Lasso Implementation in Instrumental Regression

Gewei Cao 3461232

Contents

1 Introduction

In causal interpretation regression, researchers usually face the problem of endogeneity, which means interested independent variable **X** is correlated with the error term **ε**, this may be caused by omitted variable, reverse causality and so on. Such endogeneity will cause the estimated causal effect of **X** on dependent variable **y** to be biased. Therefore, instrument variable (IV) regression is used to solve such problem.

IV regression has 2 stages, in the first stage regression people use some exogenous instrument variable **z** to predict suspected endogenous independent variable **X**, then in the second stage regression, regress **y** on the predicted $\hat{\boldsymbol{X}}$ and find the causal effect **β**. Thus, in the first stage, people mainly focus on prediction, rather than interpretation, thus here Machine Learning (ML) methods could be valid. For example, *Lennon, Rubin & Waddell(2021)* tested the performance of different ML-based algorithms in IV regression.

In this paper, I will implement Lasso-based methods in the first stage of IV regression, in order to compare them with the standard 2SLS method. This paper will only focus on the estimation of causal effect, and will not pay attention to coefficient inference, such as significance test and confidence interval. My data generating process (DGP) is based on the DGP proposed in *Lennon, Rubin & Waddell(2021)*, which is designed for IV regression simulation. After setting the basic DGP and regression model, I will manipulate my

DGP to show the properties of these methods. Finally, I will conclude and give some advice about the Lasso and ML implementation in IV regression. In addition, this paper dose not contain any empirical data and empirical practice, because it already has so many pages, but I will try to cover many situations that we may encounter in empirical study in the simulation section.

2 Methods

2.1 Lasso regression and Lasso in IV regression

Lasso regression is a regularization method, which minimize:

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|$$

where *n* is the number of observations, and *p* is the number of independent variables, *y* is dependent variable, and *x* are independent variables. In addition, in Lasso, *p>>n* is allowed.

Such shrinkage method could make some independent variables' effect to be zero, which only have weak or moderate effect on *y*. By doing this, our model could reduce variance at the cost of some increased bias in bias-variance trade-off. Meanwhile, Lasso could be used in variable selection, because it makes some variables have zero coefficients. Lasso works well when some β_j have some value close to zero, but it could give heavily biased estimates if all β_j have values far from zero, this helps us understand the importance of sparsity assumption in *Belloni et al.(2012)*, the whole statement is in the next paragraph.

In the IV regression, suggested by *Belloni et al.(2012)*, Lasso could be used in the first stage. That is, regress **X** on all instrumental variables **z** and all control variables **C** by Lasso, and then use the predicted $\widehat{\boldsymbol{X}}$ and all control variables **C** to regress **y**. *Belloni et al.(2012)* showed that the IV estimator obtained by Lasso in the first stage is root-n consistent and asymptotically normal when the first stage is approximately sparse[1]. In addition, the tuning parameter λ is chosen by 10-fold cross-validation.

2.2 Post-Lasso in the IV regression

As mentioned above, Lasso has the ability for variable selection. We can use Lasso before the first stage, regress **X** on all instrumental variables **z** and all control variables **C** by Lasso regression, and use those **z** whose coefficients are non-zero as selected instrumental variables. Then use these selected instruments in the ordinary 2SLS regression. Such process is called post-Lasso. *Belloni et al.(2012)* also showed that the IV estimator obtained by post-Lasso in the first stage is root-n consistent and asymptotically normal when the first stage is approximately sparse[2]. In addition, the tuning parameter λ is chosen by 10-fold cross-validation. There is another algorithm proposed by *Chernozhukov et al.(2015)*, and we will discuss it in section 4.6.

[1] this means "when the conditional expectation of the endogenous variables given the instruments can be well-approximated by a relatively small set of variables whose identities may be unknown" said by *Belloni et al (2012)*.

[2] The same as footnote 1 (above)

2.3 PCA in IV regression

Lennon, Rubin & Waddell(2021) also mentioned that PCA could be preferred in the first stage, because PCA in IV regression "preserve linearity in both stages", and this enables the exclusion restriction of **z** holds. Thus, I add PCA to our comparison.

The principal component analysis (PCA) works as follows:

Let $Z_1, Z_2...Z_M$ denote the linear combinations of $X_1, X_2...X_P$, $M<P$, here X are independent variables,

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$$

then we fit linear regression model

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} \quad for\ i = 1,2, \dots ., n$$

Each Z_m is called a principal component, the first principal component Z_1 captures the direction where data varies the most, and the second Z_2 captures the direction where data varies the most after subtracting the influence of the first principal component. In other words, principal components are orthogonal to each other. After independent variables being standardized, the eigenvectors of variance-covariance matrix of independent variable matrix give us the directions of principal components, these directions will be the same as directions of PC loadings, we can find that by the meaning of dot product: $Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$, each element in vector Z_m is a dot product of loading vector $\mathbf{\Phi}$ and vector X_i, and this means the length (could be negative only to denote direction) of projection of X_i on the direction of PC_m. In addition, $\mathbf{\Phi}$ for each Z satisfies $\sum_{j=1}^{p} \phi_{jm}^2 = 1 \quad for\ \forall m$, which ensures that we could find the maximum variance of Z_m.

In IV regression, we at first implement PCA on our **z**, and then use the first k principal components as the instruments in the first stage, and then calculate the second stage coefficients. The number of PCs, k, is chosen by 10-fold cross-validation of PCR, the regression of interested variable on **z**.

3 Baseline Data Generating Process (DGP)

My DGP is based on the DGP used in *Lennon, Rubin & Waddell(2021)*, it contains two settings, the first is called the low-complexity case, the other one is called the high-complexity case. The high-complexity case is proposed by *Belloni et al.(2012)*. Both settings are designed to have the problem of endogeneity, and do not have any control variables for the sake of simplicity, therefore, they could be used to compare different methods in IV regression. I take low-complexity DGP in this section as baseline, and in the next section I will adjust it in order to better compare 2SLS and ML-based IV regressions.

3.1 Low-complexity case

We have n = 100 observations, and $|\mathbf{z}|$ = 7 instruments, **what we are interested in is the effect of x_1 on y,** and there is one endogenous omitted variable x_2 that we cannot observe. Such setting is over-identified, in reality researcher could have multiple instruments and one instrumented variable, here traditional 2SLS might works well, and we could compare ML based 2SLS performance with traditional 2SLS.

$$True\ function\!:\ \ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon_y$$

$$x_1 = g(\mathbf{z}) + \varepsilon_c \qquad and\ g(\mathbf{z}) = \sum_{i=1}^{7} z_i$$

$$x_2 = 1 + \varepsilon_c$$

$$\varepsilon_y = \beta_2 x_2 + \delta$$

Here $\beta_0 = \beta_1 = \beta_2 = 1$, common error $\varepsilon_c \sim N(0, 1)$, and δ is distributed uniformly from -1 to 1.

For instrument variables **z**, they follow multivariate normal distribution centered at 0, and for variance-covariance matrix $\Sigma_z\ where\ Cov(z_i, z_j) = 0.6^{|i-j|}$.

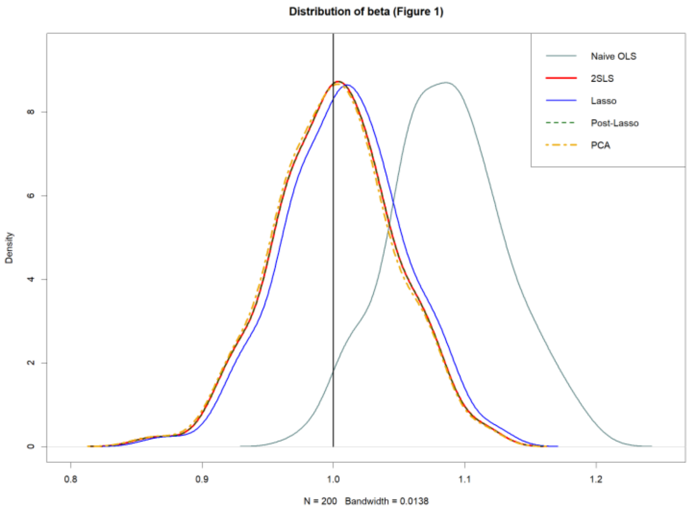
```
library(MASS)
n <- 100
beta <- c(1,1,1)
sigmaz <- matrix(NA,7,7)
mu <- rep(0,7)
for (i in 1:7) { for (j in 1:7) {  sigmaz[i,j] <- 0.6^abs(i-j) }}
dgp_low <- function(n,mu,sigmaz){
  error_c <- rnorm(n,0,1)
  delta <- runif(n,-1,1)
  z <- mvrnorm(n,mu,sigmaz)
  x1 <- z[,1]+z[,2]+z[,3]+z[,4]+z[,5]+z[,6]+z[,7]+error_c
  x2 <- 1+error_c
  error_y <- x2+delta
  y <- 1+x1+x2+error_y
  data <- data.frame(y,x1,z)
  return(data) }
```

