Reproducible code for STA640 final project

Yi Chen

2024-05-03

Load required packages

```
#install.packages('gsynth')
#install.packages('Synth')
library(gsynth)
```

```
## ## Syntax has been updated since v.1.2.0.
```

```
## ## Comments and suggestions -> yiqingxu@stanford.edu.
```

```
library(Synth)
```

```
## ##
## ## Synth Package: Implements Synthetic Control Methods.
```

```
## ## See https://web.stanford.edu/~jhain/synthpage.html for additional information.
```

```
require(parallel)
```

```
## Loading required package: parallel
```

```
require(foreach)
```

```
## Loading required package: foreach
```

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(GGally)
```

```
## Loading required package: GGally
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
set.seed(123)
```

Set work directory and load the factors and factor loadings

Please change it accordingly.

```
rm(list=ls())
setwd('/Users/yi/Library/CloudStorage/Dropbox/Duke/2024 spring/causal inference/final project')
## fixed factors and loadings
load("FLSource.RData")
```

Data generation Process

I start with the following data generating process (DGP)

$$Y_{it} = \alpha_{it} D_{it} + 1 \cdot x_{it,1} + 3 \cdot x_{it,2} + \lambda_t \mu_i + \eta_i + \xi_t + 5 + \epsilon_{it}$$

```
# The simulation function is referred to the code provided by Xu (2017)

simulate<-function(Ntr, Nco, T0, p, r, m=0, w=1, D.sd=1, att=c(1:10),
                   beta=NULL, mu=0, fixF=FALSE, fixL=FALSE, fsize=1,
                   FE=0, seed=NULL,unif=FALSE,AR1=0) {

  ## p, number of covariates (p=0, no covariates)
  ## Ntr, Nco, N
  ## T0, T
  ## r
  ## m: number of Z
  ## overlap w = [0,1], w==1 means complete overlap
  ## D.sd: heterogeneity
  ## beta: true coefficients for X
  ## att: average treatment effect
  ## fsize: relative importance of factors vs. covariates
  ## trend: importance of a linear time trend
  ## FE: to include unit and time fixed effects
  ## fixF: factors as given
  ## fixL: loadings as given
  ## AR1: Autoregressive(1) coefficient

  if (is.null(seed)==FALSE) {
    set.seed(seed)
  }
  N<-Ntr+Nco
  T<-T0+length(att)


  Tr<-1:Ntr   # treatment
  Co<-(Ntr+1):N # control
  pre<-1:T0
  pst<-c(1:T)[-pre]

  ## ###########################
  ## Data generating process
  ## ###########################

  rr<-m + r # observed and unobserved

  ## loadings: get 10 (for the construction of X), use the first 1:r columns
  ss<-sqrt(3) # to ensure variance =1
  if (rr > 0) {
    if (fixL==FALSE) {
      lambda<-matrix(runif(N*rr,min=-ss,max=ss),N,rr)  # loadings
    } else {
      lambda<-L.source[c(c(1:Ntr),c(501:(500+Nco))),1:rr]
    }
```

```r
  lambda[1:Ntr,] <- lambda[1:Ntr,]+(1-w)*2*ss # overlap of loadings, w determines the shift of the un
}

## factors
if (fixF==FALSE) { # F not given
  factor<-matrix(rnorm(T*rr),T,rr)   # factors
} else {
  if (unif==FALSE) {
    factor<-F.source[(dim(F.source)[1]-T+1):dim(F.source)[1],1:rr]
    # always use the last T rows, first rr columns
  }  else { # uniform distribution
    factor<-F.u.source[(dim(F.u.source)[1]-T+1):dim(F.u.source)[1],1:rr]
  }
}

## fixed effects
if (FE==1) {
  ## unit fixed effects
  if (fixL==FALSE) {
    alpha<-runif(N,min=-ss,max=ss)
  } else {
    alpha <- L.source[c(c(1:Ntr),c(501:(500+Nco))),20] # the last column
  }
  alpha[1:Ntr]<-alpha[1:Ntr]+(1-w)*2*ss
  ## time fixed effects
  if (fixF==FALSE) {
    xi<-rnorm(T,0,1)
  } else {
    if (unif==FALSE) {
      xi<-F.source[(dim(F.source)[1]-T+1):dim(F.source)[1],20] # the 20th (last) column
    }  else {
      xi<-F.u.source[(dim(F.u.source)[1]-T+1):dim(F.u.source)[1],20] # the 20th (last) column
    }

