# Motivation: Confounders and Control Variables

Probably most empirical economics papers are interested in estimating a causal effect $\alpha$ of some explanatory variable $d$ on an dependent variable $y$. (Note that $d$ is not necessarily a dummy, I just stick to prevalent notation in the literature). If our data does not come from a randomized experiment this is generally a hard task because there are essentially always some confounders. Unless we appropriately control for confounders (or circumvent them e.g. by instrumental variable estimation), we get an inconsistent estimator $\hat \alpha$ of the causal effect.

Consider the following simple example where our OLS estimator $\hat \alpha$ has a positive bias if we don’t control for the confounder xc:

n = 10000

xc = rnorm(n,0,1) # confounder d = xc + rnorm(n,0,1)

alpha = 1; beta = 1;

y = alpha\*d + beta \* xc + rnorm(n, 0,1)

# OLS regression without controlling for confounder coef(lm(y~d))[2]

## d

## 1.485069

If we add the confounder as control variable, we get an unbiased OLS estimator:

coef(lm(y~d+xc))[2] ## d

## 0.9987413

In empirical studies, the main problem is typically that we just don’t have data for all relevant confounders. However, there are some applications where we don’t have too many observations but a lot of potential control variables, in particular if we also consider non-linear effects and interaction terms. An example can be found in the vignette (Chernozhukov, Hansen and Spindler, 2016) of the hdm package (which implements the post-double-selection from Belloni et. al.). The example is based on the empirical growth literature, which tries to estimate the causal effect of some variable, like initial GDP, on GDP growth. Gross country panel data sets are not very large, but we can observe a lot of variables for a country. If one is optimistic, one may believe that some subset of the potential control variables is able to effectively control for all confounders.

But if we don’t have enough observations, we need some method to select the relevant control variables.

# Framework of Monte Carlo Simulation

Let us study different approaches for variable selection using a Monte-Carlo simulation with the following data generating process for $y$ and $d$. (If Mathjax is not well rendered on a blog aggregator click here.):

\[y = \alpha d + \sum\_{k=1}^{K\_c} \beta^{cy}\_k {x^c\_k} + \sum\_{k=1}^{K\_y} \beta^y\_k {x^y\_k} +

\varepsilon^y\]

\[d = \sum\_{k=1}^{K\_c} \beta^{cd}\_k {x^c\_k} + \sum\_{k=1}^{K\_e} \beta^e\_k {x^e\_k} + \varepsilon^d\]

We have the following potential control variables that are all independently normally distributed from each other:

$x^c\_k$ is one of $K\_c$ confounders that directly affect both $d$ and $y$. For a consistent OLS estimation of $\alpha$, we need to control for all confounders.

$x^y\_k$ is one of $K\_y$ variables that only affect the dependent variable $y$ but not the explanatory variable $d$. Whether we would add it or not in an OLS regression should not affect the bias of our estimator $\hat \alpha$.

$x^e\_k$ is one of $K\_e$ variables that only affect $d$ but not through any other channel the dependent variable $y$. It constitutes a *source of exogenous variation* in $d$. We can estimate in an OLS regression $\alpha$ more precisely if we don’t add any $x^e\_k$ to the regression. Also, we will see that adding fewer $x^e\_k$ can reduce the bias of an OLS estimator that arises if we have not perfectly controlled for all confounders.

We also observe $K\_u$ variables $x^u\_k$ that neither affect $y$ nor $d$. They are just uncorrelated noise variables that we ideally leave out of our regressions.

The following code simulates a data set with $n=1000$ observations, $K\_c=50$ confounders, $K\_y=50$ variables that affect only $y$, just a single observed variable $x^e\_k$ that provides a source of exogenous variation and $K\_u=1000$ explanatory variables that are uncorrelated with everything else. The causal effect of interest $\alpha$, as well as all other regression coefficients and standard deviations are equal to 1.