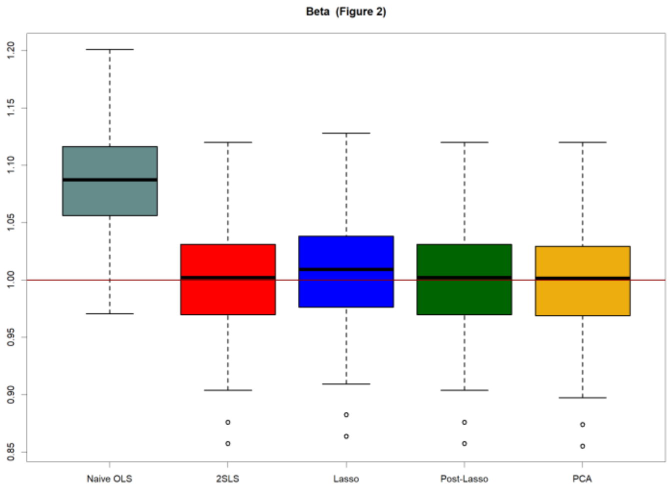
4 Simulation and Manipulation

4.1 Baseline Simulation

I simulate T = 200 times, and because we only compare the estimated coefficients, we only need a training data to estimate.

```
library(glmnet)
library(pls)
library(AER)
T <- 200
Coef <- matrix(NA,T,5)
colnames(Coef) <- c("Naive OLS","2SLS","Lasso","Post-Lasso","PCA")
set.seed(527)
for (i in 1:T) {
  training <- dgp_low(n,mu,sigmaz)
  colnames(training) <- c("y","x","z1","z2","z3","z4","z5","z6","z7")
  Instruments_tr <- as.matrix(training[,3:9])
  ## benchmark OLS and IV
  NaiveOLS <- lm(y~x,data=training)
  Coef[i,1] <- coef(NaiveOLS)[2]
  IVbenchmark <- ivreg(training$y~training$x|Instruments_tr)
  Coef[i,2] <- coef(IVbenchmark)[2]
  ## lasso
  grid <- 10^seq(-3,3,length=200)
  Lasso <- glmnet(Instruments_tr,training$x, lambda = grid,alpha=1)
  cv.out<-cv.glmnet(Instruments_tr, training$x,alpha=1)
  best_lambda<-cv.out$lambda.min
  lasso.xhat<-predict(Lasso, s=best_lambda, newx = Instruments_tr)
  Lasso.2sls <- lm(training$y~lasso.xhat)
  Coef[i,3] <- coef(Lasso.2sls)[2]
  ## post-lasso
  cv.out.post<-cv.glmnet(Instruments_tr, training$x, alpha=1)
  best_lambda<-cv.out.post$lambda.min
  Lasso.post <- glmnet(Instruments_tr,training$x, lambda = best_lambda,alpha=1)
  coelasso <- coef(Lasso.post)
  select <- ifelse(coelasso==0,FALSE,TRUE)[-1]
  Instruments_tr2 <- as.matrix(training[,3:9][,select])
  IVpostLasso <- ivreg(training$y~training$x|Instruments_tr2)
  Coef[i,4] <- coef(IVpostLasso)[2]
  # PCA
  PCR<-pcr(training$x~Instruments_tr, scale=TRUE,validation="CV")
  cverr <- RMSEP(PCR)$val[1,,-1]
  imin<-which.min(cverr)
  PCA_IV <- PCR$scores[,1:imin]
  IVPCR <- ivreg(training$y~training$x|PCA_IV)
  Coef[i,5] <- coef(IVPCR)[2]
}
```





	Naive OLS	2SLS	Lasso	Post-Lasso	PCA
Mean	1.086	1.001	1.008	1.001	1.000
Variance	0.002	0.002	0.002	0.002	0.002

Table 1: Summary of Coefficients in Baseline Simulation

As we can see above, figure 1 and figure 2 show that the naïve OLS coefficient is strongly biased, and the performance of Lasso is worse than the performance of 2SLS, Post-lasso and PCA. From the summary table 1 we also find that the mean coefficient obtained by PCA is smaller than all others, this could be found in figure 1 as well, although such advantage is not significant. In addition, from figure 1 we can find that the coefficients' distribution of Post-Lasso is the same as 2SLS, this shows that in most simulated cases, Lasso before the first stage chooses all 7 instruments. What makes Lasso performs worse in the first stage than 2SLS and other ML methods? Perhaps because our function $g(\boldsymbol{z}) = \sum_{i=1}^{7} z_i$ whose all coefficients of **z** are non-zero value let regularization by Lasso becomes a disadvantage, in other words, the bias caused by Lasso is not mitigated by lower model variance, we will find if this suspect is correct in the next simulation.

4.2 Weak Instruments case

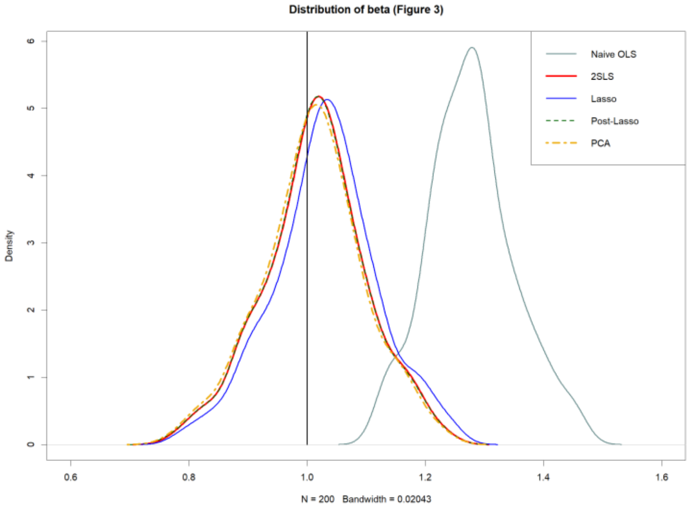
In this case, we change $x_1 = g(\boldsymbol{z}) + \varepsilon_c \quad and\ g(\boldsymbol{z}) = \sum_{i=1}^{7} z_i$ to

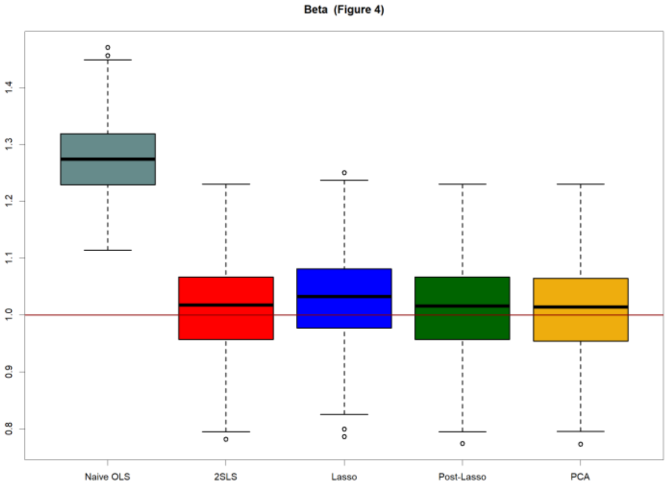
$$x_1 = g(\boldsymbol{z}) + \varepsilon_c \qquad and\ g(\boldsymbol{z}) = \boldsymbol{z}\tilde{\beta}\ \ where\ \tilde{\beta} = [1, 0.8, 0.6, 0.5, 0.3, 0.2, 0.02]^T$$

Such manipulation allows Lasso performs better in case of prediction, and this setting implies that z_5 z_6 z_7 are weak or moderate instruments, and other instruments are relatively strong, such situation usually happens in empirical research.

Here I only show the changed DGP function, the simulation code could be found in Appendix.

```
beta_tild <- c(1,0.8,0.6,0.5,0.3,0.2,0.02)
dgp_low2 <- function(n,mu,sigmaz){
  error_c <- rnorm(n,0,1)
  delta <- runif(n,-1,1)
  z <- mvrnorm(n,mu,sigmaz)
  x1 <- z%*%beta_tild+error_c
  x2 <- 1+error_c
  error_y <- x2+delta
  y <- 1+x1+x2+error_y
  data <- data.frame(y,x1,z)
  return(data)
}
```





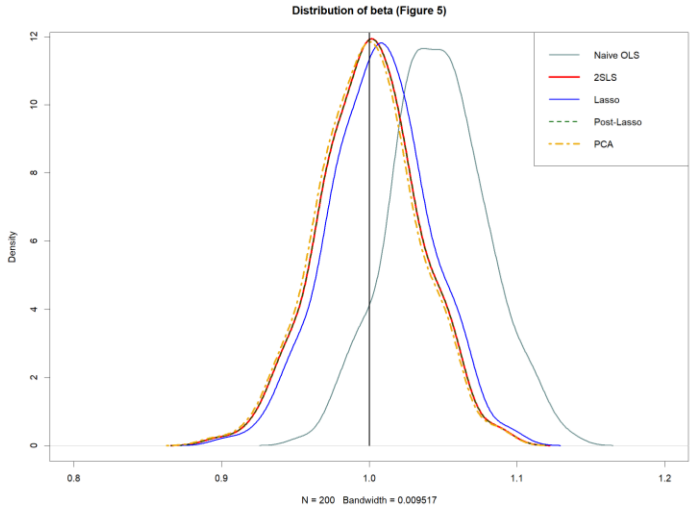
	Naive OLS	2SLS	Lasso	Post-Lasso	PCA
Mean	1.277	1.014	1.030	1.014	1.010
Variance	0.005	0.007	0.008	0.007	0.008

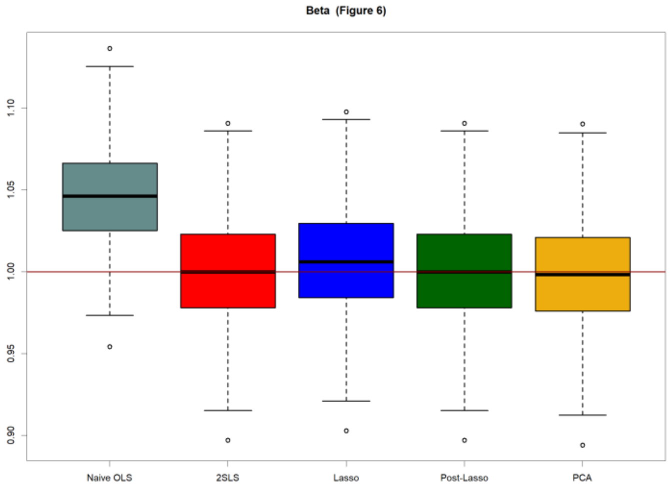
Table 2: Summary of Coefficients in Weak Instruments Simulation

Here we can find in figure 3 & 4 that in the case of weak instruments, naïve OLS is more biased, and all estimated coefficients have larger variance than baseline case. From table 2 and figure 3 we can say that PCA still has a insignificant advantage over others. Meanwhile, in such weak instrument setting, Post-Lasso performs very slightly better than 2SLS, this might be caused by variable selection function of Lasso. Nevertheless, Lasso still performs worse than 2SLS and the other ML approaches. Thus, here we find that the worse performance of Lasso is not caused by the form of *g(z)*, in other words, the absence of weak instruments.