  }
}

## error
e <- matrix(rnorm(T*N),T,N) # disturbances

## time varying covariates: always generated by the same first two factors
truebeta<-beta
if (p!=0) {
  X<-array(0,dim=c(T,N,p))    # regressor matrix, must be T by N  by  p
  for (j in 1:p) {
    X[,,j] <- matrix(rnorm(T*N),T,N)  +
      0.5 * factor[,1:2] %*% t(lambda[,1:2]) +
      0.25 * matrix(1,T,2) %*% t(lambda[,1:2]) +
      0.25 * factor[,1:2] %*% matrix(1,2,N)+1
  }
}

## treatment assignment
```

```
D<-cbind(rbind(matrix(0,T0,Ntr),matrix(1,(T-T0),Ntr)),matrix(0,T,Nco))

## the treatment effect
eff<-matrix(c(rep(0,T0),att),T,N)+rbind(matrix(0,T0,N),matrix(rnorm((T-T0)*N,0,D.sd),(T-T0),N))

## outcome variable
Y0<- e  +  matrix(mu,T,N) # error + grand mean
if (r>0) {
  Y0 <- Y0 + fsize*factor%*%t(lambda)
}
if (FE==1) {
  Y0 <- Y0 + matrix(alpha,T,N,byrow=TRUE) + matrix(xi,T,N,byrow=FALSE)
}
if (p!=0) { # covariates
  for (k in 1:p) {
    Y0<-Y0+X[,,k]*truebeta[k]
  }
}
Y1 <- Y0 + eff
if (AR1>0) {
  for (t in 2:T) {
    Y0[t,]<-Y0[(t-1),]*AR1 + Y0[t,]
    Y1[t,]<-Y1[(t-1),]*AR1 + Y1[t,]
  }
  eff.acc<-Y1-Y0 # accumulative effect, T*N matrix
}
Y<-(matrix(1,T,N)-D)*Y0+D*Y1
if (AR1>0) {
  Y.lag<-matrix(NA,T,N); Y.lag[2:T,]<-Y[1:(T-1),]
}

## substract error
Y.bar <- Y - e

## panel structure
panel<-as.data.frame(cbind(rep(101:(100+N),each=T),rep(1:T,N),
                           c(Y),c(Y0),c(Y1),c(D),c(eff),c(e),c(Y.bar),
                           rep(mu,T*N),rep(1,T*N),
                           c(rep(1,T*Ntr),rep(0,T*Nco))))
cname<-c("id","time","Y","Y0","Y1","D","eff","error","Ybar","mu","X0","treat")
if (p!=0) {
  for (i in 1:p) {
    panel<-cbind(panel,c(X[,,i]))
    cname<-c(cname,paste("X",i,sep=""))
  }
}
if (rr > 0) {
  for (i in 1:rr) {
    panel<-cbind(panel,rep(factor[,i],N))
    cname<-c(cname,paste("F",i,sep=""))
  }
}
if (m > 0) {
```

```r
    for (i in 1:m) {
      panel<-cbind(panel,rep(lambda[,i],each=T))
      cname<-c(cname,paste("Z",i,sep=""))
    }
  }
  if (r > 0) {
    for (i in 1:r) {
      panel<-cbind(panel,rep(lambda[,(m+i)],each=T))
      cname<-c(cname,paste("L",i,sep=""))
    }
  }
  if (FE==1) {
    panel<-cbind(panel,rep(alpha,each=T),rep(xi,N))
    cname<-c(cname,"alpha","xi")
  }
  if (AR1>0) {
    panel<-cbind(panel,c(Y.lag),c(eff.acc))
    cname<-c(cname,"Y_lag","eff_acc")
  }
  colnames(panel)<-cname
  return(panel)
}
```

For simplicity, specify only one treated unit and 40 potential control units. Let $T_0 = 20$, $T = 30$.