Hdm-growth.R

|  |
| --- |
| library(hdm) |
|  | library(restorepoint) |
|  | source("lasso\_tools.R") |
|  | source("orthoML.R") |
|  |  |
|  |  |
|  | data(GrowthData) |
|  | dim(GrowthData) |
|  | ## [1] 90 63 |
|  | y = GrowthData[, 1, drop = T] |
|  | d = GrowthData[, 3, drop = T] |
|  | X = as.matrix(GrowthData)[, -c(1, 2, 3)] |
|  | dX = as.matrix(GrowthData)[, -c(1, 2)] |
|  |  |
|  | varnames = colnames(GrowthData) |
|  | dvar = varnames[3] |
|  | xnames = varnames[-c(1, 2, 3)] # names of X variables |
|  | dandxnames = varnames[-c(1, 2)] # names of D and X variables |
|  |  |
|  |  |
|  | # create formulas by pasting names (this saves typing times) |
|  | fmla = as.formula(paste("Outcome ~ ", paste(dandxnames, collapse = "+"))) |
|  | ls.effect = lm(fmla, data = GrowthData) |
|  | coef(ls.effect)[2] |
|  |  |
|  | # Double selection |
|  | lasso.effect = rlassoEffect(x = X, y = y, d = d, method = "double selection") |
|  | summary(lasso.effect) # 0.05001 |
|  |  |
|  | # Partialling out |
|  | lasso.effect = rlassoEffect(x = X, y = y, d = d, method = "partialling out") |
|  | summary(lasso.effect) # 0.04981 |
|  |  |
|  | # Normal lasso using gamlr |
|  | lasso.gamlr = gamlr(x=dX,y=y,free=1) |
|  | post\_lasso\_coef(lasso.gamlr, dX,y)[1] # -0.05023 |
|  |  |
|  | # Taddy's orthoML |
|  | set.seed(1) |
|  | orthoML( x=X, d=d, y=y, nfold=5)$coef # -0.0657 |
|  |  |
|  |  |
|  | # As a test custom double selection |
|  | double\_selection(x=X,y=y,d=d, method="hdm")[1] |
|  | double\_selection(x=X,y=y,d=d, method="gamlr")[1] |