4.3 Highly Correlated Instruments case

In empirical research, sometimes there will be a set of highly correlated instruments, for such case, we change the var-covariance matrix of instruments to be $\Sigma_z \; where \; Cov(z_i, z_j) = 0.9^{|i-j|}$, and all other settings are the same as the baseline case. The simulation code and DGP code are similar with before, you can find it in R file.





	Naive OLS	2SLS	Lasso	Post-Lasso	PCA
Mean	1.046	1.000	1.007	1.000	0.998
Variance	0.001	0.001	0.001	0.001	0.001

Table 3: Summary of Coefficients in Highly Correlated Instruments Simulation

Here we obtain nearly the same results as baseline case, and the difference is all estimates have smaller variances, this gives us confidence to use Post-Lasso, PCA or traditional 2SLS in the case of highly correlated coefficients.

4.4 High Dimensional Instrument Set case

Here we assume researcher could find a lot of instruments, although in economic researches this might be hard. The motivation to make such assumption is that in high dimensional case, people may attempt to use ML-based methods. The DGP is as following:

We expand **z** to 60 dimensions, and **z** follows the same distribution as baseline case, meanwhile here

$$x_1 = g(\mathbf{z}) + \varepsilon_c \quad and \; g(\mathbf{z}) = \sum_{i=1}^{60} \alpha_i z_i \; where \; each \; \alpha_i \; is \; randomly \; sampled \; from \; interval \; [0, 1]$$

and all other settings keep the same as baseline. In addition, among all 200 simulations, we use the same vector $\boldsymbol{\alpha}$. The simulation code is similar with before, you can find it in R file.

```r
set.seed(527)
beta <- c(1,1,1)
sigmaz3 <- matrix(NA,60,60)
mu <- rep(0,60)
for (i in 1:60) {
  for (j in 1:60) {
    sigmaz3[i,j] <- 0.6^abs(i-j)
  }  }
alpha <- sample(seq(0,1,by=0.001),60)
dgp_low4 <- function(n,mu,sigmaz){
  error_c <- rnorm(n,0,1)
  delta <- runif(n,-1,1)
  x1 <- c()
  z <- mvrnorm(n,mu,sigmaz)
  for (i in 1:n) {
      x1[i] <- z[i,1:60]%*%alpha+error_c[i]
  }
  x2 <- 1+error_c
  error_y <- x2+delta
  y <- 1+x1+x2+error_y
  data <- data.frame(y,x1,z)
  return(data)
}
```

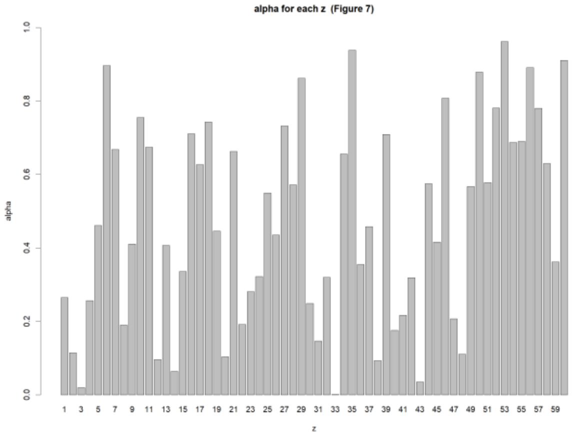
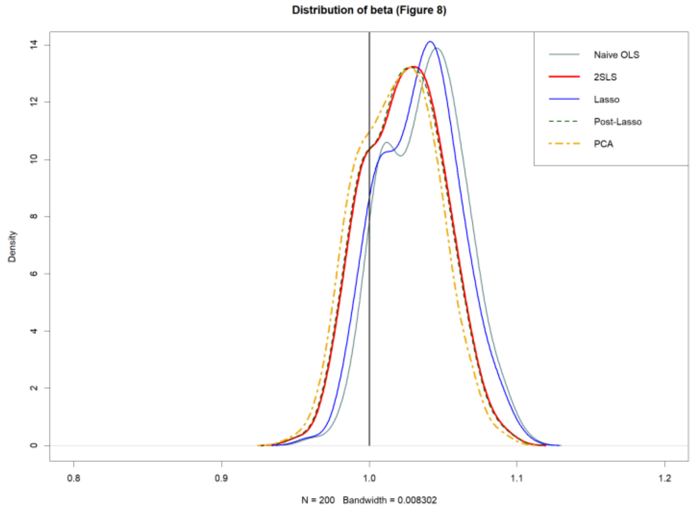
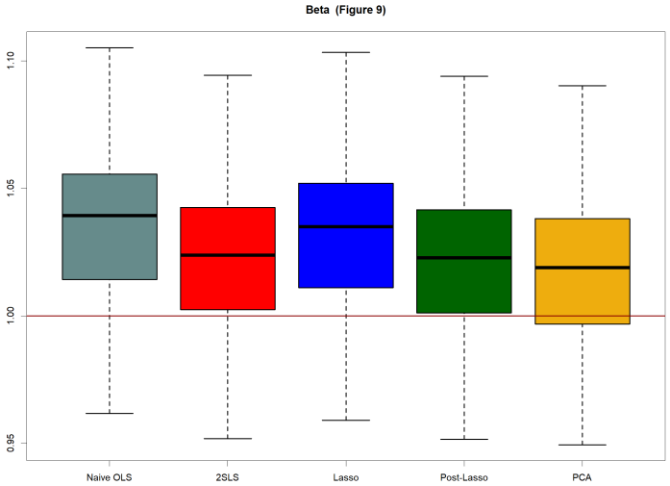


Figure 7 shows the α value of z_i from i belongs 1 to 60, in such high-dimensional setting, ML supposed to work well. In addition, do not forget that for each instrument variable, only the "neighbor" instruments relatively stronger correlated with each other, this also gives ML based IV regression a chance to outperform.





	Naive OLS	2SLS	Lasso	Post-Lasso	PCA
Mean	1.037	1.024	1.034	1.023	1.019
Variance	0.00071	0.00070	0.00072	0.00070	0.00070

Table 4: Summary of Coefficients in High Dimensional Simulation

Figure 8 & 9 show that in such high dimensional case, Post-Lasso and PCA have a little outperformed to 2SLS. Meanwhile, PCA performs slightly better than all others. However, Lasso is biased, and its coefficients' distribution is nearly similar with the one of naïve OLS, this tells us that not all ML based models performs well in high dimensional IV regression, and we have to be cautious when using ML related to IV interpretation. But dose this violate the statement in *Belloni et al.(2012)*? Probably not, look figure 7 again, actually there are not so much weak instruments, which makes the instrument space not "sparse" enough, and this might make the estimated coefficient by Lasso to be inconsistent.

According to *Lennon, Rubin & Waddell(2021)*, they decomposed the bias of estimated coefficient as:

$$\hat{\beta}^{2SLS} - \beta_1 = \frac{\beta_1 Cov(\hat{x}\,, e) + Cov(\hat{x}\,, u)}{Var(\hat{x})}$$

where e is the residual of the first stage regression, and $\hat{x}$ is the estimated independent variable by the first stage, u is the residual of naïve OLS method. The OLS based methods, for example, Post-Lasso and 2SLS, ensures that $Cov(\hat{x}\,, e) = 0$, because of orthogonality, but for Lasso, it cannot ensure $Cov(\hat{x}\,, e)$ to be 0. Meanwhile the term $Cov(\hat{x}\,, u)$ is the exclusion restriction, and they pointed out that for sufficiently flexible ML methods, by only using valid instruments, it can recover endogeneity in x. Finally, most of ML algorithms tend to reduce the variance of predictions, here is $Var(\hat{x})$, and this leveraged bias in our setting. All these three components make coefficients estimated by Lasso to be biased.

