VK-syn.R

EvaSong

2024-04-15

Xfun <- function(n1, X, R, m) {  
 mydata <- mvrnorm(n1, m, R) # 多元正态分布生成模拟连续数据  
 X1 <- matrix(NA, nrow = n1, ncol = ncol(X)) # 初始化结果矩阵  
  
 for (i in 1:ncol(X)) {  
 n <- sum(X[, i]) # 第 i 列中变量为 1 的个数  
 p <- n / nrow(X) # 计算比例  
 xi <- rmultinom(n = n1, size = 1, prob = c(1 - p, p)) # 生成随机二元变量  
  
 # one-hot编码  
 vi <- matrix(NA, nrow = n1, ncol = 2)  
 vi[which(xi[1, ] == 1), 1] <- 0  
 vi[which(xi[2, ] == 1), 2] <- 1  
  
 # 将结果存储到 X1 中  
 X1 <- cbind(X1, vi)  
 }  
  
 # 将连续数据合并到结果矩阵中  
 df <- as.data.frame(cbind(mydata, X1))  
  
 # 对离散数据进行one-hot处理  
 X1 <- model.matrix(~ . - 1, df)  
  
 return(X1)  
}  
  
#########设置###########  
library("devtools")  
if(exists("cfcausal:::summary\_CI")){  
 rm(list = c("summary\_CI"))  
}  
devtools::load\_all(".")  
library("cfcausal")  
library("dplyr")  
library("ggplot2")  
library("bannerCommenter")  
library(MASS)  
library("h2o")  
h2o.init()  
options(scipen=999)  
#### Get parameters  
suppressPackageStartupMessages(library("argparse"))  
parser <- ArgumentParser()  
# parser$add\_argument("--gmm\_star", type = "double", default = 1.5, help = "SA parameter, >=1")  
parser$add\_argument("--alpha", type="double", default=0.2, help="miscoverage")  
parser$add\_argument("--cftype", type="integer", default=2, help="confounding type")  
parser$add\_argument("--fct", type="double", default=1, help="shrinkage, <=1")  
parser$add\_argument("--save", type="logical", default=TRUE, help="save")  
parser$add\_argument("--seed", type = "double", default = 1, help = "random seed")  
parser$add\_argument("--ntrial", type = "integer", default = 30, help = "number of trials,50")  
parser$add\_argument("--path", type = "character", default = './results/synthetic/VK\_huber/', help = "save location")  
parser$add\_argument("--ntrain", type = "integer", default = 2000, help = "training numbers,3000")  
parser$add\_argument("--ntest", type = "integer", default = 5000, help = "testing numbers,10000")  
# parser$add\_argument("--errdist", type = "character", default = 'heavy', help = "error distribution,norm,heavy,norm\_p")  
parser$add\_argument("--huber\_alpha", type = "integer", default = 0.1, help = "huber alpha, [0,1]")  
args <- parser$parse\_args()  
alpha <- args$alpha  
# gmm\_star <- args$gmm\_star  
cftype<- args$cftype  
fct <- args$fct  
ntrial<- args$ntrial  
seed <- args$seed  
save <- args$save  
# errdist <- args$errdist  
n1 <- args$ntrain # 训练集个数  
ntest <- args$ntest # 测试集个数  
path <- args$path  
huber\_alpha <- args$huber\_alpha  
q<- c(alpha/2, 1- (alpha/2))  
  
# 导入数据  
# 读取vk.csv  
vk <- read.csv("data/VK2.csv")  
vk$Gender <- ifelse(vk$Gender == "Male", 1, 0)  
vk$Access <- ifelse(vk$Access == "AVFistula", 1, 0)  
# 生成模型矩阵并赋值给mm  
mm <- model.matrix(~ HTN + DM + HCV + Smoking + Heartfailure + ISHD, data = vk)  
# 把mm数据框中的虚拟编码替换到vk数据框中  
col\_names <- colnames(mm)[-1]  
for (col in col\_names) {  
 vk[, col] <- mm[, col]  
}  
# 筛选处理组和控制组  
A <- as.numeric(vk$T == 1)  
# 定义协变量矩阵  
X <- vk[, c("Gender","HTNYes","DMYes" , "HCVYes","SmokingYes", "HeartfailureYes" ,"ISHDYes","Access","Age","Durationofdialysis", "PTH", "Ca.Pre","PHPre", "CaxPProductPre")]  
X1 <- model.matrix(~ . - 1, X)  
  