Lasso\_large\_MC.R

|  |
| --- |
|  |
| # These functions perform the large simulations |
|  | # whose results are shown in the blog |
|  |  |
|  | # Both take quite a while to run |
|  |  |
|  | control.exo.sim = function() { |
|  | source("lasso\_tools.R") |
|  | source("lasso\_sim.R") |
|  | set.seed(1) |
|  |  |
|  | sim.and.est1 = function() { |
|  | #mat = lasso\_sim(alpha=1,n=800,Kc=50,Ke=50,Ky=50,Ku=500,return.what = "data") |
|  | mat = lasso\_sim(alpha=1,n=700,Kc=50,Ke=50,Ky=50,Ku=700,return.what = "data") |
|  |  |
|  |  |
|  | y = mat[,1] |
|  | d = mat[,2] |
|  | X = cbind(1,mat[,-1]); colnames(X)[1] = "const" |
|  |  |
|  | xc.cols = 3:52; xe.cols = 53:102 |
|  | c( |
|  | coef(lm.fit(y=y,x=X[,c(1:2, xc.cols[1:49])]))[2], |
|  | coef(lm.fit(y=y,x=X[,c(1:2, xc.cols[1:49], xe.cols)]))[2] |
|  | ) |
|  | } |
|  | library(dplyr) |
|  | res = replicate(1000,sim.and.est1(),simplify = FALSE) %>% bind\_rows() |
|  | colnames(res) = c("alpha.hat.no.xe","alpha.hat.control.xe") |
|  |  |
|  | sim = tibble( |
|  | reg=c(rep("dont\_control\_xe",NROW(res)),rep("control\_xe",NROW(res))), |
|  | alpha.hat = c(res[[1]],res[[2]]), |
|  | alpha = 1 |
|  | ) |
|  | saveRDS(sim,"control\_exo\_sim.Rds") |
|  |  |
|  | sim = readRDS("control\_exo\_sim.Rds") |
|  | head(sim,3) |
|  | library(ggplot2) |
|  | ggplot(sim, aes(x=alpha.hat, fill=reg)) + geom\_density() + facet\_wrap(~reg) + geom\_vline(xintercept=1, alpha=0.7) |
|  |  |
|  | sim %>% |
|  | group\_by(reg) %>% |
|  | summarize( |
|  | bias = mean(alpha.hat-alpha), |
|  | rmse = sqrt(mean((alpha.hat-alpha)^2)) |
|  | ) |
|  | } |
|  |  |
|  |  |
|  | # Takes very long to run |
|  | lasso.sel.sim = function() { |
|  | source("lasso\_tools.R") |
|  | source("lasso\_sim.R") |
|  |  |
|  | models1 = lasso\_sim\_default\_models(names=c("gamlr\_simple","rlasso\_double\_sel")) |
|  |  |
|  | models2 = list( |
|  | rlasso\_double\_sel\_c106 = list(lasso.fun="rlasso",type="double\_sel", args=list(penalty=list(c=1.06))), |
|  | rlasso\_double\_sel\_c100 = list(lasso.fun="rlasso",type="double\_sel", args=list(penalty=list(c=1))) |
|  | ) |
|  |  |
|  | models = c(models1, models2) |
|  | library(dplyr) |
|  | li = replicate(n=1000, lasso\_sim(alpha=1,n=700,Kc=50,Ke=50,Ky=50,Ku=700,return.what = "details",models = models),simplify = FALSE) |
|  |  |
|  | sim = bind\_rows(li) |
|  | saveRDS(sim, "lasso\_sel\_sim.Rds") |
|  |  |
|  | sim %>% |
|  | group\_by(model) %>% |
|  | summarize( |
|  | bias = mean(coef-1), |
|  | se = sd(coef) |
|  | ) |
|  | library(ggplot2) |
|  | ggplot(sim, aes(x=coef,fill=model)) + geom\_density() + |
|  | facet\_wrap(~model, scales="free\_y") |
|  | } |
|  |  |
|  |  |
|  | # Takes very long to run |
|  | lasso.sel.sim = function() { |
|  | source("lasso\_tools.R") |
|  | source("lasso\_sim.R") |
|  |  |
|  | models1 = lasso\_sim\_default\_models(names=c("gamlr\_simple","rlasso\_double\_sel")) |
|  |  |
|  | models2 = list( |
|  | rlasso\_double\_sel\_c106 = list(lasso.fun="rlasso",type="double\_sel", args=list(penalty=list(c=1.06))), |
|  | rlasso\_double\_sel\_c100 = list(lasso.fun="rlasso",type="double\_sel", args=list(penalty=list(c=1))) |
|  | ) |
|  |  |
|  | models = c(models1, models2) |
|  | library(dplyr) |
|  | li = replicate(n=1000, lasso\_sim(alpha=1,n=100,Kc=10,Ke=10,Ky=5,Ku=20, beta.ed = 10, beta.ey = 0.5, beta.cd = 0.5, beta.cy = 10, models=c("gamlr\_simple","rlasso\_double\_sel")) |
|  | ,simplify = FALSE) |
|  |  |
|  | sim = bind\_rows(li) |
|  | saveRDS(sim, "lasso\_sim3.Rds") |
|  |  |
|  | sim = readRDS("lasso\_sim3.Rds") |
|  | sim %>% |
|  | group\_by(model) %>% |
|  | summarize( |
|  | bias = mean(coef-1), |
|  | se = sd(coef), |
|  | num.vars = mean(num.vars), |
|  | xe = mean(xe), |
|  | xc = mean(xc) |
|  | ) |
|  | library(ggplot2) |
|  |  |
|  | ggplot(sim, aes(x=coef,fill=model)) + geom\_density() + |
|  | facet\_wrap(~model, scales="free\_y") |
|  | } |
|  |  |