4.5 Control Variable case

In this setting I add some control variables, some of them are endogenous, some of them are exogenous. Because we need to show what will happen when extend ML-based IV to multiple control variables case, which is the circumstance most often occurs. Here are the settings:

$$True\ function\!:\ \ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8 + \varepsilon_y$$

$$\boldsymbol{\beta} = [1, 1, 1, 0.7, 0.5, 0.35, 0.4, 0.2, 0.1]^T, and\ \ \varepsilon_y = \beta_2 x_2 + \delta$$

$$x_1 = g(\boldsymbol{z}) + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8 + \varepsilon_c$$

$$g(\boldsymbol{z}) = \sum\nolimits_{i=1}^{7} \alpha_i z_i\ \ where\ each\ \alpha_i\ is\ randomly\ sampled\ from\ interval\ [0, 1]$$

$$x_2 = 1 + \varepsilon_c\ \ \ this\ one\ is\ unobservable$$

$$x_3 = N(0{,}2) + \varepsilon_c;\ \ x_4 = Uniform(-1{,}1) + \varepsilon_c\ \ and\ \ x_5 = N(0{,}1.5) + \varepsilon_c$$

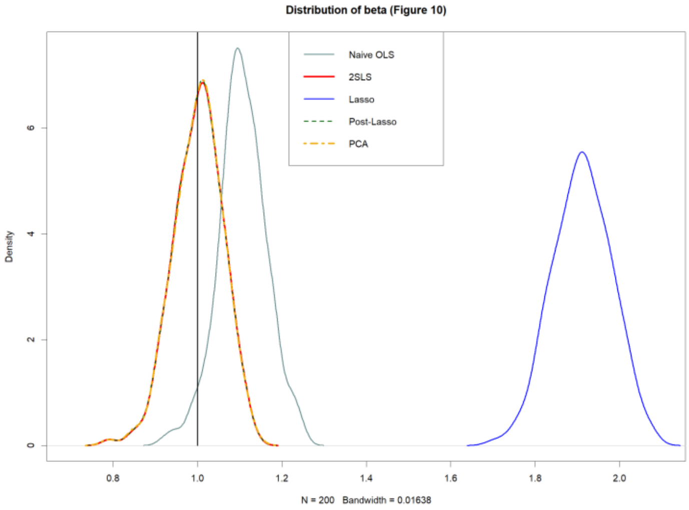
$$x_6\ \&\ x_7\ \&\ x_8 \sim N(\mathbf{0}, \Sigma), and\ \Sigma = \begin{bmatrix} 1 & 0.7 & 0.49 \\ 0.7 & 1 & 0.7 \\ 0.49 & 0.7 & 1 \end{bmatrix}$$

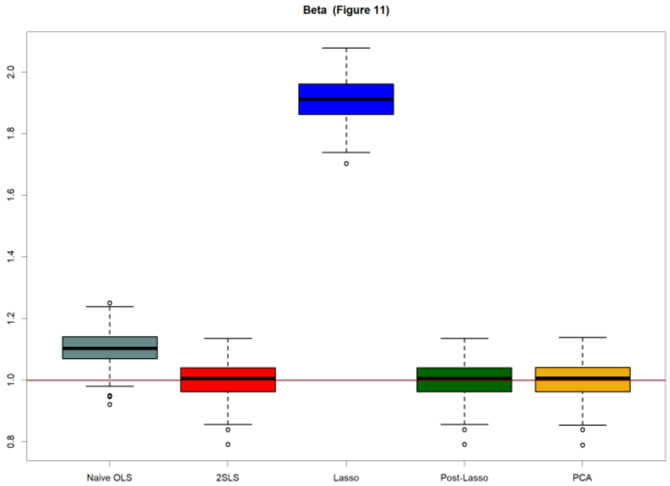
Such manipulation ensures that *x*$_3$*, x*$_4$*, x*$_5$ are endogenous (play the role as proxy controls), and *x*$_6$*, x*$_7$*, x*$_8$ are exogenous, endogenous control variables are allowed for coefficient interpretation. Meanwhile, we need to adjust our ML based IV, to fit control variable setting: *For Lasso*, in the first stage we regress *x*$_1$ on all instruments and controls by Lasso, and use the predicted value $\widehat{x_1}$ in the second stage. *For Post-Lasso*, before the first stage, we use Lasso to regress *x*$_1$ on all instruments and controls and then select instruments whose coefficients are non-zero, then use these instruments and all controls in the first stage. We do not exclude any control variables by Lasso because here we only want Lasso works on instrumental variable selection. A better algorithm is shown in 4.6.

For PCA, because PCA could extract main information of dependent variables, therefore, before the first stage, we use PCA as in section 4.1, but also contain all control variables, and then select the number of principal components that has the smallest MSE in cross validation, use these PC in the first stage.

```
set.seed(527)
beta2 <- c(0.7,0.5,0.35,0.4,0.2,0.1)
mu <- rep(0,7)
alpha <- sample(seq(0,1,by=0.001),7)
muc <- rep(0,3)
sigmac <- matrix(c(1,0.7,0.49,0.7,1,0.7,0.49,0.7,1),3,3,byrow = TRUE)
dgp_low5 <- function(n,mu,sigmaz){
  error_c <- rnorm(n,0,1)
  delta <- runif(n,-1,1)
  x3 <- rnorm(n,0,2)+error_c
  x4 <- runif(n,-1,1)+error_c
  x5 <- rnorm(n,0,1.5)+error_c
  X678 <- mvrnorm(n,muc,sigmac)
  x1 <- c()
  X <- cbind(x3,x4,x5,X678)
  z <- mvrnorm(n,mu,sigmaz)
  for (i in 1:n) { x1[i] <- z[i,1:7]%*%alpha+X[i,]%*%beta2+error_c[i] }
  x2 <- 1+error_c
  error_y <- x2+delta
  y <- 1+x1+x2+X%*%beta2+error_y
  data <- data.frame(y,x1,z,X)
  return(data) }
```

```
# Case 5--Control Variables
set.seed(527)
Coef5 <- matrix(NA,T,5)
colnames(Coef5) <- c("Naive OLS","2SLS","Lasso","Post-Lasso","PCA")
for (i in 1:T) {
  training <- dgp_low5(n,mu,sigmaz)
  Instruments_tr <- as.matrix(training[,3:9])
  Controls <- as.matrix(training[,10:15])
  ## benchmark OLS and IV
  NaiveOLS <- lm(y~x1+x3+x4+x5+V4+V5+V6,data=training)
  Coef5[i,1] <- coef(NaiveOLS)[2]
  IVbenchmark <- ivreg(training$y~training$x1+Controls|Instruments_tr+Controls)
  Coef5[i,2] <- coef(IVbenchmark)[2]
  ## lasso
  grid <- 10^seq(-3,3,length=200)
  Lasso <- glmnet(cbind(Instruments_tr,Controls),training$x1, lambda = grid,alpha=1)
  cv.out<-cv.glmnet(cbind(Instruments_tr,Controls), training$x1,alpha=1)
  best_lambda<-cv.out$lambda.min
  lasso.xhat<-predict(Lasso, s=best_lambda, newx = cbind(Instruments_tr,Controls))
  Lasso.2sls <- lm(training$y~lasso.xhat)
  Coef5[i,3] <- coef(Lasso.2sls)[2]
  ## post-lasso
  cv.out.post<-cv.glmnet(cbind(Instruments_tr,Controls), training$x1, alpha=1)
  best_lambda<-cv.out.post$lambda.min
  Lasso.post<- glmnet(cbind(Instruments_tr,Controls),training$x1,lambda = best_lambda,alpha=1)
  coelasso <- coef(Lasso.post)
  select <- ifelse(coelasso==0,FALSE,TRUE)[2:8]
  Instruments_tr2 <- as.matrix(training[,3:9][,select])
  IVpostLasso <- ivreg(training$y~training$x1+Controls|Instruments_tr2+Controls)
  Coef5[i,4] <- coef(IVpostLasso)[2]
  # PCA
  PCR<-pcr(training$x1~Instruments_tr+Controls, cale=TRUE,validation="CV")
  cverr <- RMSEP(PCR)$val[1,,-1]
  imin<-which.min(cverr)
  PCA_IV <- PCR$scores[,1:imin]
  ttt <- lm(training$x1~PCA_IV)
  xp <- predict(ttt)
  IVPCR <- lm(training$y~xp+Controls)
  Coef5[i,5] <- coef(IVPCR)[2]
}
```

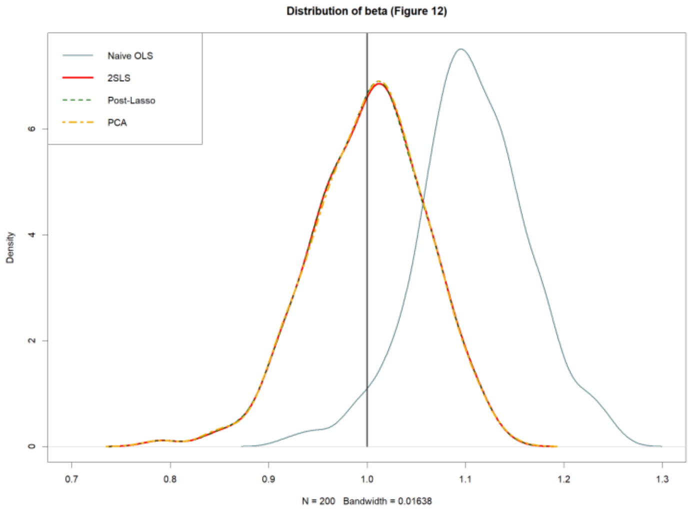




	Naive OLS	2SLS	Lasso	Post-Lasso	PCA
Mean	1.107	1.002	1.911	1.002	1.002
Variance	0.003	0.003	0.005	0.003	0.003

Table 5: Summary of Coefficients in Control Variables Simulation

From figure 10 & 11 we can find that with control variables Lasso is heavily biased, even worse than naïve OLS results, this show that some ML based IV estimate could be biased strongly if there are multiple controls and only using "naïve algorithm", but the IV control selection is out of scope of this paper, *Dutt & Tsetlin(2020)* gave us an example of single instrument and multiple controls, we will discuss a more complex situation in 4.6. Meanwhile, Post-Lasso and PCA performs similar to 2SLS, figure 12 is the zoomed version of figure 11 except Lasso, this also shows that PCA and Post-Lasso do not have significant advantage over 2SLS given multiple controls. In addition, coefficients variance given controls only increased a little than baseline setting, but still much smaller than weak instruments case. Figure 12 also indirectly proves that control variables not only increase prediction accuracy (exogenous controls), but also make interested coefficient more accurate (endogenous controls), although the setting in this case has some differences with baseline case and weak instruments case.



