UKB analysis

Nir Keret

3/13/2023

First we load the UKB dataset, and the RCPP (make sure the files are in the same folder)

```
library(Rcpp)
library(survival)
library(psych)
UKB = read.csv("synthUKB.csv")
sourceCpp("cox-subsampling-reservoir.cpp", verbose=F)
```

Define some functions that will be needed later for the subsampling:

```
build_score = function(ret)
{
  tmp = ret$hess
  ret_hess = tmp + t(tmp)
  diag(ret_hess) = diag(tmp)
  ret[["hess"]] = -ret_hess
  return(ret)
}

information_score_matrix = function(beta,weights=NULL,times,truncs=NULL,status,covariates,
                                    samp_prob=NULL,information_mat=T,score_res_mat=T) {
  if(is.null(weights)) {weights = c(-1)}
  if(is.null(truncs)) {truncs = c(-1)}

  ret = rcpp_score_wrapper(beta,weights,times,truncs,status,covariates,
                           1*information_mat,1*score_res_mat)
  if(information_mat)
  {
    ret = build_score(ret)
  }

  if(is.null(samp_prob)) return(ret)
  if(samp_prob == "A")
  {
    ret[["samp_prob"]] = rcpp_A_OPT(ret[["residual"]],solve(ret[["hess"]]))
  }
  if(samp_prob == "L")
  {
    ret[["samp_prob"]] = rcpp_L_OPT(ret[["residual"]])
  }

  return(ret)
}
```

Preprocess the data: standardize the SNPs, define categorical variables as factors, remove observations whose observed time was before entry (standard Cox regression cannot handle this kind of "prevalent" observations)

```
SNPcols = which(colnames(UKB) %in% paste("SNP",1:139,sep="."))
UKB[,SNPcols] = scale(UKB[,SNPcols])

prev = which(UKB$V <= UKB$R)
UKB = UKB[-prev,]

UKB$alcohol = as.factor(UKB$alcohol)
UKB$breastfed = as.factor(UKB$breastfed)
UKB$education = as.factor(UKB$education)
UKB$snoring = as.factor(UKB$snoring)
UKB$chronotype = as.factor(UKB$chronotype)
UKB$balding = as.factor(UKB$balding)
UKB$physical_activity = as.factor(UKB$physical_activity)
```

Now we choose the number of censored (pseudo-)observations (denoted as q) per observed event, and the number of batches (denoted as K) for the reservoir-sampling algorithm. K should be chosen such that the RAM is sufficient upon expanding the batch into time-dependent form. q_0 is the number of censored observations sampled for the pilot uniform estimator.

```
cens_ind = which(UKB$delta == 0)
events_ind = which(UKB$delta == 1)

n_events = sum(UKB$delta)
n_cens = length(cens_ind)

K = 150

cens_groups = split(1:nrow(UKB),1:K)
q = 300  * n_events
q0 = 500000
```

Create the factors using the dietary information.

```
dietary_vars = which(colnames(UKB) %in% c("beef","cheese","poultry","lamb","pork",
"raw_veg","cooked_veg","processed_meat","oily_fish","non_oily_fish","fresh_fruit",
"dried_fruit","salt","tea","water","bread","cereal"))
UKB = cbind(matrix(NA,nrow(UKB),4),UKB)  #creating space for the factors
colnames(UKB)[1:4] = paste("PA",1:4,sep = "")

faUKB  = fa(r=UKB[,dietary_vars],
            nfactors = 4,
            fm="pa",
            max.iter=300,
            rotate="varimax")
UKB[,1:4] = faUKB$scores
```