R <- cov(X1[,-(1:8)]) # 连续变量的样本协方差  
m <- colMeans(X1[,-(1:8)]) # 连续变量样本均值  
  
Xfun<-function(n1,X,R,m){#需要读入原始数据矩阵X和需要生成的样本个数n1  
 mydata<-mvrnorm(n1,m,R)#多元正态分布生成模拟连续数据  
 #第一列  
 n<-sum(X[,1])#第一列中变量为1的个数  
 p<-n/dim(X)[1]  
 x1<-rmultinom(n=n1, size=1, prob=c(1-p,p))  
 v1<-rep(NA,n1)  
 v1[which(x1[1,]==1)]<-0  
 v1[which(x1[2,]==1)]<-1  
 #第二列  
 n<-sum(X[,2])#第二列中变量为1的个数  
 p<-n/dim(X)[1]  
 x2<-rmultinom(n=n1, size=1, prob=c(1-p,p))  
 v2<-rep(NA,n1)  
 v2[which(x2[1,]==1)]<-0  
 v2[which(x2[2,]==1)]<-1  
 #第三列  
 n<-sum(X[,3])#第三列中变量为1的个数  
 p<-n/dim(X)[1]  
 x3<-rmultinom(n=n1, size=1, prob=c(1-p,p))  
 v3<-rep(NA,n1)  
 v3[which(x3[1,]==1)]<-0  
 v3[which(x3[2,]==1)]<-1  
 #第四列  
 n<-sum(X[,4])#第四列中变量为1的个数  
 p<-n/dim(X)[1]  
 x4<-rmultinom(n=n1, size=1, prob=c(1-p,p))  
 v4<-rep(NA,n1)  
 v4[which(x4[1,]==1)]<-0  
 v4[which(x4[2,]==1)]<-1  
 #第五列  
 n<-sum(X[,5])#第五列中变量为1的个数  
 p<-n/dim(X)[1]  
 x5<-rmultinom(n=n1, size=1, prob=c(1-p,p))  
 v5<-rep(NA,n1)  
 v5[which(x5[1,]==1)]<-0  
 v5[which(x5[2,]==1)]<-1  
 #第六列  
 n<-sum(X[,6])#第六列中变量为1的个数  
 p<-n/dim(X)[1]  
 x6<-rmultinom(n=n1, size=1, prob=c(1-p,p))  
 v6<-rep(NA,n1)  
 v6[which(x6[1,]==1)]<-0  
 v6[which(x6[2,]==1)]<-1  
 #第七列  
 n<-sum(X[,7])#第七列中变量为1的个数  
 p<-n/dim(X)[1]  
 x7<-rmultinom(n=n1, size=1, prob=c(1-p,p))  
 v7<-rep(NA,n1)  
 v7[which(x7[1,]==1)]<-0  
 v7[which(x7[2,]==1)]<-1  
 #第8列  
 n<-sum(X[,8])#第八列中变量为1的个数  
 p<-n/dim(X)[1]  
 x8<-rmultinom(n=n1, size=1, prob=c(1-p,p))  
 v8<-rep(NA,n1)  
 v8[which(x8[1,]==1)]<-0  
 v8[which(x8[2,]==1)]<-1  
 #离散和连续数据合并形成最终的模拟数据,离散数据做one hot 处理  
 V<-as.matrix(mydata)  
 df<-cbind(V,v1,v2,v3,v4,v5,v6,v7,v8)  
 df<-as.data.frame(df)  
 X1<-model.matrix(~ .-1,df )  
 return(X1)  
}  
  
