Uniformization example

2024-01-10

Data preparation

Load all needed packages.

Put all files in the same directory. Source all c++ and r function files.

```
sourceCpp("lik.cpp")
source("functions.R", local = knitr::knit_global())
```

We will generate data based on 3-state transition model that permits communication between all states.

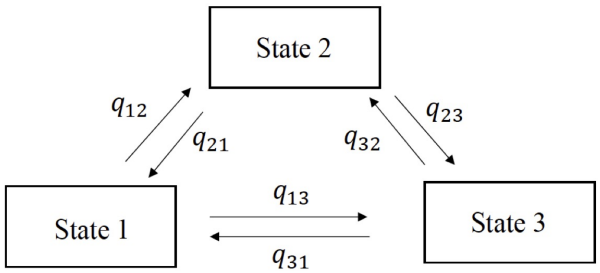

Figure 1: Running times under three sample size settings for 6 methods.

Single dataset example

Generate a dataset "generate_dt()".

```
store_all_data1=generate_dt(1)
head(store_all_data1[[1]])
```

```
##   subject time state keep
## 1       1   12     1    1
## 2       1   16     1    2
## 3       1   19     2    3
## 4       1   50     2    4
## 5       1   84     2    5
## 6       1   99     2    6
```

Transform data from long to wide format and get transition number.

```
sim.df1=store_all_data1[[1]]
sim.df1.wide=ddply(sim.df1,.(subject),msm_long_to_wide)
sim.df1.wide$time=sim.df1.wide$stop-sim.df1.wide$start

#assign transition number
sim.df1.wide$trans=1
sim.df1.wide$trans[sim.df1.wide$from==1&sim.df1.wide$to==1]=1
sim.df1.wide$trans[sim.df1.wide$from==1&sim.df1.wide$to==2]=2
sim.df1.wide$trans[sim.df1.wide$from==1&sim.df1.wide$to==3]=3
sim.df1.wide$trans[sim.df1.wide$from==2&sim.df1.wide$to==1]=4
sim.df1.wide$trans[sim.df1.wide$from==2&sim.df1.wide$to==2]=5
sim.df1.wide$trans[sim.df1.wide$from==2&sim.df1.wide$to==3]=6
sim.df1.wide$trans[sim.df1.wide$from==3&sim.df1.wide$to==1]=7
sim.df1.wide$trans[sim.df1.wide$from==3&sim.df1.wide$to==2]=8
sim.df1.wide$trans[sim.df1.wide$from==3&sim.df1.wide$to==3]=9
dataset1=cbind(1,sim.df1.wide[,c(6,7)])
```

The initial values θ_0 will be estimated roughly by assuming that there is no interval censoring and that all time points are recorded accurately.

```
theta0=c(0.02,0.01,0.01,0.05,0.01,0.03)
Q= Qmatrix(theta0[1],theta0[2],theta0[3],theta0[4],theta0[5],theta0[6])
iniQ=crudeinits.msm(state~time, subject = subject, data = sim.df1, qmatrix = Q)

#initial
theta0=round(c(iniQ[1,2],iniQ[1,3],iniQ[2,1],iniQ[2,3],iniQ[3,1],iniQ[3,2]),5)
```

Likelihood estimation (Ward approximation based on Nelder-Mead).

```
msm_ward_NM=optim(theta0,glike_ward,dataset2=dataset1,method = "Nelder-Mead",hessian = T,control=list(fn
#likelihood estimates
likes_ward_NM <- msm_ward_NM$par
#variance
likes_ward_NM.se <- diag(solve(msm_ward_NM$hessian))
```

Likelihood estimation (Ward approximation based on BFGS).

```
msm_ward_BFGS=optim(theta0,glike_ward,dataset2=dataset1,method = "Nelder-Mead",
                    hessian = T,control=list(fnscale=6000,maxit=1000))
#likelihood estimates
likes_ward_BFGS <- msm_ward_BFGS$par
#variance
likes_ward_BFGS.se <- diag(solve(msm_ward_BFGS$hessian))
```

Likelihood estimation (Canonical decomposition based on Nelder-Mead).

```
msm_CD_NM=optim(theta0,glike_decomp,dataset2=dataset1,method = "Nelder-Mead",
                hessian = T,control=list(fnscale=6000,maxit=1000))
#likelihood estimates
likes_CD_NM <- msm_CD_NM$par
#variance
likes_CD_NM.se <- diag(solve(msm_CD_NM$hessian))
```