Define the cut-points for expanding the data into time-dependent form. The time grid here is all observed events. Additionally, define the time-dependent function for sex $f(t) = \ln(t)I(t < A) + \ln(A)I(t \geq A)$. Keep all rows in the time-dependent dataset that correspond to the observed events.

```
cutpoints = sort(unique(UKB$V[UKB$delta == 1]))
time_func = function(x,A) ifelse(x<A,log(x),log(A))

events_TD = survSplit(Surv(R,V,delta) ~., data=UKB[events_ind,],cut=cutpoints)
```

```
events_TD = events_TD[events_TD$delta == 1,]
events_TD$sex_time = events_TD$sex * time_func(events_TD$V,65)
```

```
fmla = as.formula("Surv(R,V,delta,type = 'counting') ~ PA1 + PA2 + PA3 + PA4 + alcohol +
                  balding + BMI + breastfed + chronotype + diabetes + education +
                  family_history + hormones + drugs +physical_activity + screening +
                  sex_time + sleep + smoking + snoring + vitamin_D + miss_vitamin_D +
                  SNP.1 + SNP.2 + SNP.3 + SNP.4 + SNP.5 + SNP.6 +  SNP.7 + SNP.8 +
                  SNP.9 + SNP.10 + SNP.11 + SNP.12 + SNP.13 + SNP.14 + SNP.15 +
                  SNP.16 + SNP.17 + SNP.18 + SNP.19 + SNP.20 + SNP.21 + SNP.22 +
                  SNP.23 + SNP.24 + SNP.25 + SNP.26 + SNP.27 + SNP.28 + SNP.29 +
                  SNP.30 + SNP.31 + SNP.32 + SNP.33 + SNP.34 + SNP.35 + SNP.36 +
                  SNP.37 + SNP.38 + SNP.39 + SNP.40 + SNP.41 + SNP.42 + SNP.43 +
                  SNP.44 + SNP.45 + SNP.46 + SNP.47 + SNP.48 + SNP.49 + SNP.50 +
                  SNP.51 + SNP.52 + SNP.53 + SNP.54 + SNP.55 + SNP.56 + SNP.57 +
                  SNP.58 + SNP.59 + SNP.60 + SNP.61 + SNP.62 + SNP.63 + SNP.64 +
                  SNP.65 + SNP.66 + SNP.67 + SNP.68 + SNP.69 + SNP.70 + SNP.71 +
                  SNP.72 + SNP.73 + SNP.74 + SNP.75 + SNP.76 + SNP.77 + SNP.78 +
                  SNP.79 + SNP.80 + SNP.81 + SNP.82 + SNP.83 + SNP.84 + SNP.85 +
                  SNP.86 + SNP.87 + SNP.88 + SNP.89 + SNP.90 + SNP.91 + SNP.92 +
                  SNP.93 + SNP.94 + SNP.95 + SNP.96 + SNP.97 + SNP.98 + SNP.99 +
                  SNP.100 + SNP.101 + SNP.102 + SNP.103 + SNP.104 + SNP.105 +
                  SNP.106 + SNP.107 + SNP.108 + SNP.109 + SNP.110 + SNP.111 +
                  SNP.112 + SNP.113 + SNP.114 + SNP.115 + SNP.116 + SNP.117 +
                  SNP.118 + SNP.119 + SNP.120 + SNP.121 + SNP.122 + SNP.123 +
                  SNP.124 + SNP.125 + SNP.126 + SNP.127 + SNP.128 + SNP.129 +
                  SNP.130 + SNP.131 + SNP.132 + SNP.133 + SNP.134 + SNP.135 +
                  SNP.136 + SNP.137 + SNP.138 + SNP.139 + PC1 + PC2 + PC3 + PC4 +
                  PC5 + PC6")
```

Now we can estimate the regression coefficients using the subsampling procedures. Following is the respective analyses for the uniform/L-R-opt/A-R-opt methods. During the first batch ($m = 1$), we create the reservoir with q pseudo-observations and the event pseudo-observations. These are then potentially replaced by new pseudo-observations as more batches are processed.

Uniform

We first create the reservoir:

```
W_sum = 0

for(m in 1:K)
{
  ## Expanding to time-dependent dataset
  indices = cens_groups[[m]]
  timedep_dat = survSplit(Surv(R,V,delta) ~., data=UKB[indices,],cut=cutpoints)
  timedep_dat = timedep_dat[timedep_dat$delta == 0,]
  timedep_dat$sex_time = timedep_dat$sex * time_func(timedep_dat$V,65)

  if(m == 1)
  {
    subsample_U = rbind(timedep_dat[1:q,],events_TD)
  }
```

```
  W_batch = nrow(timedep_dat)
  W_sum = W_sum + W_batch

  # random sampling with uniform weights among censored
  qB = rbinom(1,q,W_batch / W_sum)

  samp_ind_unif = sample(1:W_batch,qB,replace = T)

  slots = sample(1:q,qB) #sample the slots to replace

  subsample_U[slots,] = timedep_dat[samp_ind_unif,] #updating the reservoir

  rm(timedep_dat)
  gc() #clear the memory
}
```