sdfun <- function(X){  
 rep(1, nrow(X))  
}  
  
taufun <- function(X){  
 2 / (1 + exp(-5 \* (0.01\*X[, 1] - 0.5))) \* 2 / (1 + exp(-5 \* (0.01\*X[, 2] - 0.5)))  
}  
  
pscorefun <- function(X){  
 x1<-X[, 1]  
 x1<-(x1-min(x1))/(max(x1)-min(x1))  
 (1 + pbeta(1-x1, 2, 4)) / 4  
}  
  
get\_Y1obs <- function(X){  
 if(errdist=='norm'){  
 return(taufun(X) + sdfun(X) \* rnorm(dim(X)[1],0,400))  
 }else if(errdist=='heavy'){  
 return(taufun(X) + sdfun(X) \* rlogis(dim(X)[1], -99.03384, 326.54018))  
 }else if(errdist=='norm\_p'){  
 y <- rnorm(dim(X)[1],0,400)  
 sam <- sample(1:dim(X)[1], 0.1\*dim(X)[1])  
 y[sam] = y[sam] + 1500  
 return(taufun(X) + sdfun(X) \* y)  
 }  
}  
  
taufun0 <- function(X){  
 taufun(X) + 10\*sin(X[, 3])\*(1/(1+exp(-0.05\*X[, 3])))  
}  
  
get\_Y0obs <- function(X){  
 if(errdist=='norm'){  
 return(taufun0(X) + sdfun(X) \* rnorm(dim(X)[1],0,30))  
 }else if(errdist=='heavy'){  
 return(taufun0(X) + sdfun(X) \* rlogis(dim(X)[1], -0.524069, 15.708925))  
 }else if(errdist=='norm\_p'){  
 return(taufun(X) + sdfun(X) \* rnorm(dim(X)[1],0,30))  
 }  
}  
  
shrink <- function(set,fc){  
 newset <- set  
 idx <- is.finite(set[,1])  
 center <- (set[idx,2] + set[idx,1])/2  
 halflen <- (set[idx,2] - set[idx,1])/2  
 newset[idx,] <- cbind(center-halflen\*fc, center+halflen\*fc)  
 return(newset)  
}  
  
print\_list <- list("sa\_huber", "sa\_mean", "sa\_cqr", "ite\_nuc", "sa\_naive")  
record <- replicate(length(print\_list),matrix(0,nrow=ntrial,ncol=3), simplify=FALSE)  
  
gmm\_values <- seq(1, 3, by = 0.5) # 创建一个从1到3的序列，步长为0.5  
errdist\_values <- c("norm", "heavy", "norm\_p") # errdist的值数组  
  
for(errdist in errdist\_values){  
 for(gmm\_star in gmm\_values){  
  
 #create a new path for files  
 if(fct <1){  
 folder <- paste0(path,"/", errdist, "/", "fct\_", fct,"/") #忽略，我们取fct==1  
 }else{  
 folder<- paste0(path, "/", errdist, "/", "gmm\_",gmm\_star,"/")  
 }  
 print(folder)  
 dir.create(folder, recursive=TRUE, showWarnings = FALSE)  
  
 for (trial in 1:ntrial){  
 ##---------------------------------------------------------------  
 ## Generate observed data -  
 ##---------------------------------------------------------------  
 X<-Xfun(n1,X1,R,m)  
 n<-nrow(X)  
 ps <- pscorefun(X)  
 T <- as.numeric(runif(n)<ps)  
  
 Y0 <- get\_Y0obs(X)  
 Y1 <- get\_Y1obs(X)  
  
 Y\_obs <- Y1\*T + Y0\*(1-T)  
  
 Y1[which(T==0)] <- NA  
 Y0[which(T==1)] <- NA  
  
 ##----------------------------------------------------------------  
 ## bonferroni -  
 ##----------------------------------------------------------------  
  
 obj1\_ite <- conformal\_SA(X, Y1, gmm\_star, type = "mean", outfun='RF')  
 obj0\_ite <- conformal\_SA(X, Y0, gmm\_star, type = "mean", outfun='RF')  
  