Likelihood estimation (Canonical decomposition based on BFGS).

```
msm_CD_BFGS=optim(theta0,glike_decomp,dataset2=dataset1,method = "BFGS",
                  hessian = T,control=list(fnscale=6000,maxit=1000))
#likelihood estimates
likes_CD_BFGS <- msm_CD_BFGS$par
#variance
likes_CD_BFGS.se <- diag(solve(msm_CD_BFGS$hessian))
```

Likelihood estimation (Canonical decomposition based on Fisher).

```
Q= Qmatrix(theta0[1],theta0[2],theta0[3],theta0[4],theta0[5],theta0[6])

#Prevent code stuck
withTimeout(
  tryCatch(
    {
      CD_Fisher <- msm(state~time, subject = subject, data = sim.df1, qmatrix = Q,
                       opt.method="fisher",analyticp = FALSE,control=list(fnscale=6000,maxit=1000))
      p          <- CD_Fisher$estimates
      p.se       <- sqrt(diag(CD_Fisher$covmat))
      CD_Fisheres=CD_Fisher$estimates.t
      CD_Fisheres.se=(exp( p+1.96*p.se)-CD_Fisher$estimates.t)/1.96
    }
    , error = function(e) { skip_to_next <<- TRUE; rep(0,6)}), timeout = 60)
```

Likelihood estimation (Uniformization based on Nelder-Mead).

```
msm_unif_NM=optim(theta0,glike,dataset2=dataset1,method = "Nelder-Mead",
                  hessian = T,control=list(fnscale=6000,maxit=1000))
#likelihood estimates
likes_unif_NM <- msm_unif_NM$par
#variance
likes_unif_NM.se <- diag(solve(msm_unif_NM$hessian))
```

Likelihood estimation (Uniformization based on BFGS).

```
msm_unif_BFGS=optim(theta0,glike,dataset2=dataset1,method = "BFGS",
                    hessian = T,control=list(fnscale=6000,maxit=1000))
#likelihood estimates
likes_unif_BFGS <- msm_unif_BFGS$par
#variance
likes_unif_BFGS.se <- diag(solve(msm_unif_BFGS$hessian))
```

Likelihood estimation (Uniformization TBNR).

```
params1=theta0
jump_matrix=getallinten_second(params1,dataset1)

Unif_res_iter=likelihood_part2(as.matrix(jump_matrix))

#score vector
socrevec=Unif_res_iter[1:6]

#information matrix
  m_upper=Unif_res_iter[7:27]
  l <- length(m_upper)
  n <- length(socrevec)
  # Reconstruct
  m2 <- matrix(NA, n, n)
  m2[lower.tri(m2, diag = TRUE)] <- m_upper
  m2=t(m2)
  m2[lower.tri(m2)] <- t(m2)[lower.tri(m2)]  # If symmetric, fill also upper half
  infmatrix=m2

#N-R

  params2=solve(infmatrix,infmatrix %*%params1-socrevec)
  itra=1
  while  (sqrt(sum((params2-params1)^2))>0.0001) {
    params1=params2
    jump_matrix=getallinten_second(params1,dataset1)

    Unif_res_iter=likelihood_part2(as.matrix(jump_matrix ))

    socrevec=Unif_res_iter[1:6]

    m_upper=Unif_res_iter[7:27]

    l <- length(m_upper)
    n <- length(socrevec)
    # Reconstruct
    m2 <- matrix(NA, n, n)
    m2[lower.tri(m2, diag = TRUE)] <- m_upper
    m2=t(m2)
```

```
    m2[lower.tri(m2)] <- t(m2)[lower.tri(m2)]  # If symmetric, fill also upper half
    infmatrix=m2

    params2=solve(infmatrix,infmatrix %*%params1-socrevec)

    itra=itra+1
    print(itra)
  }
```

```
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
```

```
likes_unif_TBNR <-params2
#variance
likes_unif_TBNR.se <- diag(solve(infmatrix))
```

The results for all methods for one dataset are provided in the table below,

Parameter	True	WardNM	WardBFGS	CDNM	CDBFGS	CDFisher	UnifNM	UnifBFGS	UnifTBNR
q12	0.02	0.0228310	0.0228310	0.0228310	0.0227825	0.0227931	0.0228310	0.0227825	0.0227928
q13	0.01	0.0080967	0.0080967	0.0080967	0.0081681	0.0081353	0.0080967	0.0081681	0.0081356
q21	0.01	0.0086868	0.0086868	0.0086868	0.0086654	0.0086513	0.0086868	0.0086654	0.0086515
q23	0.05	0.0471221	0.0471221	0.0471221	0.0471625	0.0472033	0.0471221	0.0471625	0.0472016
q31	0.01	0.0135628	0.0135628	0.0135628	0.0135738	0.0135423	0.0135628	0.0135738	0.0135424
q32	0.03	0.0249257	0.0249257	0.0249257	0.0248987	0.0249165	0.0249257	0.0248987	0.0249165

Run simulation study with 100 replications.