```r
Ntr = 1
Nco = 40
T0 = 20
simdata<-simulate(Ntr=Ntr,Nco=Nco,T0=T0,p=2,r=2,m=0,w=1,D.sd=1,beta=c(1,3),
                  mu=5,att=c(1:10),fsize=1,FE=1,fixF=TRUE, fixL=TRUE)
head(simdata)
```

```
##    id time        Y       Y0       Y1 D eff       error     Ybar mu X0 treat
## 1 101    1 6.409181 6.409181 6.409181 0   0 -0.56047565 6.969657  5  1     1
## 2 101    2 5.973575 5.973575 5.973575 0   0 -0.23017749 6.203753  5  1     1
## 3 101    3 4.877956 4.877956 4.877956 0   0  1.55870831 3.319248  5  1     1
## 4 101    4 4.284091 4.284091 4.284091 0   0  0.07050839 4.213582  5  1     1
## 5 101    5 9.472339 9.472339 9.472339 0   0  0.12928774 9.343051  5  1     1
## 6 101    6 8.950004 8.950004 8.950004 0   0  1.71506499 7.234939  5  1     1
##            X1         X2         F1         F2         L1         L2      alpha
## 1  0.05620745  0.9226368  0.9021509 -1.0916904 -0.1359097 -0.1737427 -0.8535746
## 2  2.57565884  0.1057750 -0.3494070 -1.4527681 -0.1359097 -0.1737427 -0.8535746
## 3  0.73819369 -0.2880781  0.5899300 -0.7816915 -0.1359097 -0.1737427 -0.8535746
## 4 -1.06129387  0.3505345 -0.1578084 -0.7984798 -0.1359097 -0.1737427 -0.8535746
## 5  0.74677802  1.3226294  0.7978737 -0.7184602 -0.1359097 -0.1737427 -0.8535746
## 6 -0.85501609  1.1717709  1.0865618 -0.2511037 -0.1359097 -0.1737427 -0.8535746
##            xi
## 1 -0.06794850
## 2 -1.13555204
## 3 -0.75677320
## 4 -0.08333068
## 5  0.46557078
## 6  0.53226379
```

```r
true.effect<-matrix(simdata$eff,30,41)[,1]
```

Standard Synthetic Control Method

```r
dataprep.out <-
  dataprep(simdata,
           predictors = c("X1", "X2"),
           dependent     = "Y",
           unit.variable = "id",
           time.variable = "time",
           treatment.identifier  = 101,
           controls.identifier   = c(102:141),
           time.predictors.prior = c(1:20),
           time.optimize.ssr     = c(1:20),
           time.plot             = c(1:30)
  )

# Run synth
synth.out <- synth(dataprep.out)
```

```
##
## X1, X0, Z1, Z0 all come directly from dataprep object.
##
##
## ****************
##  searching for synthetic control unit
##
##
## ****************
## ****************
## ****************
##
## MSPE (LOSS V): 8.756496
##
## solution.v:
##  0.3880988 0.6119012
##
## solution.w:
##  0.02504122 0.02579798 0.01996188 0.02513658 0.02280001 0.02384903 0.1024963 0.0252746 0.0228754 0.02
```