4.6 Control Variable case with Chernozhukov et al. (2015) “Double Selection” algorithm

4.6.1 DGP

Chernozhukov et al.(2015) gave us an algorithm which could deal with DGP setting as in 4.5 better, which there are multiple instruments and multiple controls, this algorithm considering both selecting instrument variables and selecting control variables by Lasso or Post-Lasso. First of all, this algorithm needs all controls to be exogenous, therefore we change DGP in 4.5 by subtracting $\varepsilon_c$from *x₃ x₄ x₅*, which makes them exogenous now, although when I tried including endogenous controls the results are still looks pretty. Then our DGP which follows the assumption in *Chernozhukov et al.(2015)* is shown below, in addition, the sparsity assumption is also satisfied.

$$True\ function\!:\ \ y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8 + \varepsilon_y$$

$$\boldsymbol{\beta} = [1, 1, 1, 0.7, 0.5, 0.35, 0.4, 0.2, 0.1]^T, and\ \ \varepsilon_y = \beta_2 x_2 + \delta$$

$$x_1 = g(\boldsymbol{z}) + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6 + \beta_7 x_7 + \beta_8 x_8 + \varepsilon_c$$

$$g(\boldsymbol{z}) = \sum\nolimits_{i=1}^{7} \alpha_i z_i\ \ where\ each\ \alpha_i\ is\ randomly\ sampled\ from\ interval\ [0, 1]$$

$$x_2 = 1 + \varepsilon_c\ \ \ this\ one\ is\ unobservable$$

$$x_3 {\sim} (0,2),\ \ x_4 {\sim} Uniform(-1,1),\ \ x_5 {\sim} N(0,1.5)$$

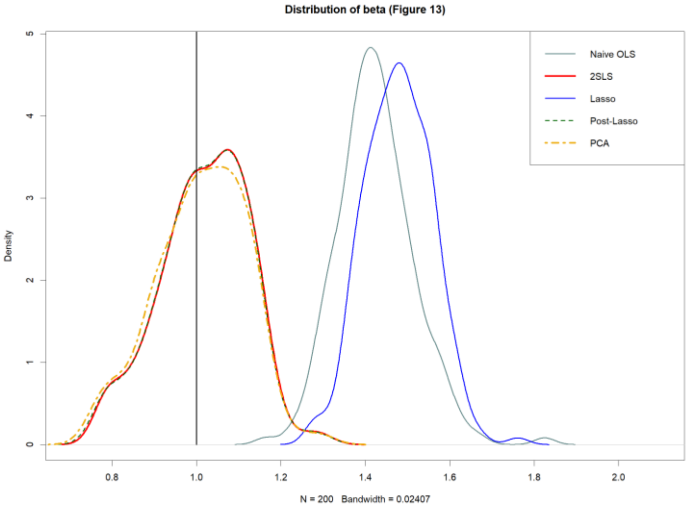
$$x_6\ \&\ x_7\ \&\ x_8 {\sim} N(\mathbf{0}, \Sigma), and\ \Sigma = \begin{bmatrix} 1 & 0.7 & 0.49 \\ 0.7 & 1 & 0.7 \\ 0.49 & 0.7 & 1 \end{bmatrix}$$

In addition, instead of **z** ~ multinormal distribution, we specify:

$$z_{\boldsymbol{i}} = \boldsymbol{\Pi} x_i + \xi_i\ where\ \boldsymbol{\Pi}\ (\text{7 rows, 6 columns}), value\ in\ each\ row\ is\ sampled\ randomly\ from\ interval\ [0, 1],\ \ \ and\ \xi_i {\sim} N(\mathbf{0}, \Sigma_z)$$

Such setting ensures that x_i is orthogonal to ξ_i, and this is what *Chernozhukov et al.(2015)* assumed, this makes z_i and x_i are correlated, and only given x_i, instruments are valid, it is realistic in empirical researching. In addition, we use the same $\boldsymbol{\Pi}$ and $\boldsymbol{\alpha}$ through 200 simulations.

Before we start new simulation, figure 13 shows the result when all controls in 4.5 are exogenous, our conclusion from 4.5 holds and the bias of Lasso is reduced.



4.6.2 "Double Selection" Algorithm in Chernozhukov et al.(2015)

Step 1: use Lasso or Post-Lasso regress x_1 on $X = [x_3, x_4, x_5, x_6, x_7, x_8]$ and $\boldsymbol{z}$ (100×7) to obtain $\hat{\gamma}$ *and* $\hat{\delta}$

Step 2: use Lasso or Post-Lasso regress *y* on X to obtain $\hat{\theta}$

Step 3: use Lasso or Post-Lasso regress $\widehat{x_1} = X\hat{\gamma} + \boldsymbol{z}\hat{\delta}$ on X to get $\hat{v}$

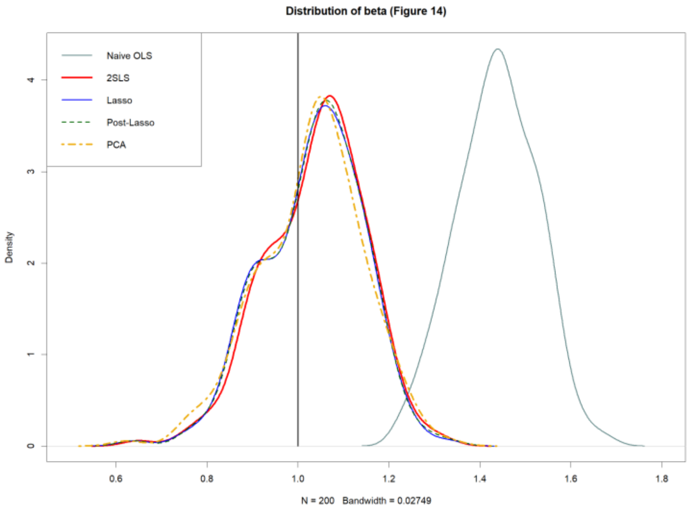
Step 4: let $\hat{\rho}^y = y - X\hat{\theta}$, $\hat{\rho}^d = x_1 - X\hat{v}$, and $\hat{\vartheta} = X\hat{\gamma} + \boldsymbol{z}\hat{\delta} - X\hat{v}$, then we can obtain interested coefficient $\widehat{\beta_1}$ by IV regression, regress $\hat{\rho}^y$ on $\hat{\rho}^d$