 ##----------------------------------------------------------------  
 ## inexact ite method assuming no unobserved confounder -  
 ##----------------------------------------------------------------  
  
 CIfun\_inexact <- conformalIte(X, Y\_obs, T, alpha = alpha,  
 algo = "nest", exact=FALSE, type = "CQR",  
 #lofun = 'RF', upfun = 'RF', citype = "mean",  
 quantiles = c(alpha/2, 1- (alpha/2)), outfun = "quantRF", useCV = FALSE)  
  
 ##----------------------------------------------------------------  
 ## Train on Group1  
 ##----------------------------------------------------------------  
 obj\_mean <- nested\_conformalSA(X, Y1, Y0, T, gmm\_star, type = "mean",quantiles=list(), outfun='RF')  
 obj\_cqr <- nested\_conformalSA(X, Y1, Y0, T, gmm\_star, type = "CQR",quantiles=q, outfun='quantRF')  
 obj\_huber <- nested\_conformalSA(X, Y1, Y0, T\_obs, gmm\_star, type = "mean", quantiles=list(), outfun='huberBoosting', outparams=list(huber\_alpha = huber\_alpha))  
 ##先把数据集随机分成两组，在其中一组中使用算法一，算法一分别对treated组和control组做，记录下参数与拟合的函数  
 ##----------------------------------------------------------------  
 ## getting prediction bands on Group2  
 ##----------------------------------------------------------------  
 obj\_bands\_mean <- predict.nested(obj\_mean, X, Y\_obs, T, alpha = alpha)  
 obj\_bands\_cqr <- predict.nested(obj\_cqr, X, Y\_obs, T, alpha = alpha)  
 obj\_bands\_huber <- predict.nested(obj\_huber, X, Y\_obs, T, alpha = alpha)  
  
 ##对于另外一组，使用另外一组做出预测区间，并检查预测区间的覆盖率。也就是stepII的步骤2.4  
  
 ##----------------------------------------------------------------  
 ## generate testing data -  
 ##----------------------------------------------------------------  
  
 ##Testing  
 Xtest <- Xfun(ntest,X1,R,m)  
 pstest <- pscorefun(Xtest)  
 Ttest <- as.numeric(runif(ntest)<pstest)  
 id1 <- which(Ttest==1)  
 id0 <- which(Ttest==0)  
  
 Y0test <- rep(NA,ntest)  
 Y0test[id0] <- get\_Y0obs(Xtest[id0,])  
 Y0test[id1] <- samplecf(Xtest[id1,], taufun0, sdfun, case=cftype, gmm=gmm\_star)  
  
 #这里用的是case2，也就是appendix C里的方法产生反事实预测Y0  
 Y1test <- rep(NA,ntest)  
 Y1test[id1] <- get\_Y1obs(Xtest[id1,])  
 Y1test[id0] <- samplecf(Xtest[id0,],taufun, sdfun, case=cftype, gmm=gmm\_star)  
  
 ##----------------------------------------------------------------  
 ## ITE & evaluation -  
 ##----------------------------------------------------------------  
  
 ite <- Y1test - Y0test  
  
 ci\_mean\_copy <-fit\_and\_predict\_band(obj\_bands\_mean,Xtest, 'quantRF')  
 ci\_mean <-shrink(ci\_mean\_copy, fc=fct)#csq-m  
 ci\_mean[, 3] <- ci\_mean\_copy[,3]  
 ci\_mean[, 4] <- ci\_mean\_copy[, 4]  
  
 ci\_cqr\_copy <- fit\_and\_predict\_band(obj\_bands\_cqr,Xtest, 'quantRF')  
 ci\_cqr <- shrink(ci\_cqr\_copy, fc=fct)#csa-q  
 ci\_cqr[, 3] <- ci\_cqr\_copy[,3]  
 ci\_cqr[, 4] <- ci\_cqr\_copy[, 4]  
  