```r
print(synth.tables   <- synth.tab(
  dataprep.res = dataprep.out,
  synth.res    = synth.out)
)
```

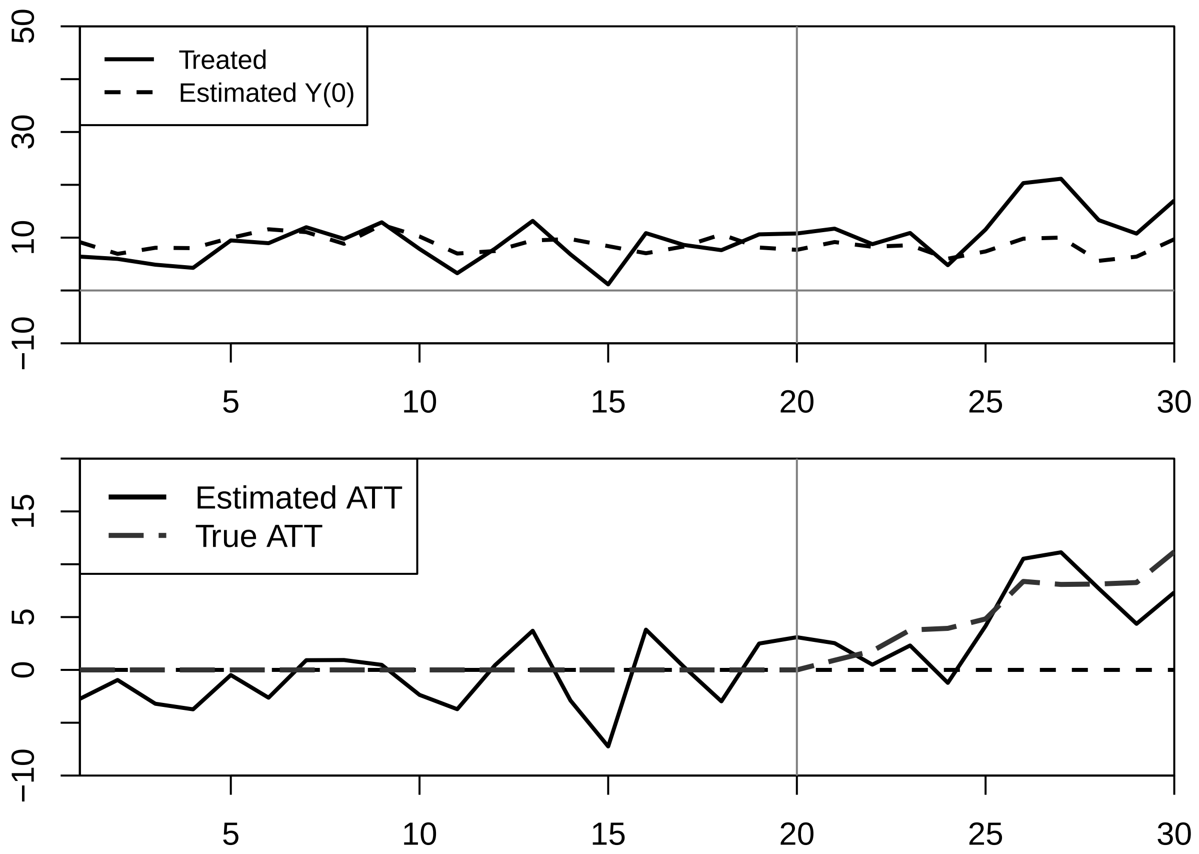
```
## $tab.pred
##    Treated Synthetic Sample Mean
## X1   0.986     0.986       0.966
## X2   0.938     0.938       1.012
##
## $tab.v
##    v.weights
## X1 0.388
## X2 0.612
```

```
##
## $tab.w
##     w.weights unit.names unit.numbers
## 102     0.025        102          102
## 103     0.026        103          103
## 104     0.020        104          104
## 105     0.025        105          105
## 106     0.023        106          106
## 107     0.024        107          107
## 108     0.102        108          108
## 109     0.025        109          109
## 110     0.023        110          110
## 111     0.023        111          111
## 112     0.024        112          112
## 113     0.022        113          113
## 114     0.023        114          114
## 115     0.023        115          115
## 116     0.023        116          116
## 117     0.020        117          117
## 118     0.019        118          118
## 119     0.022        119          119
## 120     0.027        120          120
## 121     0.022        121          121
## 122     0.026        122          122
## 123     0.027        123          123
## 124     0.025        124          124
## 125     0.023        125          125
## 126     0.020        126          126
## 127     0.025        127          127
## 128     0.024        128          128
## 129     0.019        129          129
## 130     0.021        130          130
## 131     0.021        131          131
## 132     0.024        132          132
## 133     0.026        133          133
## 134     0.019        134          134
## 135     0.023        135          135
## 136     0.025        136          136
## 137     0.023        137          137
## 138     0.019        138          138
## 139     0.024        139          139
## 140     0.021        140          140
## 141     0.023        141          141
##
## $tab.loss
##           Loss W   Loss V
## [1,] 2.43636e-14 8.756496
```

```
par(mfcol=c(2,1),mar=c(2,3,1,1),lend=1)
path.plot(synth.res    = synth.out,
          dataprep.res = dataprep.out,
          Ylab         = c("Y"),
          Xlab         = c("Year"),
          Ylim         = c(-10,50),
```

