1 <- do.call("cbind", l1)
  ll11 <- ll1[, ll1[3, ] <= 0.35 | ll1[3, ] >= 0.65]
  ll21 <- ll2[, ll2[3, ] > 0.35 & ll2[3, ] < 0.65]
  pall <- rowMeans(ll1)[2]
  return(c(pall, rowMeans(ll11)[2], rowMeans(ll21)[2]))
}
```

Table 2: Setting corresponding to no missing tests.

```
result.analysis(sens=0.75, spec=1, noeve=0.9, des="NMISS")
result.analysis(sens=0.61, spec=0.995, noeve=0.9, des="NMISS")
result.analysis(sens=1, spec=0.9, noeve=0.9, des="NMISS")
result.analysis(sens=1, spec=1, noeve=0.7, des="NMISS")
result.analysis(sens=0.75, spec=1, noeve=0.7, des="NMISS")
result.analysis(sens=0.61, spec=0.995, noeve=0.7, des="NMISS")
result.analysis(sens=1, spec=0.9, noeve=0.7, des="NMISS")
```

Table 3: Setting corresponding to missing all tests after first positive.

```
result.analysis(sens=1, spec=1, noeve=0.9, des="NTFP")
result.analysis(sens=0.75, spec=1, noeve=0.9, des="NTFP")
result.analysis(sens=0.61, spec=0.995, noeve=0.9, des="NTFP")
result.analysis(sens=1, spec=0.9, noeve=0.9, des="NTFP")
result.analysis(sens=1, spec=1, noeve=0.7, des="NTFP")
result.analysis(sens=0.75, spec=1, noeve=0.7, des="NTFP")
result.analysis(sens=0.61, spec=0.995, noeve=0.7, des="NTFP")
result.analysis(sens=1, spec=0.9, noeve=0.7, des="NTFP")
```

4.3 Simulation studies incorporating omics data

Here we present the code to obtain **Supplemental Table 1** discussed in **Section 3** of the paper [1] incorporating data from a cardiovascular disease omics study. This code requires the data matrix *meta* below.

```
simul3 <- function(sens, spec, noeve, des){
Xmat <- meta[1:100, sample(1:625, 100, replace=F)]
simdata <- simout(Xmat, testtimes=1:4, sensitivity=sens, specificity=spec, noevent=noeve,
              betas=c(rep(2, 5), rep(0, ncol(Xmat)-5)), design=des)
# Calculate variable importance
vimp <- icrsf(data=simdata, subject=ID, testtimes=time, result=result, sensitivity=sens,
             specificity=spec, Xmat=Xmat, root.size=3, ntree=1000, ns=sqrt(ncol(Xmat)), node=2)
o.r <- rev(order(vimp))
ord <- o.r[1:5]
d <- 1 * (is.element(ord, 1:5))
return(mean(d))
}
simul3(sens=1, spec=1, noeve=0.9, des="NMISS")
simul3(sens=0.75, spec=1, noeve=0.9, des="NMISS")
simul3(sens=0.61, spec=0.995, noeve=0.9, des="NMISS")
simul3(sens=1, spec=0.9, noeve=0.9, des="NMISS")
simul3(sens=1, spec=1, noeve=0.7, des="NMISS")
simul3(sens=0.75, spec=1, noeve=0.7, des="NMISS")
simul3(sens=0.61, spec=0.995, noeve=0.7, des="NMISS")
simul3(sens=1, spec=0.9, noeve=0.7, des="NMISS")
simul3(sens=1, spec=1, noeve=0.9, des="NTFP")
simul3(sens=0.75, spec=1, noeve=0.9, des="NTFP")
simul3(sens=0.61, spec=0.995, noeve=0.9, des="NTFP")
simul3(sens=1, spec=0.9, noeve=0.9, des="NTFP")
simul3(sens=1, spec=1, noeve=0.7, des="NTFP")
simul3(sens=0.75, spec=1, noeve=0.7, des="NTFP")
simul3(sens=0.61, spec=0.995, noeve=0.7, des="NTFP")
simul3(sens=1, spec=0.9, noeve=0.7, des="NTFP")
```

4.4 Visualization of Variable Importance in simulation studies

Here we present the code used to produce **Figure 1** in **Section 3** of the paper [1].