In this part, "doparallel" package will be used.

First, detect number of cores.

```
numCores <- detectCores()
coreuse <- numCores-1
```

Here, we generate 500 datasets.

```
store_all_data1=generate_dt(500)
```

Run each method

Ward(NM)

```
start.time <- Sys.time()
result_ward <- mclapply(1:500, like_each_ward, mc.cores = coreuse)
end.time <- Sys.time()
time.takenward <- round(end.time - start.time,2)
time.takenward
```

```
## Time difference of 2.59 mins
```

```
WardNMres=apply(noquote(t(list.cbind(result_ward))), 2, as.numeric )
WardNMres=apply(WardNMres,2,function(x){ x[is.na(x)]=0
return(x)})

WardNMres_fail=sum(WardNMres[,1]==0)
WardNMres1=WardNMres[WardNMres[,1]!=0,]
WardNMres_mean=round(colMeans(WardNMres1[,1:6]),4)
```

Ward(BFGS)

```
start.time <- Sys.time()
result_ward1 <- mclapply(1:500, like_each_ward1, mc.cores = coreuse)
```

```
## Warning in mclapply(1:500, like_each_ward1, mc.cores = coreuse): scheduled
## cores 4, 1 encountered errors in user code, all values of the jobs will be
## affected
```

```
end.time <- Sys.time()
time.takenward1 <- round(end.time - start.time,2)
time.takenward1
```

```
## Time difference of 1.32 mins
```

```
WardBFGSres=apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric )
```

```
## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion
```

```
## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_ward1))), 2, as.numeric): NAs
## introduced by coercion
```

```
WardBFGSres=apply(WardBFGSres,2,function(x){ x[is.na(x)]=0
return(x)})

WardBFGSres_fail=sum(WardBFGSres[,1]==0)
WardBFGSres1=WardBFGSres[WardBFGSres[,1]!=0,]
WardBFGSres_mean=round(colMeans(WardBFGSres1[,1:6]),4)
```

Canonical decomposition(NM)

```
start.time <- Sys.time()
result_decomp <- mclapply(1:100, like_each_decomp, mc.cores = coreuse)
end.time <- Sys.time()
time.takendecomp <- round(end.time - start.time,2)
time.takendecomp
```

```
## Time difference of 59.15 secs
```

```
CDNMres=apply(noquote(t(list.cbind(result_decomp))), 2, as.numeric )
CDNMres=apply(CDNMres,2,function(x){ x[is.na(x)]=0
return(x)})

CDNMres_fail=sum(CDNMres[,1]==0)
CDNMres1=CDNMres[CDNMres[,1]!=0,]
CDNMres_mean=round(colMeans(CDNMres1[,1:6]),4)
```

Canonical decomposition(BFGS)

```
start.time <- Sys.time()
result_decomp1 <- mclapply(1:500, like_each_decomp1, mc.cores = coreuse)
```

```
## Warning in mclapply(1:500, like_each_decomp1, mc.cores = coreuse): scheduled
## cores 4, 1 encountered errors in user code, all values of the jobs will be
## affected
```

```
end.time <- Sys.time()
time.takendecomp1 <- round(end.time - start.time,2)
time.takendecomp1
```

```
## Time difference of 2.22 mins
```

```
CDBFGSres=apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric )
```

```
## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_decomp1))), 2, as.numeric): NAs
## introduced by coercion
```

```
CDBFGSres=apply(CDBFGSres,2,function(x){ x[is.na(x)]=0
return(x)})

CDBFGSres_fail=sum(CDBFGSres[,1]==0)
CDBFGSres1=CDBFGSres[CDBFGSres[,1]!=0,]
CDBFGSres_mean=round(colMeans(CDBFGSres1[,1:6]),4)
```