 #CSA-huber  
 ci\_huber\_copy <- fit\_and\_predict\_band(obj\_bands\_huber, Xtest,'quantRF')  
 ci\_huber <- shrink(ci\_huber\_copy, fc=fct)  
 ci\_huber[, 3] <- ci\_huber[,3]  
 ci\_huber[, 4] <- ci\_huber[, 4]  
  
 #bonferroni  
 ci0\_ite <- predict.conformalmsm(obj0\_ite, Xtest,alpha = alpha/2)  
 ci1\_ite <- predict.conformalmsm(obj1\_ite, Xtest,alpha = alpha/2)  
 ci\_ite <- cbind(ci1\_ite[,1] - ci0\_ite[,2], ci1\_ite[,2] - ci0\_ite[,1])  
  
 #ite-nuc  
 ci\_inexact <- CIfun\_inexact(Xtest)  
  
 ## 最后得到四组结果：CSA-huber, CSA-M, CSA-Q, ITE-NUC, BART  
 ci\_list <- list(ci\_huber, ci\_mean, ci\_cqr, ci\_inexact, ci\_ite)  
  
 ##打印区间  
 data <- cbind(ci\_huber, ci\_mean, ci\_cqr, ci\_inexact, ci\_ite)  
 colnames(data) <- c(  
 "huber\_low", "huber\_high", "huber\_y1\_mean","huber\_y0\_mean",  
 "mean\_low", "mean\_high", "mean\_y1\_mean","mean\_y0\_mean",  
 "cqr\_low", "cqr\_high","cqr\_y1\_mean","cqr\_y0\_mean",  
 "nuc\_low", "nuc\_high",  
 "bart\_low", "bart\_high")  
 df <- as.data.frame(t(data))  
 write.csv(data, file=paste0(folder, 'ntrial\_', trial, '.csv'))  
  
 for(i in 1:length(ci\_list)){  
 #保形区间  
 ci <- ci\_list[[i]]  
 #区间长度  
 diff <- ci[, 2] - ci[, 1]  
 # 找出符合条件的索引  
 index <- which(diff > 9999999) # 人为设定  
 # 将符合条件的值修改为Inf  
 ci[index, 2] <- Inf # 人为修改  
 ci[index, 1] <- Inf # 人为修改  
 diff[index] <- Inf # 人为修改  
  
 #覆盖率  
 coverage <- mean((ite >= ci[, 1]) & (ite <= ci[, 2]),na.rm = TRUE)  
 #平均区间长度（有限值）  
 len <- mean(diff[is.finite(diff)])  
 #无限长度的区间个数  
 n\_inf <- sum(is.infinite(diff))  
  
 #输出  
 print(paste0(print\_list[i], " coverage, ",coverage, ', lens ', len))  
 #第i组，第trial次实验的：覆盖率、区间长度、差值（区间长度）是否有限  
 record[[i]][trial,] <- c(coverage,len,n\_inf)  
 }  
 print(paste0("################# trial ",trial," #################"))  
 }  
 ##----------------------------------------------------------------  
 ## Save results --  
 ##----------------------------------------------------------------  
  
 #coverage data  
 coverage <-c()  
 for (i in 1:length(print\_list)){coverage[[i]]<- as.vector(record[[i]][,1])}  
  
 data <- data.frame(Coverage=unlist(coverage),  
 group=rep(c("CSA-huber", "CSA-M", "CSA-Q", "ITE-NUC", "CSA-B"),  
 each=ntrial))  
  
 if(save){  
 write.csv(data, paste0(folder,'coverage', '.csv'), row.names = FALSE)  
 }  
  
 #length data  
 Interval\_length <-c()  
 for (i in 1:length(print\_list)){Interval\_length[[i]]<- as.vector(record[[i]][,2])}  
 data <- data.frame(Interval\_length= unlist(Interval\_length),  
 group=rep(c("CSA-huber", "CSA-M","CSA-Q", "ITE-NUC", "CSA-B"),each=ntrial))  
  
 if(save){  
 write.csv(data, paste0(folder,'len','.csv'), row.names = FALSE)  
 }  
 }  
}