Simulation Code

library(EMtree)   
library(PLNmodels)  
library(SpiecEasi)  
library(MInt)  
  
library(ROCR)  
library(parallel)  
library(tidyverse)

# Data simulation

## Covariates

diff=c("easy","hard")  
path<-"~/Data"  
  
for( difficulty in diff){  
 n<-switch(difficulty,"easy"=100,"hard"=50)  
 nbspecies<-switch(difficulty,"easy"=20,"hard"=30)  
 covar<-data.frame(X1=round(runif(n)\*5),X2=rnorm(n,0,2), X3=(runif(n)))  
 saveRDS(covar,paste0(path,"/covar\_",difficulty,".rds"))  
}

## Counts

types<-c("cluster","scale-free","erdos")  
  
for( difficulty in diff){  
 nbspecies<-switch(difficulty,"easy"=20,"hard"=30)  
 for(type in types){  
 cat(type,difficulty,"\n")  
 obj<-mclapply(1:100,function(nbgraph){  
 cat("\n",nbgraph,":")  
 covar <- readRDS(paste0(path,"/covar\_",difficulty,".rds"))  
 dat <- data\_from\_scratch(type, p=nbspecies, r=10, covar=covar, signed=TRUE)  
 saveRDS(dat,paste0(path,"/Signed.Data\_",type,"\_",difficulty,nbgraph,".rds"))  
 }, mc.cores=1) # 1 can be changed to the wanted number of cores  
 }  
}

# Inference methods

dat<-readRDS(paste0(path,"TPFN/correct\_Data/Signed.Data\_",type,"\_",difficulty,nbgraph,".rds"))  
 Y<-dat[[1]]  
 edgesOrigin<-ifelse(abs(F\_Sym2Vec(dat[[2]]))<1e-16,0,1)  
 covar <- readRDS(paste0(path,"TPFN/correct\_Data/covar\_",difficulty,".rds"))  
 m<- model.matrix(~X1+X2+X3,covar)# l'intercept est enlevé dans le PLN de resample

## SpiecEasi

model<-lm(U~m)  
 T1<-Sys.time()  
 inf<-spiec.easi(model$residuals, method='mb', lambda.min.ratio=1e-3, nlambda=100,  
 pulsar.params=list(rep.num=20, thresh=0.1))  
 inf<-F\_Sym2Vec(as.matrix(inf$refit[[1]]))

## gCoda

inf <- F\_Sym2Vec(abs(gcoda(Y, counts=T, covar=covar, nlambda=100, lambda.min.ratio =1e-3)$opt.icov)>1e-16)\*1

## MInt

inf <- F\_Sym2Vec(eval\_store\_mint(Y,covar,path)>1e-16)\*1

## ecoCopula

my\_mod=manyglm(Y~m, family="negativ.binomial")  
 inf = F\_Sym2Vec(cgr(my\_mod, method="AIC")$best\_graph$graph)

## MRFcov

mrfres<- MRFcov(data = cbind(Y,m[,-1]), n\_nodes = p, family = 'poisson',   
 symmetrise = "mean",n\_cores=3, n\_covariates = 3)$graph  
 inf=F\_Sym2Vec(1\*(mrfres!=0))

## EMtree

resample<-ResampleEMtree(counts=Y, covar\_matrix=m, S=S, maxIter=200, cond.tol=cond.tol,cores=3)  
 if(length(resample)!=3) browser()  
 pmat<-resample$Pmat  
 # inf<- 1\*(ifelse(colSums(ifelse(pmat<2/p,0,1))/B >0.8,1,0))  
 inf<-F\_Sym2Vec(1\*(freq\_selec(pmat, Pt=2/p)>0.8))

TPFN\_compute<-function(methods,diffs,types, cores=3,B=100,S=10){  
 for(method in methods){  
 for( difficulty in diffs){#c(1e-3,1e-2,1e-1,1,10  
 # for(th in c(0.005,0.05,0.1,0.5)){  
 # for(S in c(2,5,10,20)){  
 path<-"/Users/raphaellemomal/simulations/Simu/PLN.2.0/"  
   
 cond.tol<-switch(difficulty,"easy"=1e-12,"hard"=1e-6)  
 for(type in types){  
 T1<-Sys.time()  
 cat(method,", ",difficulty,type,":")  
   