Canonical decomposition Fisher

```
start.time <- Sys.time()
result_fisher <- mclapply(1:500, like_each_fisher, mc.cores = 9)
end.time <- Sys.time()
time.takenFisher <- round(end.time - start.time,2)
time.takenFisher
```

```
## Time difference of 3.11 mins
```

```
CDFisher=apply(noquote(t(list.cbind(result_fisher))), 2, as.numeric )
CDFisher=apply(CDFisher,2,function(x){ x[is.na(x)]=0
return(x)})

#failure times
CDF_fail=sum(CDFisher[,1]==0)
CDFisher1=CDFisher[CDFisher[,1]!=0,]
CDFisherres_mean=round(colMeans(CDFisher1[,1:6]),4)
```

Uniformization(NM)

```
 start.time <- Sys.time()
 result_each <- mclapply(1:500, like_each, mc.cores = coreuse)
 end.time <- Sys.time()
 time.takenUnifor_NM <- round(end.time - start.time,2)
 time.takenUnifor_NM
```

```
## Time difference of 13.33 mins
```

```
UnifNMres=apply(noquote(t(list.cbind(result_each))), 2, as.numeric )
UnifNMres=apply(UnifNMres,2,function(x){ x[is.na(x)]=0
return(x)})

UnifNMres_fail=sum(UnifNMres[,1]==0)
UnifNMres1=UnifNMres[UnifNMres[,1]!=0,]
UnifNMres_mean=round(colMeans(UnifNMres1[,1:6]),4)
```

Uniformization(BFGS)

```r
start.time <- Sys.time()
result_each1 <- mclapply(1:500, like_each1, mc.cores = coreuse)
```

```
## Warning in mclapply(1:500, like_each1, mc.cores = coreuse): scheduled cores 4,
## 1 encountered errors in user code, all values of the jobs will be affected
```

```r
end.time <- Sys.time()
time.takenUnifor_BFGS <- round(end.time - start.time,2)
time.takenUnifor_BFGS
```

```
## Time difference of 7.46 mins
```

```r
UnifBFGSres=apply(noquote(t(list.cbind(result_each1))), 2, as.numeric )
```

```
## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion
```

```
## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion

## Warning in apply(noquote(t(list.cbind(result_each1))), 2, as.numeric): NAs
## introduced by coercion
```

```
 UnifBFGSres=apply(UnifBFGSres,2,function(x){ x[is.na(x)]=0
return(x)})

UnifBFGSres_fail=sum(UnifBFGSres[,1]==0)
UnifBFGSres1=UnifBFGSres[UnifBFGSres[,1]!=0,]
UnifBFGSres_mean=round(colMeans(UnifBFGSres1[,1:6]),4)
```

Uniformization TBNR

```
start.time <- Sys.time()
result_nr <- mclapply(1:500, like_each_nr, mc.cores = coreuse)
end.time <- Sys.time()
time.takenUnifor_TBNR <- round(end.time - start.time,2)
time.takenUnifor_TBNR
```

```
## Time difference of 2.62 mins
```

```
 UnifTBNRres=apply(noquote(t(list.cbind(result_nr))), 2, as.numeric )
 UnifTBNRres=apply(UnifTBNRres,2,function(x){ x[is.na(x)]=0
return(x)})

UnifTBNRres_fail=sum(UnifTBNRres[,1]==0)
UnifTBNRres1=UnifTBNRres[UnifTBNRres[,1]!=0,]
UnifTBNRres_mean=round(colMeans(UnifTBNRres1[,1:6]),4)
```

Mean of likelihood estimates of all methods.

Parameter	True	WdNM	WdBFGS	CDNM	CDBFGS	CDFisher	UnifNM	UnifBFGS	UnifTBNR
q12	0.02	0.0201	0.0201	0.0200	0.0201	0.0199	0.0201	0.0201	0.0201
q13	0.01	0.0101	0.0101	0.0101	0.0101	0.0196	0.0101	0.0101	0.0101
q21	0.01	0.0099	0.0099	0.0103	0.0099	0.0173	0.0099	0.0099	0.0099
q23	0.05	0.0503	0.0503	0.0499	0.0503	0.0504	0.0503	0.0503	0.0503
q31	0.01	0.0101	0.0101	0.0099	0.0101	0.0102	0.0101	0.0101	0.0101
q32	0.03	0.0301	0.0301	0.0304	0.0301	0.0367	0.0301	0.0301	0.0301

Running time and failure times

Method	Time	Failure
WDNM	155.40 secs	0
WDBFGS	79.20 secs	112
CDNM	59.15 secs	0
CDBFGS	133.20 secs	112
CDFisher	186.60 secs	109
UnifNM	799.80 secs	0

Method	Time	Failure
UnifBFGS	447.60 secs	112
UnifTBNR	157.20 secs	0