```
          Legend        = c("Treated","Estimated Y(0)"),
          Legend.position = c("topleft")
)
abline(h=0,col="gray50",lty=1)
abline(v=20,col="gray50",lty=1,lwd=1)

par(lend=1)
gaps.plot(synth.res     = synth.out,
          dataprep.res = dataprep.out,
          Ylab         = c("Estimated ATT"),
          Xlab         = c("Year"),
          Ylim         = c(-10, 20),
          Main         = ""
)
lines(1:30,true.effect,col="gray20",lty=5,lwd=2.5)
abline(v=20,col="gray50",lty=1,lwd=1)
legend("topleft",
       legend=c("Estimated ATT","True ATT"),
       col=c(1,"gray20"),
       lty=c(1,5),lwd=c(2.5,2.5))
```



Generalized Synthetic Control Method

```
system.time(
  out <- gsynth(Y ~ D + X1 + X2, data = simdata,
                index=c("id","time"), inference="parametric",
```

```r
                se = TRUE, nboots = 1000, r = c(0, 5), CV = TRUE,
                force = "two-way", parallel = TRUE, cores = 4)
)
```

```
## Parallel computing ...
## Cross-validating ...
##  r = 0; sigma2 = 1.77693; IC = 0.99439; PC = 1.67180; MSPE = 1.50434
##  r = 1; sigma2 = 1.45788; IC = 1.19234; PC = 2.01519; MSPE = 1.19745
##  r = 2; sigma2 = 0.99973; IC = 1.19914; PC = 1.82490; MSPE = 1.03294
##  r = 3; sigma2 = 0.94775; IC = 1.51797; PC = 2.15156; MSPE = 0.85662
##  r = 4; sigma2 = 0.89882; IC = 1.82537; PC = 2.44173; MSPE = 0.80854*
##  r = 5; sigma2 = 0.84869; IC = 2.11658; PC = 2.68588; MSPE = 0.89193
##
##  r* = 4
##
## Simulating errors ...Bootstrapping ...
##

##    user  system elapsed
##   1.207   0.602  18.937
```

```r
## save the results
Y<-out$Y.dat
tb<-out$est.att
Yb<-out$Y.bar[,1:2] ## treated average and counterfactual average
tr<-out$tr
pre<-out$pre
T<-out$T
T0<-out$T0
p<-out$p
m<-out$m
Ntr<-out$Ntr
F.hat<-out$factor
L.tr<-out$lambda.tr
L.co<-out$lambda.co
time<-out$time
time.bf<-time[unique(T0)]
show <- 1:30
```

```r
par(mfcol=c(2,1),mar=c(2,3,1,1),lend=1)
# raw plot
plot(time[show],Yb[show,1],type="n",ylim=c(-10, 50),main="",ylab="",xlab="")
lines(time[show],Yb[show,2],col="gray20",lty=5,lwd=2.5)
lines(time[show],Yb[show,1],col=1,lty=1,lwd=2.5)
abline(v=time.bf,col="gray50",lty=1,lwd=1)
legend("topleft",legend=c("Treated",
                          "Estimated Y(0) Average for the Treated"),
       cex=1.3, seg.len=2, col=c("1","gray20"),
       fill=NA,border=NA, lty=c(1,5), lwd=c(2.5,2.5), merge=T,bty="n")
## gap plot
par(lend=1)
plot(time[show],tb[show,1],type="n",ylim=c(-10,20),main="",ylab="",xlab="")
polygon(c(rev(time[show]),
          time[show]),
        c(rev(tb[show,4]), tb[show,3]), col = '#80808050', border = NA)
```

```
abline(h=0,col="gray50",lty=1)
abline(v=time.bf,col="gray50",lty=1,lwd=1)
lines(time[show],tb[show,1],lwd=2)
#lines(1:T,rowMeans(true.effect),col="gray20",lty=5,lwd=2.5)
lines(1:T,true.effect,col="gray20",lty=5,lwd=2.5)
legend("topleft",
       legend=c("Estimated ATT","True ATT","95% Confidence Intervals"),
       seg.len=2, cex=1.3, col=c(1,"gray20","#80808050"),
       lty=c(1,5, 1),lwd=c(2.5,2.5,15), bty="n",border=NA)
```