R implementation of variable selection, and estimation methodology, `Discnet`, for discrete survival time with frailty

Introduction

In this article, we show how `Discnet` can be implemented for variable selection, and estimation, of significant drivers (covariates) associated with discrete survival time, such as time-to-pregnancy, which is both left truncated and right censored. We begin by assigning the current working directory, and libraries required. The key R-function is `Discnet` which is defined in `Discnet.R`. All the auxilary R-programs required for this demo are given in `Preload.R` while the ones related to `Discnet` are provided in the library folder `ElasticLib`, both of which need to be kept in the same folder as that of `Discnet.R` and `demo.R`

```
source("Preload.R")  # Required for demo.R
sourceDir("./ElasticLib") # Required for Discnet.R
source("Discnet.R")
```

Generating right censorsored and left truncated discrete survival time with frailty

In this section, we follow the true model as described in Section 4 of the attached manuscript. The key function is *GenData* which can be called as follows,

$$\begin{aligned} function: \quad & GenData\ (n,\, p,\, minfreq,\, maxfreq,\, baseline,\, cencor.prob,\, rho.vec,\, block,\, frailty\_Q, \\ & Left.trun.prob,\, NZ,\, T.max) \end{aligned}$$

Arguments: First, we specify the design parameters; namely, sample size (n), number of covariates (p), minimum and maximum frequencies ($minfreq$, $maxfreq$) for survival time, multinomial parameters for the distribution of left truncation [Tr $\leq$ 3,Tr=1 means no truncation] ($left.trun.prob$), maximum survival time ($T.max$), censoring parameter ($censor.prob$) controlling proportion of overall right censoring, and correlation structure of block diagonal compound symmetry between covariates ($rho.vec$, $block$). Next, we specify the model parameters; vector of non-zero coefficients (NZ), baseline parameters ($baseline$), and frailty parameter ($fraily\_Q$)

Values: The function generates a list of objects including the key output file, discrete survival data ($data\_s$) along with other components inheriting simulation specifications such as true vector of coefficients ($beta$) etcs.

The following section of codes generates the data, and reports the observed right censoring and left truncation. Finally, data is split into two parts, 1. $data\_x$ having only covariates and "id" information, 2. $data\_y$ having survival outcome information. In the next section we discuss `Discnet` implementation.

```
rho.vec= c(0.7,0.4) # Two types of pairwise correlations used
block=c(3,2) # Two blocks of compound symmetry used with correlations mentioned as before
Lt_dist <-c(0.6,0.2,0.2) # Non-informative left truncation distribution, 1 indicates no truncation
n <- 150 # sample size
p <- 150 # Number of covariates
cp <- 0.017 # Censor probability parameter that leads to roughly 20% censoring in this setting
maxf <- 45 # maximum frequency  allowed for time-to-event
```

```r
baseline<-dgamma((1:10)-2, shape=5, scale = 1)*2- c(9,7,5,3,0,-1,-2,-4,-6,-8);
                     #Increasing baseline parameters with time
NZ= c(-4,-4,-4,8,8)
#Vector of TRUE NON-ZERO covariate coefficients starting from covariate 1 through the last NON-ZERO one
frailty_Q = 1 # frailty parameter: In this standard deviation of gaussian noise
T.max = 10 # number of baseline parameters
#==================Generating data from a discrete frailty model================
seed= 124;
set.seed(seed)
Simout <- GenData(n=n,p=p,minfreq=2, maxfreq = maxf,baseline=baseline,cencor.prob=cp,rho.vec= rho.vec
                  ,block=block,frailty_Q=frailty_Q,Left.trun.prob=Lt_dist,NZ= NZ, T.max=T.max)
```

```
## 7-th simulation: rejected
##  12-th simulation: rejected
##  12-th simulation: rejected
##  15-th simulation: rejected
##  19-th simulation: rejected
##  42-th simulation: rejected
##  46-th simulation: rejected
##  64-th simulation: rejected
##  75-th simulation: rejected
##  75-th simulation: rejected
##  75-th simulation: rejected
##  82-th simulation: rejected
##  82-th simulation: rejected
##  82-th simulation: rejected
##  87-th simulation: rejected
##  89-th simulation: rejected
##  103-th simulation: rejected
##  103-th simulation: rejected
##  116-th simulation: rejected
##  116-th simulation: rejected
##  127-th simulation: rejected
##  130-th simulation: rejected
##  133-th simulation: rejected
##  135-th simulation: rejected
##  144-th simulation: rejected
##  [1] "took 1 many steps"
```

```r
family= Simout$family
baseline=Simout$baseline
TrueTheta=Simout$beta
test.seq=Simout$test.seq

data_s <- Simout$data_s
Censor <- 1-mean(data_s$delta)
cat("Observed Censoring:  ", Censor, "\n")
```

```
## Observed Censoring:   0.2666667
```

```r
Trunc  <- mean(data_s$Lt >1)
cat("Observed left truncation:  ", Trunc, "\n")
```

```
## Observed left truncation:   0.3333333
```

```r
T.max <- max(data_s$time)
L.min <- min(data_s$time)
time.length <- (T.max - L.min+1)

outcome <- c("time", "delta", "Lt");
clus <- c("id") # id specifying subjects to include subject specific frailty
data_x  <- data_s[,c(clus,paste0("X.",1:p))]
data_y <- data_s[,outcome]
```