   
 obj<-mclapply(1:B,function(nbgraph){# attention signed data !! Signed.Data\_  
 cat("\ngraph ",nbgraph)  
 ########################  
 # récupérer les données et covariables associées générées précédemment, possibilités graphs signés  
 dat<-readRDS(paste0(path,"TPFN/correct\_Data/Signed.Data\_",type,"\_",difficulty,nbgraph,".rds"))  
 Y<-dat[[1]]  
 edgesOrigin<-ifelse(abs(F\_Sym2Vec(dat[[2]]))<1e-16,0,1)  
 covar <- readRDS(paste0(path,"TPFN/correct\_Data/covar\_",difficulty,".rds"))  
 m<- model.matrix(~X1+X2+X3,covar)# l'intercept est enlevé dans le PLN de resample  
 p=ncol(Y)  
 ########################  
 # inférence du réseau par les différentes méthodes  
 if(method=="MRFcov"){  
 T1<-Sys.time()  
 mrfres<- MRFcov(data = cbind(Y,m[,-1]), n\_nodes = p, family = 'poisson',   
 symmetrise = "mean",n\_cores=3, n\_covariates = 3)$graph  
 inf=F\_Sym2Vec(1\*(mrfres!=0))  
 T2<-Sys.time()  
 time<-difftime(T2,T1)  
 }  
 if(method=="MInt"){  
 T1<-Sys.time()  
 inf <- F\_Sym2Vec(eval\_store\_mint(Y,covar,path)>1e-16)\*1  
 T2<-Sys.time()  
 time<-difftime(T2,T1)  
 }  
 if(method=="EMtree"){  
 T1<-Sys.time()  
 resample<-ResampleEMtree(counts=Y, covar\_matrix=m, S=S, maxIter=200, cond.tol=cond.tol,cores=3)  
 if(length(resample)!=3) browser()  
 pmat<-resample$Pmat  
 # inf<- 1\*(ifelse(colSums(ifelse(pmat<2/p,0,1))/B >0.8,1,0))  
 inf<-F\_Sym2Vec(1\*(freq\_selec(pmat, Pt=2/p)>0.8))  
 T2<-Sys.time()  
 time<-difftime(T2,T1)  
 }  
 if(method=="gCoda"){  
 T1<-Sys.time()  
 inf <- F\_Sym2Vec(abs(gcoda(Y, counts=T, covar=covar, nlambda=100, lambda.min.ratio =1e-3)$opt.icov)>1e-16)\*1  
 T2<-Sys.time()  
 time<-difftime(T2,T1)  
 }  
 if(method=="SpiecEasi"){  
 U<-t(clr.matrix(Y,mar=1))  
 model<-lm(U~m)  
 T1<-Sys.time()  
 inf<-spiec.easi(model$residuals, method='mb', lambda.min.ratio=1e-3, nlambda=100,  
 pulsar.params=list(rep.num=20, thresh=0.1))  
 inf<-F\_Sym2Vec(as.matrix(inf$refit[[1]]))  
 T2<-Sys.time()  
 time<-difftime(T2,T1)  
 }  
 if(method=="ecoCopula"){  
 T1<-Sys.time()  
 my\_mod=manyglm(Y~m, family="negativ.binomial")  
 inf = F\_Sym2Vec(cgr(my\_mod, method="AIC")$best\_graph$graph)  
 T2<-Sys.time()  
 time<-difftime(T2,T1)  
 }  
 ########################  
 # comparaison de inf à edgesOrigin  
 tpfn<-c(table(inf,edgesOrigin))  
 res<-data.frame("TN"=tpfn[1],"FP"=tpfn[2],"FN"=tpfn[3],"TP"=tpfn[4] ,method=method,  
 times=time,unit=attr(time, "units"), type=type,difficulty=difficulty,  
 graph=nbgraph)  
 return(res)  
 },mc.cores=cores)  
 ########################  
 # obj est une liste de taille B qui contient les TPFN pour une méthode\*type de graph\*difficulté  
 res<-as\_tibble(do.call(rbind,obj))%>%  
 mutate(TN=as.numeric(as.character(TN)),  
 FN=as.numeric(as.character(FN)),  
 TP=as.numeric(as.character(TP)),  
 FP=as.numeric(as.character(FP)),  
 times=ifelse(unit=="mins",60\*as.numeric(as.character(times)),as.numeric(as.character(times))),  
 sum=TN+TP+FP+FN,FDR=FP/(TP+FP))  
   
 res<-res%>% mutate(FN=ifelse(is.na(sum),FP,FN),FP=ifelse(is.na(sum),0,FP),  
 TP=ifelse(is.na(sum),0,TP))  
 ########################  
 # res est un tibble qi contient pour B graphs d'un type\*difficulté, les TPFN corrigés, times et FDR pour une méthode  
 saveRDS(res, paste0(path,"/TPFN/Results/",method,"\_",type,"\_TFPN\_",difficulty,".rds"))  
 # saveRDS(res,paste0(path,"/TPFN/results/",method,"\_",type,"\_TFPN\_",difficulty,".rds"))  
 # saveRDS(lapply(obj, function(x){x[[2]]}), paste0(path,"/TPFN/results/Ghat",method,"\_",type,"\_TFPN\_",difficulty,".rds"))  
 T2<-Sys.time()  
 cat(difftime(T2,T1),attr(difftime(T2,T1), "units"),"\n")  
 # times<-c(times,difftime(T2,T1))  
 }  
   
 }  
 # }  
 }  
}