```
library(ggpubr)
# The code used to produce {\bf Figure 1(a)-1(c)} of the paper is shown below:
vimp10 <- mclapply(1:100, function(x) simul1(sens=0.61, spec=0.995, noeve=0.9, des="NMISS"))
vimp1 <- colSums(do.call("rbind", vimp10))
Var.names <- paste0("X", 1:100)
names(vimp1) <- "vimp1"
v1 <- data.frame(Var.names, vimp1)
v1$Var.names <- factor(v1$Var.names, levels = v1$Var.names)
v1$group <- c(rep("imp",5), rep("nonimp",95))
```

```
g1 = ggbarplot(v1,
               x = "Var.names",
               y = "vimp1",
               color = "group",
               fill = "group",
               palette = c("#FC4E07", "lightgray"),
               xlab = "Variable Index",
               ylab = "Varialbe Importance",
               main="(a)",
               font.main=10,
               x.text.angle = 90,
               ggtheme = theme_bw()
) + rremove("grid")+rremove("legend")+theme(plot.title = element_text(hjust = 0.5))

vimp20 <- simul2(sens=0.61, spec=0.995, noeve=0.9, des="NMISS")
sel.id <- vimp20[[2]]
vimp2 <- c(vimp20[[1]][id], vimp20[[1]][-id])
v2 <- data.frame(Var.names, vimp2)
v2$Var.names <- factor(v2$Var.names, levels = v2$Var.names)
v2$group <- c(rep("high",2), rep("low",3), rep("nonsig", 95))

g2 = ggbarplot(v2,
               x = "Var.names",
               y = "vimp2",
               color = "group",
               fill = "group",
               palette = c("#FC4E07", "blue", "lightgray"),
               xlab = "Variable Index",
               ylab = "Varialbe Importance",
               main="(b)",
               font.main=10,
               x.text.angle=90,
               ggtheme = theme_bw()
) + rremove("grid")+rremove("legend")+theme(plot.title = element_text(hjust = 0.5))

vimp30 <- lapply(result, function(x) getvimp(x, 1))
vimp3 <- colSums(do.call("rbind", vimp30))
names(vimp3) <- "vimp3"
v3 <- data.frame(Var.names, vimp3)
v3$Var.names <- factor(v3$Var.names, levels = v3$Var.names)
v3$group <- c(rep("imp",5), rep("nonimp",95))

g3 = ggbarplot(v3,
               x = "Var.names",
               y = "vimp3",
               color = "group",
               fill = "group",
               palette = c("#FC4E07", "lightgray"),
               xlab = "Variable Index",
               ylab = "Varialbe Importance",
               main="(c)",
               font.main=10,
               x.text.angle=90,
               ggtheme = theme_bw()
) + rremove("grid")+rremove("legend")+theme(plot.title = element_text(hjust = 0.5))

ggarrange(g1, g2, g3, nrow=3, ncol=1)
```

4.5 Effects of self-reported outcomes on variable importance by Random Survival Forests

Here we present the function used to produce **Figure 1** in the **Supplementary Materials** of the paper [1].

```
library(randomForestSRC)
norm.err <- function(testtimes, sensitivity, specificity, noevent, beta, nbiomarker, n.tree){
  Xmat <- matrix(rnorm(10000, 0, 1), nrow=100, ncol=100)
  ntest <- length(unique(testtimes))
  nsub <- nrow(Xmat)
  ncov <- ncol(Xmat)
  betas <- c(rep(beta, nbiomarker), rep(0, ncov-nbiomarker))
  blambda <- -log(noevent)/max(testtimes)
  lambdat <- blambda * exp(Xmat%*%betas)
  y <- rexp(nsub, lambdat)
  time <- rep(testtimes, nsub)
  id <- rep(1:nsub, each = ntest)
  y <- rep(y, each = ntest)
  obs <- y < time
  prob <- ifelse(obs, sensitivity, 1-specificity)
  result <- rbinom(length(obs), 1, prob)
  simdata <- data.frame(id, time, result)
  test.result <- matrix(simdata$result, nrow = nsub, byrow = T)
  ytime <- apply(test.result, 1, function(x) which(x==1)[1])
  cens <- ifelse(is.na(ytime), 0, 1)
  dat <- data.frame(ytime, cens, Xmat)
  sim.out <- rfsrc(Surv(ytime, cens) ~ . , data = dat, mtry = round(sqrt(ncov)),
                  splitrule = "logrank", importance = "permute", ntree=n.tree,
                  forest=TRUE)
  vimp <- sim.out$importance
  return(vimp=vimp)
}