Lasso\_SIM.R

|  |
| --- |
| example = function() { |
|  | # Try to find zero rlasso selection |
|  | set.seed(9) |
|  | lasso\_sim(alpha=1,n=500,Kc=50,Ke=50,Ky=50,Ku=500,sd.noise=0.1) |
|  |  |
|  | set.seed(9) |
|  | mat = lasso\_sim(alpha=1,n=500,Kc=50,Ke=50,Ky=50,Ku=500,sd.noise=0.1,return.what = "data") |
|  |  |
|  | y = mat[,1] |
|  | d = mat[,2] |
|  | X = cbind(1,mat[,-1]); colnames(X)[1] = "const" |
|  |  |
|  | lasso1 = rlasso(x=X[,-2],y=d) |
|  | coef(lasso1)[coef(lasso1)!=0] |
|  |  |
|  | lasso2 = rlasso(x=X[,-2],y=y) |
|  | coef(lasso2)[coef(lasso2)!=0] |
|  |  |
|  |  |
|  | res = double\_selection(d=d,y=y,x=X[,-2],lasso.fun = "rlasso") |
|  | res |
|  |  |
|  | # Find a good example without were simple lasso works and |
|  | set.seed(1) |
|  | lasso\_sim(alpha=1,n=700,Kc=50,Ke=50,Ky=50,Ku=700,return.what = "details") |
|  |  |
|  | n = 700 |
|  | penalty = list(homoscedastic = FALSE, |
|  | X.dependent.lambda = FALSE, lambda.start = NULL, c = 1.1, gamma = 0.1/log(n)) |
|  |  |
|  | models = list( |
|  | rlasso\_double\_sel\_c106 = list(lasso.fun="rlasso",type="double\_sel", args=list(penalty=list(c=1.06))), |
|  | rlasso\_double\_sel\_c100 = list(lasso.fun="rlasso",type="double\_sel", args=list(penalty=list(c=1))) |
|  | ) |
|  | set.seed(1) |
|  | lasso\_sim(alpha=1,n=700,Kc=50,Ke=50,Ky=50,Ku=700,return.what = "details",models = models) |
|  | rlasso |
|  |  |
|  | lasso\_sim(alpha=1,n=500,Kc=30,Ke=30,Ky=30,Ku=300,return.what = "details") |
|  |  |
|  |  |
|  | cor(mat[,1],mat[,3]) |
|  | cor(mat[,1],mat[,4]) |
|  | cor(mat[,1],mat[,5]) |
|  | cor(mat[,1],mat[,6]) |
|  |  |
|  |  |
|  | beta.e = 2 |
|  | # If everything is fairly sparse, it all seems to work |
|  | # including normal gamlr lasso |
|  | lasso\_sim(beta.e = beta.e, Kc=5,Ke=5,Ky=5, Ku=500, n=500) |
|  |  |
|  | # Make uniformely less sparse |
|  | lasso\_sim(beta.e = beta.e,Kc=50,Ke=50,Ky=50, Ku=500, n=500) |
|  | # gamlr still works somewhat but double selection and oml break down |
|  |  |
|  | # Just add a lot of exogenous variables |
|  | lasso\_sim(beta.e = beta.e,Kc=5,Ke=50,Ky=5, Ku=400, n=500) |
|  |  |
|  | # Just add a lot of confounders: gamlr works worst |
|  | lasso\_sim(beta.e = beta.e,Kc=50,Ke=5,Ky=5, Ku=400, n=500) |
|  |  |
|  | # Just add a lot of variables that just affect y |
|  | lasso\_sim(beta.e = beta.e,Kc=5,Ke=5,Ky=50, Ku=400, n=500) |
|  |  |
|  |  |
|  | beta.e = 2 |
|  | # If everything is fairly sparse, it all seems to work |
|  | # including normal gamlr lasso |
|  | lasso\_sim(beta.e = beta.e, Kc=5,Ke=5,Ky=5, Ku=500, n=500) |
|  |  |
|  | # Make uniformely less sparse |
|  | lasso\_sim(beta.e = beta.e,Kc=50,Ke=50,Ky=50, Ku=500, n=500) |
|  | lasso\_sim(beta.e = beta.e,Kc=50,Ke=50,Ky=50, Ku=500, n=500) |
|  | # gamlr still works somewhat but double selection and oml break down |