Now we can fit the model and derive the variance:

```
subsample_U$weights = c(rep(W_sum / q,q),rep(1,n_events))
fit_samp_unif = coxph(fmla,weights = weights,robust = F,data = subsample_U)
Ucoef = coef(fit_samp_unif)

#calculating the variance:
tmp_U2 = information_score_matrix(Ucoef,weights = subsample_U$weights,
                                  truncs = subsample_U$R, times = subsample_U$V,
                                  status = subsample_U$delta,
                                  covariates = model.matrix(fmla,data = subsample_U)[,-1])
ind_rm = (1:n_events)+q
order_rm = tmp_U2$ord[!(tmp_U2$ord %in% ind_rm)]
Score_U = tmp_U2$residual / (q*subsample_U$weights[1:q][order_rm])
phi_mat_U = cov(Score_U)
I_inv_U = solve(tmp_U2$hess)
var_U = I_inv_U + I_inv_U %*% phi_mat_U %*% I_inv_U/q

#remove objects to clear space
rm(fit_samp_unif)
rm(timedep_dat)
rm(subsample_U)
rm(Score_U)
gc()
```

L-opt

We first create the reservoir:

```
W_sum = 0
for(m in 1:K)
{
  ## Expanding to time-dependent dataset
  indices = cens_groups[[m]]
  timedep_dat = survSplit(Surv(R,V,delta) ~., data=UKB[indices,],cut=cutpoints)
  timedep_dat$sex_time = timedep_dat$sex * time_func(timedep_dat$V,65)
  timedep_dat = rbind(events_TD,timedep_dat[timedep_dat$delta == 0,])
```

```
  if (m == 1)
  {
    cens_timedp = (n_events+1):nrow(timedep_dat) #censored pseudo-observations row
    event_timedp = 1:n_events # events pseudo-observations rows
    unif_cont_ind = sample(cens_timedp,q0,replace = T)
    samp_ind_unif = c(unif_cont_ind,event_timedp)

    timedep_dat["weights"] = length(cens_timedp) * K / q0
    timedep_dat$weights[event_timedp] = 1

    #getting the pilot uniform estimate
    fit_samp_unif = coxph(fmla,weights = weights,data = timedep_dat
                          ,subset = samp_ind_unif,robust = F)
    Ucoef = coef(fit_samp_unif)

    rm(fit_samp_unif)
    gc()

    subsample_L = rbind(timedep_dat[1:q,],timedep_dat[1:n_events,])
  }

  timedep_dat$weights = K
  timedep_dat$weights[1:n_events] = 1

  tmp_L1 = information_score_matrix(Ucoef,times = timedep_dat$V,truncs = timedep_dat$R,
                                    weights = timedep_dat$weights,status = timedep_dat$delta,
                                    covariates = model.matrix(fmla,data = timedep_dat)[,-1],
                                    samp_prob = "L",information_mat = F)

# this is the sampling weight and not the sampling probability -
#(we need the unnormalized weight for the reservoir sampling):
  W_batch = sum(tmp_L1$samp_prob)
  W_sum = W_sum + W_batch

  n_cens = nrow(timedep_dat) - n_events
  delta_ord = timedep_dat$delta[tmp_L1$ord]
  cens_ind_ord = which(!delta_ord)

  # random sampling with L-optimal weights among censored
  qB = rbinom(1,q,W_batch / W_sum)

  samp_ind_censL = sample(1:n_cens,qB,replace = T,prob = tmp_L1$samp_prob)

  tmpsubsetL = timedep_dat[tmp_L1$ord[cens_ind_ord][samp_ind_censL],]
  tmpsubsetL$weights = 1 / (q * tmp_L1$samp_prob[samp_ind_censL])

  slots = sample(1:q,qB)

  subsample_L[slots,] = tmpsubsetL #updating the reservoir

  rm(tmpsubsetL)
  rm(tmp_L1)
  rm(timedep_dat)
```

```
  gc()
  print(m)
}
```

Now we can fit the model and derive the variance:

```
#now we can normalize the weights once we have W_sum of all observations:
subsample_L$weights[1:q] = subsample_L$weights[1:q] * W_sum

fit_samp_opt_L = coxph(fmla,weights = weights,robust = F,init = Ucoef,data = subsample_L)
Lcoef = coef(fit_samp_opt_L)

#calculating the variance
tmp_L2 = information_score_matrix(Lcoef,weights = subsample_L$weights,
                                  truncs = subsample_L$R,times = subsample_L$V,
                                  status = subsample_L$delta,
                                  covariates = model.matrix(fmla,data = subsample_L)[,-1])
ind_rm = (1:n_events)+q
order_rm = tmp_L2$ord[!(tmp_L2$ord %in% ind_rm)]
Score_L = tmp_L2$residual / (q*subsample_L$weights[1:q][order_rm])
phi_mat_L = cov(Score_L)
I_inv_L = solve(tmp_L2$hess)
var_opt_L = I_inv_L + I_inv_L %*% phi_mat_L %*% I_inv_L/q

rm(fit_samp_opt_L)
rm(timedep_dat)
rm(subsample_L)
rm(Score_L)
gc()
```