tnorm <- function(nsim, testtimes, sensitivity, specificity, noevent, beta, nbiomarker, n.tree){
  result <- mclapply(1:nsim, function(x) norm.err(testtimes, sensitivity, specificity,
                                                noevent, beta, nbiomarker, n.tree),
                    mc.cores=2)
  return(c(sensitivity, specificity, noevent, result))
}


scale <- seq(1, 0.5, by=-0.05)
pkg1 <- matrix(NA, nrow=11,ncol=4)
pkg2 <- matrix(NA, nrow=11,ncol=4)
pkg3 <- matrix(NA, nrow=11,ncol=4)
pkg4 <- matrix(NA, nrow=11,ncol=4)
pkg5 <- matrix(NA, nrow=11,ncol=4)
pkg6 <- matrix(NA, nrow=11,ncol=4)
```

```
for (i in 1:11){
  pkg1[i,] <-  tnorm(nsim=100, testtimes=1:4, sensitivity = 1, specificity=scale[i],
                     noevent=0.85, beta=2, nbiomarker=5, n.tree=1000)
  pkg2[i,] <-  tnorm(nsim=100, testtimes=1:4, sensitivity = 1, specificity=scale[i],
                     noevent=0.75, beta=2, nbiomarker=5, n.tree=1000)
  pkg3[i,] <-  tnorm(nsim=100, testtimes=1:4, sensitivity = 1, specificity=scale[i],
                     noevent=0.5, beta=2, nbiomarker=5, n.tree=1000)
}
for (i in 1:11){
  pkg4[i,] <-  tnorm(nsim=100, testtimes=1:4, sensitivity = scale[i], specificity=1,
                     noevent=0.85, beta=2, nbiomarker=5, n.tree=1000)
  pkg5[i,] <-  tnorm(nsim=100, testtimes=1:4, sensitivity = scale[i], specificity=1,
                     noevent=0.75, beta=2, nbiomarker=5, n.tree=1000)
  pkg6[i,] <-  tnorm(nsim=100, testtimes=1:4, sensitivity = scale[i], specificity=1,
                     noevent=0.5, beta=2, nbiomarker=5, n.tree=1000)
}

spec <- rep(seq(1, 0.5, -0.05), 3)
grp <- c(rep("15%",11),rep("25%",11),rep("50%",11))
dat <- cbind.data.frame(spec, c(pkg1[,4], pkg2[,4], pkg3[,4]), grp)
colnames(dat) <- c("Spec", "Proportion", "grp")
p = ggplot(data=dat, aes(x=spec, y=Proportion, group=grp)) +
  geom_line(aes(linetype=grp, color=grp)) +
  geom_point(aes(shape=grp, color=grp))+
  scale_x_reverse()+
  scale_y_continuous(limits=c(0, 1), breaks=seq(0,1, by=0.2))+theme_bw()
p1=ggpar(p,legend.title="Event Rate", xlab="Specificity",
         font.main=c("bold",12),palette = "jco", main="(a) Sensitivity=1", font.legend=8)
p2=p1+theme(legend.position=c(0.8,0.85))+theme(plot.title = element_text(hjust = 0.5))+
  rremove("grid")

sens <- rep(seq(1, 0.5, -0.05), 3)
dat1 <- cbind.data.frame(sens, c(pkg4[,4], pkg5[,4], pkg6[,4]), grp)
colnames(dat1) <- c("Sens", "Proportion", "grp")
q = ggplot(data=dat1, aes(x=Sens, y=Proportion, group=grp)) +
  geom_line(aes(linetype=grp, color=grp)) +
  geom_point(aes(shape=grp, color=grp))+
  scale_x_reverse()+
  scale_y_continuous(limits=c(0, 1), breaks=seq(0,1, by=0.2))+theme_bw()
q1=ggpar(q, font.main=c("bold",12),
         legend.title="Event Rate", xlab="Sensitivity",palette = "jco",
         main="(a) Specificity=1", font.legend=8)
q2=q1+theme(legend.position=c(0.8,0.85))+theme(plot.title = element_text(hjust = 0.5))+
  rremove("grid")
ggarrange(p2,q2, nrow=1, ncol=2)
```

5 References

1. Xu, H., Gu, X., Tadesse, M. G., Balasubramanian, R. (2018). A modified Random Survival Forests algorithm for variable selection in the presence of imperfect self-reports or laboratory based diagnostic tests, Journal of Computational and Graphical Statistics.