|  |  |
|  | # Just add a lot of exogenous variables |
|  | lasso\_sim(beta.e = beta.e,Kc=5,Ke=50,Ky=5, Ku=400, n=500) |
|  |  |
|  | # Just add a lot of variables that just affect y |
|  | lasso\_sim(beta.e = beta.e,Kc=5,Ke=5,Ky=50, Ku=400, n=500) |
|  |  |
|  |  |
|  | # Just add a lot of confounders: gamlr works worst if sd.x is small |
|  | lasso\_sim(beta.e = beta.e,Kc=50,Ke=5,Ky=5, Ku=400, n=500) |
|  | lasso\_sim(beta.e = beta.e,sd.d=5,Kc=50,Ke=5,Ky=5, Ku=400, n=500) |
|  | lasso\_sim(beta.e = beta.e,sd.d=1,sd.beta=4,Kc=50,Ke=5,Ky=5, Ku=400, n=500) |
|  |  |
|  | # Just add a lot of confounders: gamlr works worst if sd.x is small |
|  | lasso\_sim(beta.e = 1,sd.d = 1,Kc=50,Ke=5,Ky=50, Ku=400, n=500) |
|  | lasso\_sim(beta.e = 1,sd.d = 1,sd.beta=4,Kc=50,Ke=5,Ky=50, Ku=400, n=500) |
|  | lasso\_sim(beta.e = 1,sd.d = 10,Kc=50,Ke=1,Ky=50, Ku=400, n=500) |
|  |  |
|  | # Just add a lot of confounders: gamlr works worst |
|  | lasso\_sim(beta.e = beta.e,beta.cd = 5,Kc=50,Ke=5,Ky=50, Ku=400, n=500) |
|  |  |
|  |  |
|  |  |
|  | # Normal lasso with gamlr works very well but hdm not |
|  |  |
|  | lasso\_sim(alpha = 1,beta.e = 1.5,beta.y = 1,Kc=50,Ke=50,Ky=50, Ku=400, n=500) |
|  |  |
|  |  |
|  | lasso\_sim(alpha = 1,beta.e = 0.8,beta.y = 1,Kc=5,Ke=5,Ky=5, Ku=400, n=500) |
|  |  |
|  |  |
|  | # Where does hdm work better than gamlr? |
|  | lasso\_sim(alpha = 1,beta.e = 0,beta.y = 1.5,sd.d = 10,Kc=30,Ke=1,Ky=500, Ku=500, n=600) |
|  |  |
|  | lasso\_sim(alpha = 1,beta.e = 0,beta.y = 1.5,sd.d = 10,Kc=30,Ke=1,Ky=500, Ku=500, n=2000) |
|  |  |
|  |  |
|  | # Some simulations where both xe and xc are confounders (last chapter |
|  | # of blog) |
|  | set.seed(1) |
|  | lasso\_sim(alpha=1,n=100,Kc=20,Ke=5,Ky=5,Ku=20, beta.e = 20, beta.ey = 0.5, beta.cd = 1, beta.cy = 2) |
|  |  |
|  | set.seed(1) |
|  | lasso\_sim(alpha=1,n=100,Kc=10,Ke=10,Ky=5,Ku=20, beta.e = 10, beta.ey = 0.5, beta.cd = 0.5, beta.cy = 10) |
|  |  |
|  | } |
|  |  |
|  |  |
|  | lasso\_sim = function( |
|  | n = 500, |
|  | K = 500, |
|  | Kc = round(K/4), |
|  | Ke = round(K/4), |
|  | Ku = round(K/4), |
|  | Ky = round(K/4), |
|  | sd.xc = 1,sd.xe = 1,sd.xu = 1,sd.xy = 1, |
|  | sd.d = 1, sd.y = 1, sd.beta = 0, sd.noise = 0, |
|  | alpha = 1,beta=1, |
|  | beta.e = beta.ed, beta.cd = beta, beta.cy = beta, beta.y = beta, beta.ey = 0, beta.ed=beta, |
|  | return.what = c("data","coef","details")[3], |
|  | models = c("short\_ols", "gamlr\_simple","rlasso\_double\_sel","gamlr\_double\_sel"), |
|  | count.what = "vars" |
|  | ) { |
|  | library(restorepoint); library(dplyr) |
|  | restore.point("lasso\_sim") |
|  |  |
|  | K.vec = c(Kc=Kc, Ke=Ke,Ku=Ku,Ky=Ky) |
|  | Xc = rnorm(Kc\*n, 0, sd.xc) %>% matrix(nrow=n) |
|  | Xe = rnorm(Ke\*n, 0, sd.xe) %>% matrix(nrow=n) |