A-opt

We first create the reservoir:

```
W_sum = 0
for(m in 1:K)
{
  ## Expanding to time-dependent dataset
  indices = cens_groups[[m]]
  timedep_dat = survSplit(Surv(R,V,delta) ~., data=UKB[indices,],cut=cutpoints)
  timedep_dat$sex_time = timedep_dat$sex * time_func(timedep_dat$V,65)
  timedep_dat = rbind(events_TD,timedep_dat[timedep_dat$delta == 0,])


  cens_timedp = (n_events+1):nrow(timedep_dat)
  if (m == 1)
  {
    event_timedp = 1:n_events
    unif_cont_ind = sample(cens_timedp,q0,replace = T)
    samp_ind_unif = c(unif_cont_ind,event_timedp)

    timedep_dat["weights"] = length(cens_timedp) * K / q0
    timedep_dat$weights[event_timedp] = 1
```

```
    #getting the pilot uniform estimate
    fit_samp_unif = coxph(fmla,weights = weights,data = timedep_dat,
                          subset = samp_ind_unif,robust = F)
    Ucoef = coef(fit_samp_unif)

    # getting the I_inverse matrix to be used subsequently in all batches
    ret = information_score_matrix(Ucoef,times = timedep_dat$V,truncs = timedep_dat$R,
                                   weights = timedep_dat$weights,status = timedep_dat$delta,
                                   covariates = model.matrix(fmla,data = timedep_dat)[,-1],
                                   information_mat = T,score_res_mat = F)
    I_inv = solve(ret$hess)

    rm(ret)
    rm(fit_samp_unif)
    gc()

    subsample_A = rbind(timedep_dat[1:q,],timedep_dat[1:n_events,])
  }

  timedep_dat$weights = K
  timedep_dat$weights[1:n_events] = 1

  tmp_A1 = information_score_matrix(Ucoef,times = timedep_dat$V,truncs = timedep_dat$R,
                                    weights = timedep_dat$weights,status = timedep_dat$delta,
                                    covariates = model.matrix(fmla,data = timedep_dat)[,-1],
                                    samp_prob = "A",information_mat = F,I_inv = I_inv)

#this is actually the sampling weight and not the sampling probability -
#(we need the unnormalized weight for the reservoir sampling)
  W_batch = sum(tmp_A1$samp_prob)
  W_sum = W_sum + W_batch

  n_cens = nrow(timedep_dat) - n_events
  delta_ord = timedep_dat$delta[tmp_A1$ord]
  cens_ind_ord = which(!delta_ord)


  # random sampling with A-optimal weights among censored
  qB = rbinom(1,q,W_batch / W_sum)

  samp_ind_censA = sample(1:n_cens,qB,replace = T,prob = tmp_A1$samp_prob)

  tmpsubsetA = timedep_dat[tmp_A1$ord[cens_ind_ord][samp_ind_censA],]
  tmpsubsetA$weights = 1 / (q * tmp_A1$samp_prob[samp_ind_censA])

  slots = sample(1:q,qB)

  subsample_A[slots,] = tmpsubsetA #updating the reservoir

  rm(tmpsubsetA)
  rm(tmp_A1)
  rm(timedep_dat)
  gc()
```

```
  print(m)
}
```

Now we can fit the model and derive the variance:

```
subsample_A$weights[1:q] = subsample_A$weights[1:q] * W_sum

fit_samp_opt_A = coxph(fmla,weights = weights,robust = F,init = Ucoef,data = subsample_A)
Acoef = coef(fit_samp_opt_A)

#calculating the variance
tmp_A2 = information_score_matrix(Acoef,weights = subsample_A$weights,truncs = subsample_A$R,
                                  times = subsample_A$V,status = subsample_A$delta,
                                  covariates = model.matrix(fmla,data = subsample_A)[,-1])
ind_rm = (1:n_events)+q
order_rm = tmp_A2$ord[!(tmp_A2$ord %in% ind_rm)]
Score_A = tmp_A2$residual / (q*subsample_A$weights[1:q][order_rm])
phi_mat_A = cov(Score_A)
I_inv_A = solve(tmp_A2$hess)
var_opt_A = I_inv_A + I_inv_A %*% phi_mat_A %*% I_inv_A/q

rm(fit_samp_opt_A)
rm(timedep_dat)
rm(subsample_A)
rm(Score_A)
gc()
```