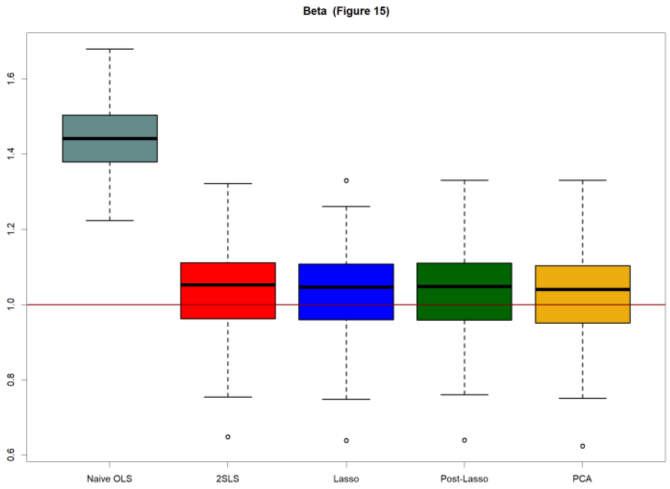
4.6.3 Simulation & Results

```r
set.seed(527)
beta2 <- c(0.7,0.5,0.35,0.4,0.2,0.1)
mu <- rep(0,7)
alpha <- sample(seq(0,1,by=0.001),7)
muc <- rep(0,3)
sigmac <- matrix(c(1,0.7,0.49,0.7,1,0.7,0.49,0.7,1),3,3,byrow = TRUE)
capitalpie <- matrix(NA,6,7)
for (i in 1:6) { capitalpie[i,] <- sample(seq(0,1,by=0.001),7) }
dgp_lowCh <- function(n,mu,sigmaz,capitalpie){
  error_c <- rnorm(n,0,1)
  delta <- runif(n,-1,1)
  x3 <- rnorm(n,0,2)
  x4 <- runif(n,-1,1)
  x5 <- rnorm(n,0,1.5)
  X678 <- mvrnorm(n,muc,sigmac)
  x1 <- c()
  X <- cbind(x3,x4,x5,X678)
  xi <- mvrnorm(n,mu,sigmaz)
  z <- X%*%capitalpie+xi
  for (i in 1:n) { x1[i] <- z[i,1:7]%*%alpha+X[i,]%*%beta2+error_c[i] }
  x2 <- 1+error_c
  error_y <- x2+delta
  y <- 1+x1+x2+X%*%beta2+error_y
  data <- data.frame(y,x1,z,X)
  return(data)
}
# Case 6
set.seed(527)
T <- 200
Coef6 <- matrix(NA,T,5)
colnames(Coef6) <- c("Naive OLS","2SLS","Lasso","Post-Lasso","PCA")
for (i in 1:T) {
  training <- dgp_lowCh(n,mu,sigmaz,capitalpie)
  Instruments_tr <- as.matrix(training[,3:9])
  Controls <- as.matrix(training[,10:15])
  ## benchmark OLS and IV
  NaiveOLS <- lm(y~x1+x3+x4+x5+V4+V5+V6,data=training)
  Coef6[i,1] <- coef(NaiveOLS)[2]
```

```
IVbenchmark <- ivreg(training$y~training$x1+Controls|Instruments_tr+Controls)
Coef6[i,2] <- coef(IVbenchmark)[2]
## lasso
grid <- 10^seq(-3,3,length=200)
cv.out<-cv.glmnet(cbind(Instruments_tr,Controls), training$x1,alpha=1,intercept = F)
best_lambda<-cv.out$lambda.min
Lasso <- glmnet(cbind(Instruments_tr,Controls),training$x1,
                lambda = best_lambda,alpha=1,intercept = F)
gammadelta <- coef(Lasso) #step 1 (above)
cv.out<-cv.glmnet(Controls, training$y,alpha=1,intercept = F)
best_lambda<-cv.out$lambda.min
theta <- coef(glmnet(Controls,training$y,  lambda = best_lambda,alpha=1,intercept = F))
# step 2 (above)
dhat <- cbind(Instruments_tr,Controls)%*%gammadelta[-1]
cv.out<-cv.glmnet( Controls,dhat,alpha=1,intercept = F)
best_lambda<-cv.out$lambda.min
vhat <- coef(glmnet( Controls,dhat, lambda = best_lambda,alpha=1,intercept = F))
# step 3 (above)
rouy <- training$y-Controls%*%theta[-1]
roud <- training$x1-Controls%*%vhat[-1]
vvv <- dhat-Controls%*%vhat[-1]
Lasso.2sls <- ivreg(rouy~roud|vvv)
# step 4 (above)
Coef6[i,3] <- coef(Lasso.2sls)[2]
## post-lasso
cv.out<-cv.glmnet(cbind(Instruments_tr,Controls), training$x1,alpha=1,intercept = F)
best_lambda<-cv.out$lambda.min
Lasso.post <- glmnet(cbind(Instruments_tr,Controls),training$x1,
    lambda = best_lambda,alpha=1,intercept = F)
coelasso <- coef(Lasso.post)
select_1 <- ifelse(coelasso==0,FALSE,TRUE)[-1]
i_t <- as.matrix(training[,3:15][,select_1])
gammadelta <- coef(lm(training$x1~i_t-1))
# step 1 (above)
cv.out<-cv.glmnet(Controls, training$y,alpha=1,intercept = F)
best_lambda<-cv.out$lambda.min
Lasso.post <- glmnet(Controls,training$y, lambda = best_lambda,alpha=1,intercept = F)
coelasso <- coef(Lasso.post)
select_2 <- ifelse(coelasso==0,FALSE,TRUE)[-1]
CC <- as.matrix(training[,10:15][,select_2])
theta <- coef(lm(training$y~CC-1))
#step 2 (above)
dhat <-cbind(Instruments_tr,Controls)[,select_1]%*%gammadelta
cv.out<-cv.glmnet( Controls,dhat,alpha=1,intercept = F)
best_lambda<-cv.out$lambda.min
Lasso.post <- glmnet( Controls,dhat, lambda = best_lambda,alpha=1,intercept = F)
coelasso <- coef(Lasso.post)
select_3 <- ifelse(coelasso==0,FALSE,TRUE)[-1]
```

```
CCd <- as.matrix(training[,10:15][,select_3])
vhat <- coef(lm(dhat~CCd-1)) #step 3 (above)
if(length(theta)==1){
  rouy <- training$y-Controls[,select_2]*theta
}else{ rouy <- training$y-Controls[,select_2]%*%theta }
if(length(vhat)==1){
  roud <- training$x1-Controls[,select_3]*vhat
  vvv <- dhat-Controls[,select_3]*vhat
}else{
  roud <- training$x1-Controls[,select_3]%*%vhat
  vvv <- dhat-Controls[,select_3]%*%vhat
  }
IVpostLasso <- ivreg(rouy~roud|vvv) #step 4 (above)
    Coef6[i,4] <- coef(IVpostLasso)[2]
     # PCA
     PCR<-pcr(training$x1~Instruments_tr+Controls,scale=TRUE,validation="CV")
     cverr <- RMSEP(PCR)$val[1,,-1]
     imin<-which.min(cverr)
     PCA_IV <- PCR$scores[,1:imin]
     ttt <- lm(training$x1~PCA_IV)
     xp <- predict(ttt)
     IVPCR <- lm(training$y~xp+Controls)
     Coef6[i,5] <- coef(IVPCR)[2]
    }
```

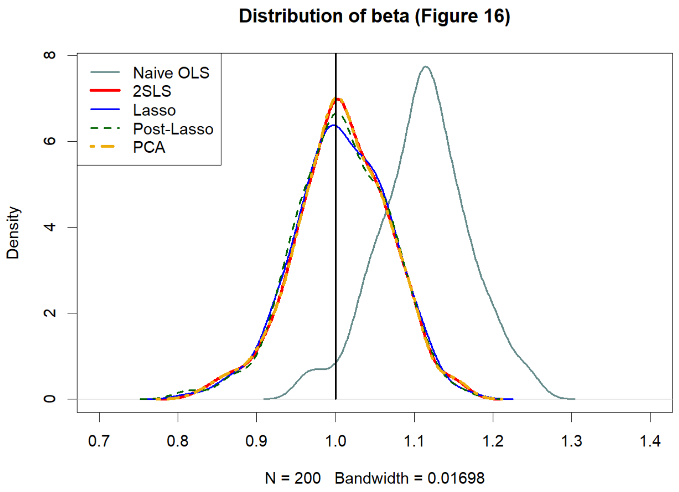




	Naive OLS	2SLS	Lasso	Post-Lasso	PCA
Mean	1.437	1.039	1.033	1.034	1.029
Variance	0.008	0.012	0.012	0.012	0.013

Table 6: Summary of Coefficients in Chernozhukov et al.(2015) Simulation

Here from figure 14 & 15 we can find that, with new algorithm, Lasso performs as good as 2SLS, Post-Lasso and PCA. From the perspective of coefficient’s mean, all ML-based algorithms even outperform to 2SLS, although this is insignificant according to figure 14. *Chernozhukov et al.(2015)* gives us confidence to use Lasso-based methods in IV regression, but this simulation also tells us that when using ML algorithms in IV causal inference, we should have some theoretical support before we try.



In addition, if we use the DGP in 4.5, which the controls **x** are not that highly correlated with **z** as in 4.6, the result shown in figure 16 gives us more confidence of using such "double selection" algorithm. But ML-based IV is designed for high dimensional regression, thus next section will show you its real power.

4.6.4 With High Dimensional Instruments case

PCA, Lasso and Post-Lasso are widely used to deal with high dimensional variables, thus, it makes sense to change our DGP to give us a 60 dimensions instruments **z**.

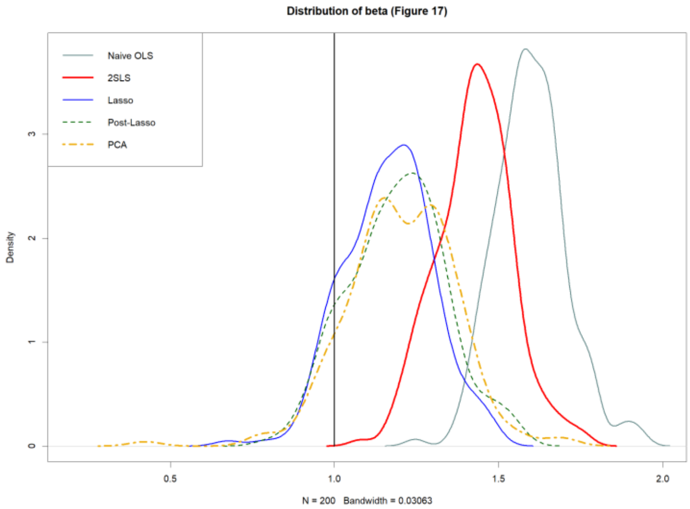
$$g(\mathbf{z}) = \sum_{i=1}^{60} \alpha_i z_i \;\; where \;\; \alpha_i = 0.7^i$$

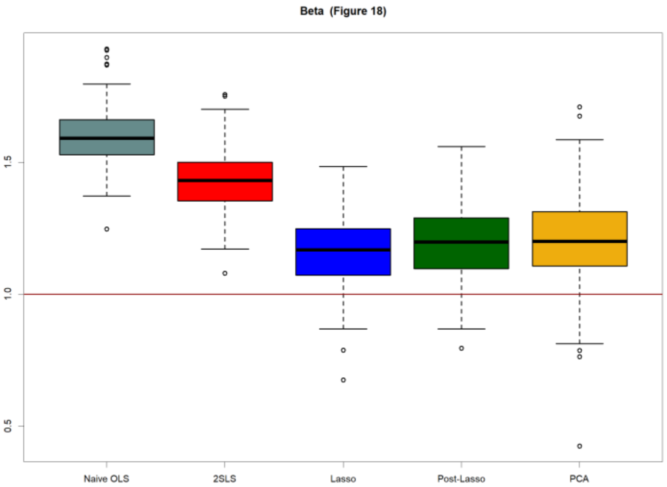
$$z_i = \mathbf{\Pi} x_i + \xi_i \; where \; \mathbf{\Pi} \; (60 \text{ rows}, 6 \text{ columns}), value \; in \; each \; row \; is \; sampled \; randomly \; from \; interval \; [0, 1]$$

$$and \; \xi_i \sim N(\mathbf{0}, \Sigma_\xi), \;\; where \; Cov(\xi_i, \xi_j) = 0.6^{|i-j|}$$