|  | Xu = rnorm(Ku\*n, 0, sd.xu) %>% matrix(nrow=n) |
|  | Xy = rnorm(Ky\*n, 0, sd.xy) %>% matrix(nrow=n) |
|  |  |
|  | d = Xe %\*% rnorm(Ke,beta.e, sd.beta) + |
|  | Xc %\*% rnorm(Kc,beta.cd, sd.beta) + rnorm(n, 0, sd.d) |
|  | y = alpha\*d + Xy %\*% rnorm(Ky,beta.y,sd.beta) + Xc %\*% rnorm(Kc,beta.cy, sd.beta) + Xe %\*% rep(beta.ey,Ke) + rnorm(n, 0, sd.y) |
|  | eps = y-alpha\*d |
|  |  |
|  | X = cbind(Xc,Xe,Xu,Xy) |
|  | if (sd.noise > 0) { |
|  | noise = rnorm(length(X),0,sd.noise) |
|  | X = X+noise |
|  | } |
|  |  |
|  | colnames(X) = c(paste0("xc",1:Kc),paste0("xe",1:Ke), paste0("xu",1:Ku), paste0("xy",1:Ky)) |
|  |  |
|  | if (return.what=="data") { |
|  | mat = cbind(y,d,X) |
|  | colnames(mat)[1:2] = c("y","d") |
|  | return(mat) |
|  | } |
|  |  |
|  | lasso\_sim\_estimate(models=models, y=y,d=d,X=X, count.what=count.what, return.what=return.what) |
|  | } |
|  |  |
|  |  |
|  |  |
|  | lasso\_sim\_default\_models = function(names) { |
|  | models = list( |
|  | short\_ols = list(type="short\_ols"), |
|  | gamlr\_free\_d = list(lasso.fun="gamlr",type="post\_lasso", args=list(free=1)), |
|  | gamlr\_simple = list(lasso.fun="gamlr",type="post\_lasso"), |
|  | rlasso\_simple = list(lasso.fun="rlasso",type="pl"), |
|  | rlasso\_double\_sel = list(lasso.fun="rlasso",type="double\_sel"), |
|  | gamlr\_double\_sel = list(lasso.fun="gamlr",type="double\_sel") |
|  | ) |
|  |  |
|  | if (!missing(names)) { |
|  | unknown = setdiff(names, names(models)) |
|  | if (length(unknown) > 0) |
|  | stop(paste0("No model with name ", unknown[1], " specified in lasso\_sim\_default\_models.")) |
|  | models = models[names] |
|  | } |
|  | models |
|  | } |
|  |  |
|  | lasso\_sim\_estimate = function(mat, models=lasso\_sim\_default\_models(),y,d,X, return.what ="details", count.what="vars") { |
|  | if (missing(y)) { |
|  | if (missing(mat)) { |
|  | stop("You either have to provide mat or the three arguments y,d,X") |
|  | } |
|  | y = mat[,1] |
|  | d = mat[,2] |
|  | X = mat[, -c(1:2)] |
|  | dX = mat[,-1] |
|  | } else { |
|  | dX = cbind(d,X) |
|  | colnames(dX)[1] = "d" |
|  | } |
|  | restore.point("lasso\_sim\_estimate") |
|  |  |
|  | if (is.character(models)) { |
|  | models = lasso\_sim\_default\_models(models) |
|  | } |
|  |  |
|  |  |
|  | library(gamlr) |
|  | library(hdm) |
|  | model\_names = names(models) |
|  | res.li = lapply(models, function(model) { |
|  | lasso\_sim\_estimate\_model(model,y=y,d=d,X=X,dX=dX) |
|  | }) |
|  |  |
|  | coefs = unlist(lapply(res.li, function(res) res$coef), use.names=FALSE) |
|  | names(coefs) = names(res.li) |
|  | if (return.what =="coef") { |
|  | return(coefs) |
|  | } |
|  |  |
|  | res = res.li[[1]] |
|  |  |
|  | counts = bind\_rows(lapply(res.li, function(res) { |
|  |  |
|  | vs = if ("vars" %in% count.what) vars\_counts(res$vars) |