`Discnet` implementation

`Discnet` methodology requires one to specify the grids for tuning penalty parameter, α (of elastic net) and ν_s (of baseline parameters). Given a pair of $\alpha\,,\nu_s$ `Discnet` follows a path of increasing λ's up-to a maximum value, which leads to selection of no variables. Eventually, λ can be tuned using either permutation based approach or BIC based approach (See the references in the main manuscript). Once λ is tuned, $\alpha\,,\nu_s$ can be tuned based on BIC. Below we describe the function *Discnet* in more details,

$$
\begin{aligned}
function: \quad & Discnet\,(data\_y,\, data\_x,\, al,\, nus,\, index,\, Measure = c(\text{"}bic\text{"},\text{"}perm\text{"}),\, lambda.large = 200,\\
& init.theta = NULL,\, init.baseline = NULL,\, lambda.perm.length = 500,\\
& family = binomial(link = \text{"}logit\text{"}),\, quick = F,\, perm.parallel = T,\, nlambda = 100,\, clus = c(\text{"}id\text{"}))
\end{aligned}
$$

Arguments: First, we specify tuning parameters; namely, $\alpha\,(al)\,,\nu_s\,(nus)$ as mentioned earlier, and two data parts, involving survival outcomes and covariates respectively, namely $data\_y$, $data\_x$ as referred to in the earlier Section. $Measure$ option specifies λ selection method, which defaults to "bic". $lambda.large$ is an initial choice for λ to decide λ_{max} for generating path following mechanism. $lambda.perm.length$ specifies no. of permutations to be considered in case $Measure = \text{"}perm\text{"}$ is selected, whereas $nlambda$ specifies no. of λ's to be considered for the path. $init.theta$ and $init.baseline$ imply starting values for model parameters (defaults to 0's). $family$ indicates the discrete hazard link specification. $quick = T$ does not compute BIC based initial estimates for permutation method, and assigns all initial estimates of the β's to 0's. By default $perm.parallel$ is set to TRUE, which uses underlying available CPU's or threads to parallelize the computations across λ's over a path in case of permutation approach. Finally, $clus$ specifies the subject id variable (needs to be a factor).

Values: The function produces a list of objects, among which $coefficients$, $StdError$ and $baseline$ give respectively estimated coefficients for covariates, their standard errors (asymptotic) and estimated baseline parameters based on re-fitted model with the covariates selected by the optimized model. sum gives a brief summary on methods used, and likelihood based measures (such as bic). Further, $theta$, $time$, and Q, give 2 rows each, such that 2nd row has the final estimates whereas 1st row has initial estimates of the corresponding parameters respectively. $theta$ contains posterior mode type estimates of random effects in addition to coefficients for covariates.

The following section of codes finds, and lists the best model (corresponds to optimal λ_{α,ν_s}) for each pair of grids along with multiple information criteria.

```r
al.vec <- c(1,0.7,0.5) # grid for alpha (elastic net penalty)
nus.vec=c(15,50) # grid for nus, penalty parameters for baselines
ParamTab <- CrossParamTab(c( "alpha","nus"), list(alpha=al.vec,nus=nus.vec))
                        # lists two way grids for alpha and nu
Mes <- "perm" # method to choose lambda (elastic net penalty),
       # "bic" is the other alternative
lambda.large= 200  # An initial choice of  large lambda  (of elastic net) that forces
                   # parameter estimates to become 0
```

```r
index <- 1:p; # NA indicates non-penalization, same group variables have same index
# for example, from among V1,..V10 if only V1,V2 are not penalized; V3,V4,V5 are dummy codes
# for a categorical variable with 4 levels whereas rest are conts then index=c(NA,NA,3,3,3,4,5,6,7,8)
for(rowind in 1:nrow(ParamTab)){
al = ParamTab[rowind,"alpha"];
nus = ParamTab[rowind,"nus"]

print(paste0("seed=",seed,", al=",al,", nus=",nus))

Dspath <- Discnet(data_y, data_x, al=al, nus=nus, index=index,  Measure = Mes,
                  lambda.large= lambda.large, nlambda = 50)

if(rowind==1){Sum.tab <- cbind.data.frame(data.seed = seed, Dspath$sum)}else
{
  Sum.tab <- rbind.data.frame(Sum.tab, cbind.data.frame(data.seed = seed, Dspath$sum))
  }
}
```

```
## [1] "seed=124, al=1, nus=15"
## Requires libraries: ElasticLib, doParallel
##  Iteration  1
## Iteration  2
## [1] "seed=124, al=1, nus=50"
## Requires libraries: ElasticLib, doParallel
##  Iteration  1
## Iteration  2
## [1] "seed=124, al=0.7, nus=15"
## Requires libraries: ElasticLib, doParallel
##  Iteration  1
## Iteration  2
## [1] "seed=124, al=0.7, nus=50"
## Requires libraries: ElasticLib, doParallel
##  Iteration  1
## Iteration  2
## [1] "seed=124, al=0.5, nus=15"
## Requires libraries: ElasticLib, doParallel
##  Iteration  1
## Iteration  2
## [1] "seed=124, al=0.5, nus=50"
## Requires libraries: ElasticLib, doParallel
##  Iteration  1
## Iteration  2
```

```r
print(Sum.tab)
```

```
##   data.seed Method quick   lambda alpha nus      aic      bic    logLik
## 1       124   perm FALSE 26.42219   1.0  15 429.1415 477.1214 -204.3177
## 2       124   perm FALSE 25.53157   1.0  50 454.3839 495.8474 -218.3315
## 3       124   perm FALSE 37.74598   0.7  15 429.1515 477.1315 -204.3227
## 4       124   perm FALSE 36.47367   0.7  50 454.4255 495.8894 -218.3522
## 5       124   perm FALSE 52.84438   0.5  15 428.8772 481.5131 -203.1906
## 6       124   perm FALSE 51.06314   0.5  50 454.4091 495.8728 -218.3440
```