Here the α_i is similar to the high complexity setting in *Lennon, Rubin & Waddell(2021)*, and such setting ensures the space of instruments more "sparse", i.e. here means that there are a lot of weak instruments. I suppose under such DGP, ML-based IV regression should perform at least as good as 2SLS, like that *Belloni et al.(2012)* and *Chernozhukov et al.(2015)* showed. In addition, simulation code is similar with 4.6.3, you can find it in R file, here I only show DGP.

```r
set.seed(527)
beta2 <- c(0.7,0.5,0.35,0.4,0.2,0.1)
mu <- rep(0,60)
alpha <- c()
for (i in 1:60) { alpha[i] <- 0.7^i }
muc <- rep(0,3)
sigmac <- matrix(c(1,0.7,0.49,0.7,1,0.7,0.49,0.7,1),3,3,byrow = TRUE)
capitalpie60 <- matrix(NA,6,60)
for (i in 1:6) {capitalpie60[i,] <- sample(seq(0,1,by=0.001),60)}
sigmaxi <- matrix(NA,60,60)
for (i in 1:60) {
  for (j in 1:60) {
    sigmaxi[i,j] <- 0.6^abs(i-j)}}
dgp_lowCh60 <- function(n,mu,sigmaxi,capitalpie){
  error_c <- rnorm(n,0,1)
  delta <- runif(n,-1,1)
  x3 <- rnorm(n,0,2)
  x4 <- runif(n,-1,1)
  x5 <- rnorm(n,0,1.5)
  X678 <- mvrnorm(n,muc,sigmac)
  x1 <- c()
  X <- cbind(x3,x4,x5,X678)
  xi <- mvrnorm(n,mu,sigmaxi)
  z <- X%*%capitalpie+xi
  for (i in 1:n) {x1[i] <- z[i,]%*%alpha+X[i,]%*%beta2+error_c[i]}
  x2 <- 1+error_c
  error_y <- x2+delta
  y <- 1+x1+x2+X%*%beta2+error_y
  data <- data.frame(y,x1,z,X)
  return(data)
}
```



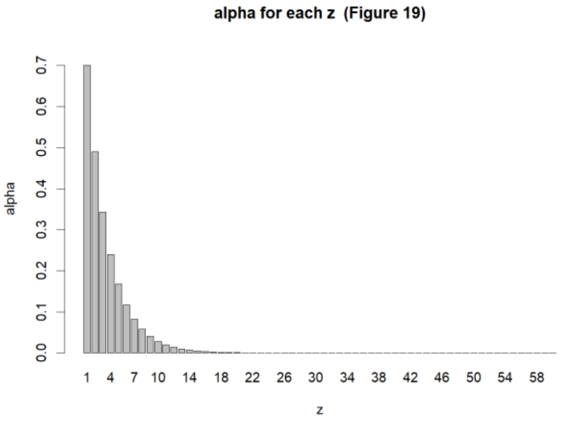


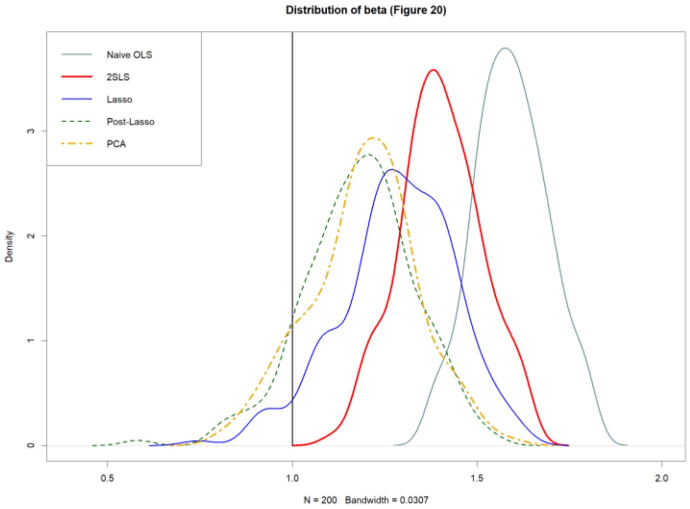
	Naive OLS	2SLS	Lasso	Post-Lasso	PCA
Mean	1.597	1.426	1.165	1.192	1.203
Variance	0.011	0.013	0.018	0.022	0.027

Table 7: Summary of Coefficients in Chernozhukov et al.(2015) Simulation

Figure 17 & 18, also table 7 show us that all ML-based IV regression outperforms 2SLS in such high-dimension setting. Although these three ML algorithms still output biased estimates, they all outperform traditional 2SLS. Thus, we can conclude that with high dimension instruments, Post-Lasso and PCA with algorithms both in 4.4 and 4.6 will performs at least as good as 2SLS, and with *Chernozhukov et al.(2015)*'s algorithm, Lasso with the other two could outperform 2SLS. Nevertheless, we notice that the variance of these three ML IV regressions are bigger than the variance of 2SLS, and PCA has the largest variance.

In addition, I use the $\boldsymbol{\alpha}$ in 4.6.4 to reproduce 4.4 (without controls), both of them are given a high dimensional instruments space. The coefficients of *g(***z***)* are shown in figure 19, the same as in 4.6.4.

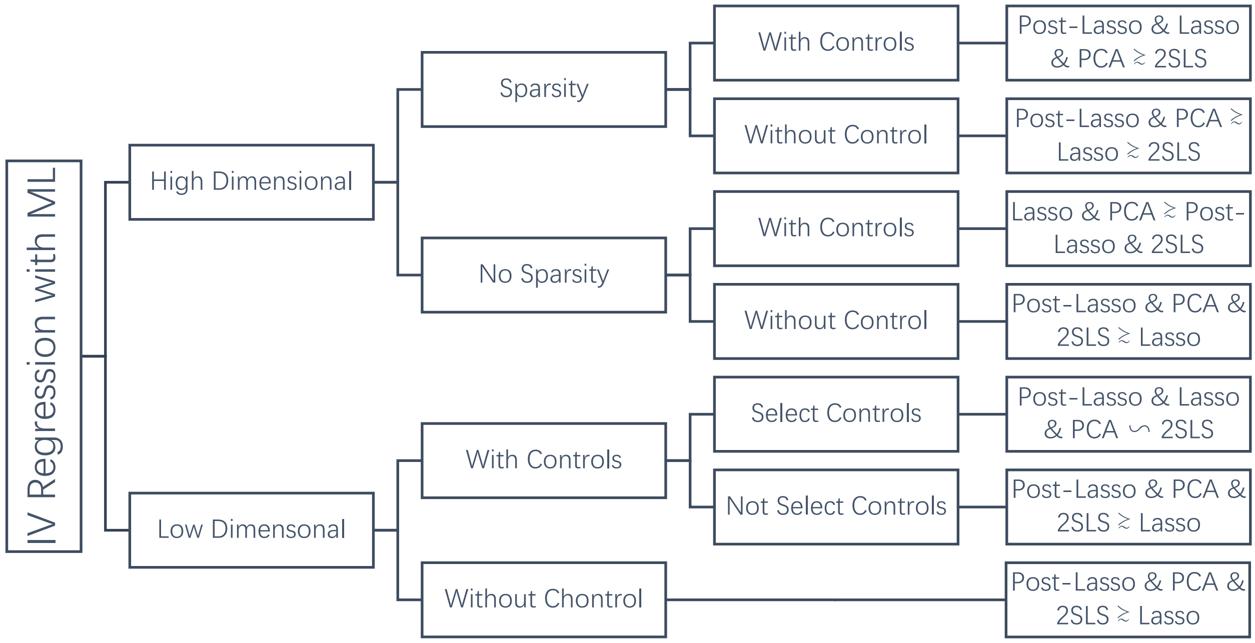




The simulation results of adjusted 4.4 is shown in figure 20, we can conclude that given a sparse instrument variable space (which is what *Belloni et al.(2012)* assumed), ML-based IV regressions outperform traditional 2SLS. Meanwhile, Lasso performs relatively worse than Post-Lasso and PCA. This could be caused by the same reason explained in the end of section 4.4.

5 Conclusion

Figure 22: Basic Conclusion Illustration


IV Regression with ML
High Dimensional
Low Dimensonal
Sparsity
No Sparsity
With Controls
Without Chontrol
With Controls
Without Control
With Controls
Without Control
Select Controls
Not Select Controls
Post-Lasso & Lasso & PCA ≳ 2SLS
Post-Lasso & PCA ≳ Lasso ≳ 2SLS
Lasso & PCA ≳ Post-Lasso & 2SLS
Post-Lasso & PCA & 2SLS ≳ Lasso
Post-Lasso & Lasso & PCA ∽ 2SLS
Post-Lasso & PCA & 2SLS ≳ Lasso
Post-Lasso & PCA & 2SLS ≳ Lasso

Firstly, this simulation study shows that, given a high dimensional instrument space, and the assumption of sparsity holds, Lasso, Post-Lasso and PCA outperform 2SLS, in other words, these three ML-based algorithms output more accurate causal coefficient than traditional 2SLS, this is shown in section 4.6.4. The second, given multiple controls and the assumption of sparsity, *Chernozhukov et al.(2015)*'s "double selection" algorithm ensures Post-Lasso and Lasso outperform 2SLS with a lower variance than PCA, shown in Figure 17. Meanwhile, if there are no controls, Figure 20 (section 4.6.4) shows that given assumption of sparsity holds, all three ML IV regressions outperform 2SLS, but Lasso is more biased than the other two, probably caused by the bias explained in section 4.4 by *Lennon, Rubin & Waddell(2021)*. Besides, if the assumption of sparsity does not hold (section 4.4, no controls), shown in Figure 7, Post-Lasso and PCA perform similar with 2SLS, and Lasso is biased as naïve OLS (Figure 8), caused by the bias explained in 4.4 and the violation of assumption. Finally, in the case of without sparsity and have controls, by using “double selection” algorithm, the result is shown in Figure 21 at Appendix, all three ML IV regressions perform similar with 2SLS, and in Figure 21.2 Lasso and PCA only shows a little advantage over 2SLS and Lasso, this tells us that the “double selection” algorithm highly depends on the assumption of sparsity.