Summary of variable selection, and estimation

In what follows, optimized model is selected, and accuracy measures such as # of FNs, FPs and MSE are computed with respect to true model. Note that we have six grid pairs for demonstration purpose in the earlier section. One quick way to summarize the variable selection and parameter estimation is to use *summary* function. Note that the below optimized model does exact subset selection (FN=0, FP=0). Thus, it is enough to print first few values to obtain the parameter estimates of the final model.

```
tune_opt <- which.min(Sum.tab$bic)
Dspath <- Discnet(data_y, data_x, al=Sum.tab$alpha[tune_opt],nus=Sum.tab$nus[tune_opt],
                index=index, Measure = Mes, lambda.large= lambda.large)
```

```
## Requires libraries: ElasticLib, doParallel
##  Iteration  1
## Iteration  2
```

```
theta_est <- Dspath$theta[2,2:(p+1),drop=F] # First coefficient is the intercept
theta_T <- TrueTheta[11:(11+p-1)]
theta_err <- sum((theta_T-theta_est)^2)

#---------False Positives and False Negatives
epsilon <- 0
NZ =  1*( abs(theta_T) > epsilon)[1:p]
Pmat <- 1*(abs(theta_est) > epsilon)
FN <- sum(NZ)-Pmat %*% NZ
FP <-  Pmat %*% (!NZ)

Data.sum <-  cbind.data.frame(data.seed = seed,Censor=Censor,Trunc=Trunc,
             frailty=Dspath$Q[2],Dspath$sum, FN=FN,FP=FP,theta_err=theta_err)
print(Data.sum)
```

```
##   data.seed    Censor     Trunc   frailty Method quick   lambda alpha nus
## 1       124 0.2666667 0.3333333 0.1977741   perm FALSE 26.42219     1  15
##         aic      bic    logLik FN FP theta_err
## 1 429.2201 477.2013 -204.3568  0  0  99.93122
```

```
#======================= Summarizing the output=====
names(summary(Dspath))
```

```
## [1] "coefficients" "baseline.eff" "Q"
```

```
summary(Dspath)$coefficients[1:10,] # NA means non-selection of the covariate
```

```
##               Estimate    StdErr   z.value      p.value
## (Intercept) -0.9947088 0.2004060 -4.963467 6.924591e-07
## X.1         -0.9471973 0.3722536 -2.544495 1.094359e-02
## X.2         -1.1063194 0.4076449 -2.713929 6.649037e-03
## X.3         -1.0377942 0.3857531 -2.690307 7.138634e-03
## X.4          1.5558477 0.3592009  4.331414 1.481551e-05
## X.5          2.3487643 0.4064351  5.778941 7.517244e-09
## X.6          0.0000000        NA        NA           NA
## X.7          0.0000000        NA        NA           NA
## X.8          0.0000000        NA        NA           NA
## X.9          0.0000000        NA        NA           NA
```

```
summary(Dspath)$baseline.eff  #Baseline estimates
```

```
##     time.1     time.2     time.3     time.4     time.5     time.6     time.7
```

```
## -0.5932636 -0.4326752 -0.4144647 -0.2458654  0.3457549  0.1085829  0.0911842
##     time.8     time.9    time.10
##  0.1536331  0.3830317  0.6040822
```

```
summary(Dspath)$Q            #frailty estimates
```

```
## [1] 0.1977741
```