We also could conclude that in a low dimensional case[3], firstly, if there is no control variable (section 4.1, 4.2, 4.3), Post-Lasso and PCA perform at least as good as 2SLS, and they are more centered at true coefficient (Figure 1, Figure 3, Figure 5), even their variances are similar with 2SLS. At the same time, Lasso is relatively biased, shown in Figure 1, 3 & 5, this should have the same reason explained in 4.4. The second point, given control variables, and only selecting instrument variables (section 4.5), Post-Lasso and PCA perform similarly with 2SLS, but Lasso is strongly biased, in both the case of allowing endogenous controls (Figure 10) and not allowing (Figure 13). The third, when selecting controls as well, we should use the “double selection” algorithm in *Chernozhukov et al.(2015)* (section 4.6.3), and no matter how high the correlation between controls and instruments is (Figure 14 & 16), three ML-based IV regressions perform similarly with 2SLS.

In summary, firstly, the low dimensional case is usually what an economics researcher faces, and in such case, if there are control variables available (of course this happens most often), then Post-Lasso, PCA and Lasso are valid in IV, but these ML-based IV do not show superiority over traditional 2SLS, and in such situation, researcher should use the “double selection” algorithm proposed by *Chernozhukov et al.(2015)*, if not, then Lasso outputs a biased estimate. Meanwhile, in such low dimensional case, ML methods are usually unnecessary. In addition, remember that if there is no control variable, Lasso is the inferior choice.

Secondly, if one economic researcher “luckily” faces a high dimensional instrument space, PCA is always an option. Here we assume that our researcher have a big set of instruments, but actually only a small number of them truly matter (strong), then we come to the sparse case, under this condition, traditional 2SLS is heavily biased, then Post-Lasso is a good option, if there are controls (of course this mostly happens), researcher should use *Chernozhukov et al.(2015)*’s algorithm, and Lasso performs as good as Post-Lasso. Now assume there are a lot of strong instruments, and the number of strong instruments is not far from the number of observations, then if there are no controls (unlikely to happen), Post-Lasso is a preferable option, but Post-Lasso and PCA will not outperform 2SLS. Remember that without controls, Lasso outputs biased result. Finally in the case of too many strong instruments (no sparsity), and with controls, *Chernozhukov et al.(2015)*’s algorithm does not show significant advantage, researcher should try to find other methods for IV estimation.

[3] Here we do not specify any assumption on sparsity, because in low dimension instrument space, this assumption holds.

Appendix

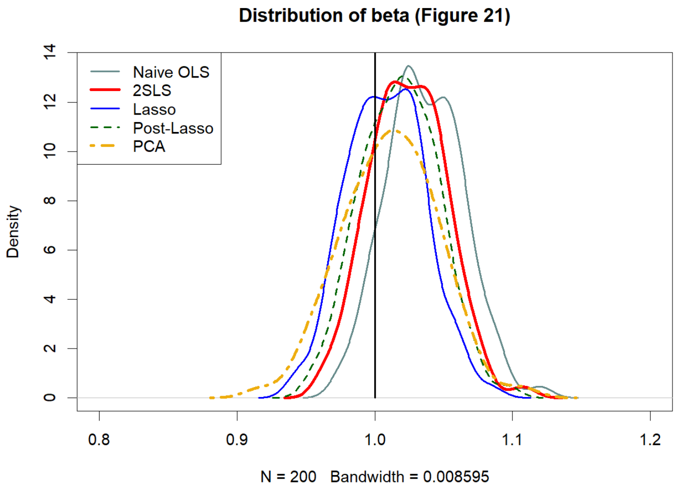
Plotting Codes:

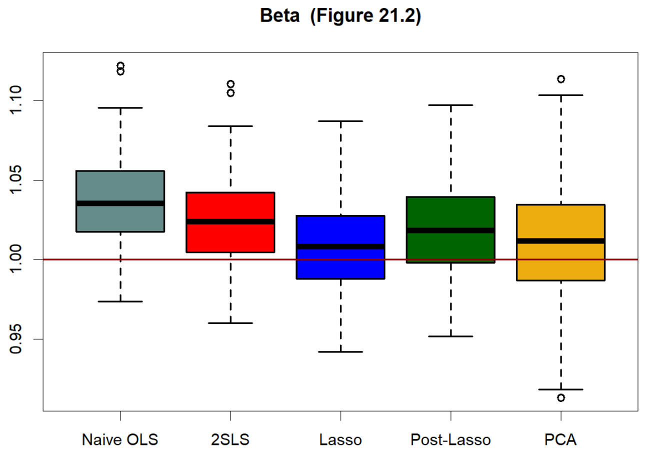
```r
# Figure 1
plot(density(Coef[,1]),lwd=2, col="paleturquoise4",main = "Distribution of beta (Figure 1)")
lines(density(Coef[,2]),col="red",lwd=3)
lines(density(Coef[,3]),col="blue",lwd=2)
lines(density(Coef[,4]),col="darkgreen",lwd=2,lty=2)
lines(density(Coef[,5]),col="darkgoldenrod2",lwd=3,lty=4)
legend("topright", c("Naive OLS", "2SLS", "Lasso","Post-Lasso","PCA"), col = c("paleturquoise4","red","blue","darkgreen","darkgoldenrod2"), lty = c(1,1,1,2,4),lwd=c(2,3,2,2,3))
lines(rep(1,11),seq(0,10,by=1),lwd=2)
# Figure 2
boxplot(Coef,col = c("paleturquoise4","red","blue","darkgreen","darkgoldenrod2"), main="Beta (Figure 2)")
abline(1,0,col="darkred",lwd=2)
```

Figure 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20 and 21 have similar codes.

```r
# Figure 7
barplot(alpha,ylim = c(0,1),names.arg = seq(1,60,by=1),
        xlab = "z",ylab = "alpha",
        main = "alpha for each z  (Figure 7)")
```

Figure 19 has similar code.





Simulation Codes:

```
# Case 2--weak instruments
T <- 200
Coef2 <- matrix(NA,T,5)
colnames(Coef2) <- c("Naive OLS","2SLS","Lasso","Post-Lasso","PCA")
set.seed(527)
for (i in 1:T) {
 training <- dgp_low2(n,mu,sigmaz)
 colnames(training) <- c("y","x","z1","z2","z3","z4","z5","z6","z7")
 Instruments_tr <- as.matrix(training[,3:9])
 ## benchmark OLS and IV
 NaiveOLS <- lm(y~x,data=training)
 Coef2[i,1] <- coef(NaiveOLS)[2]
 IVbenchmark <- ivreg(training$y~training$x|Instruments_tr)
 Coef2[i,2] <- coef(IVbenchmark)[2]
 ## lasso
 grid <- 10^seq(-3,3,length=200)
 Lasso <- glmnet(Instruments_tr,training$x,
         lambda = grid,alpha=1)
 cv.out<-cv.glmnet(Instruments_tr, training$x,alpha=1)
 best_lambda<-cv.out$lambda.min
 lasso.xhat<-predict(Lasso, s=best_lambda,
           newx = Instruments_tr)
 Lasso.2sls <- lm(training$y~lasso.xhat)
 Coef2[i,3] <- coef(Lasso.2sls)[2]
 ## post-lasso
 cv.out.post<-cv.glmnet(Instruments_tr, training$x,
             alpha=1)
 best_lambda<-cv.out.post$lambda.min
 Lasso.post <- glmnet(Instruments_tr,training$x,
            lambda = best_lambda,alpha=1)
 coelasso <- coef(Lasso.post)
 select <- ifelse(coelasso==0,FALSE,TRUE)[-1]
 Instruments_tr2 <- as.matrix(training[,3:9][,select])
 IVpostLasso <- ivreg(training$y~training$x|Instruments_tr2)
 Coef2[i,4] <- coef(IVpostLasso)[2]
 # PCA
 PCR<-pcr(training$x~Instruments_tr,
      scale=TRUE,validation="CV")
 cverr <- RMSEP(PCR)$val[1,,-1]
 imin<-which.min(cverr)
 PCA_IV <- PCR$scores[,1:imin]
 IVPCR <- ivreg(training$y~training$x|PCA_IV)
 Coef2[i,5] <- coef(IVPCR)[2]
}
```

Case 3, 4, 5, 6 have similar code.

References

Lennon, C., Rubin, E., & Waddell, G. (2021). *What can we machine learn (too much of) in 2SLS? Insights from a bias decomposition and simulation, Working Paper.*

Pushan Dutt, Ilia Tsetlin (2020). *Income distribution and economic development: Insights from machine learning, Economics & Politics Vol.33*

Victor Chernozhukov, Christian Hansen, Martin Spindler (2015). *Post-Selection and Post-Regularization Inference in Linear Models with Many Controls and Instruments, American Economic Review 2015, Papers and Proceedings*

Alexandre Belloni, Daniel Chen, Victor Chernozhukov, Christian Hansen. (2012). *Sparse Models and Methods for Optimal Instruments with an Application to Eminent Domain, Econometrica Vol.80*