|  | vs1 = if ("vars1" %in% count.what) vars\_counts(res$vars1) |
|  | vs2 = if ("vars2" %in% count.what) vars\_counts(res$vars2) |
|  | if (!is.null(vs1)) names(vs1) = paste0(names(vs1),"1") |
|  | if (!is.null(vs2)) names(vs2) = paste0(names(vs2),"2") |
|  | c(vs,vs1,vs2) |
|  | })) |
|  |  |
|  | cbind(quick\_df( |
|  | model = names(models), |
|  | coef = coefs |
|  | ),counts) |
|  | } |
|  |  |
|  | lasso\_sim\_estimate\_model = function(model, d,y,X,dX, details = TRUE) { |
|  | restore.point("estimate\_lasso\_sim\_model") |
|  |  |
|  | # post lasso with d as free variable |
|  | # post lasso without free variable in intia lasso |
|  | if (model$type == "post\_lasso") { |
|  | args = c(list(x=dX,y=y), model$args) |
|  | if (model$lasso.fun=="gamlr") { |
|  | reg = do.call(gamlr, args) |
|  | } else if (model$lasso.fun == "rlasso") { |
|  | reg = do.call(rlasso, args) |
|  | } |
|  | coefs = post\_lasso\_coef(reg,dX,y,add.var = "d") |
|  | list(coef = coefs[1], vars = names(coefs)[-1], vars1=NULL, vars2=names(coefs)[-1]) |
|  | } else if (model$type == "double\_sel") { |
|  | args = c(list(d=d,x=X,y=y,lasso.fun = model$lasso.fun, just.d.coef = TRUE), model$args) |
|  | res = do.call(double\_selection, args) |
|  | } else if (model$type == "short\_ols") { |
|  | coef = coef(lm.fit(y=y,x=cbind(1,d)))[2] |
|  | list(coef=coef, vars=NULL) |
|  | } else { |
|  | stop(paste0("Model type ", model$type ," not known. So far only model types double\_sel, lasso and post\_lasso are implemented.")) |
|  | } |
|  | } |
|  |  |
|  |  |
|  | vars\_counts = function(vars) { |
|  | if (is.null(vars)) return(c(num.vars=0, xc=0,xe=0,xu=0,xy=0)) |
|  |  |
|  |  |
|  | if (!is.character(vars)) { |
|  | vars = names(lasso\_coef(vars)) |
|  | } |
|  |  |
|  | res = c( |
|  | xc = sum(startsWith(vars,"xc")), |
|  | xe=sum(startsWith(vars,"xe")), |
|  | xu=sum(startsWith(vars,"xu")), |
|  | xy = sum(startsWith(vars,"xy")) |
|  | ) |
|  | count = sum(res) |
|  | c(num.vars=count,res) |
|  | } |

Lasso\_tools.R

|  |
| --- |
| library(hdm) |
|  | library(gamlr) |
|  | library(restorepoint) |
|  | library(dplyr) |
|  |  |
|  |  |
|  | # Return non-zero lasso coefficients |
|  | lasso\_coef = function(lasso,..., keep.intercept=FALSE) { |
|  | restore.point("lasso\_coef") |
|  | if (is(lasso,"rlasso")) { |
|  | co = coef(lasso) |
|  | if (!keep.intercept) { |
|  | if (names(co)[1]=="(Intercept)") co = co[-1] |
|  | co = co[co!=0] |
|  | return(co) |
|  | } |
|  | } else if (is(lasso, "cv.glmnet")) { |
|  | co = as.matrix(coef(lasso, s = "lambda.min")) |
|  |  |
|  | } else if (is(lasso,"glmnet")) { |
|  | stop("Not yet implemented for glmnet. Please call cv.glmnet.") |
|  | } else if (is(lasso, "gamlr")) { |
|  | co = as.matrix(coef(lasso,...)) |
|  | } else { |
|  | stop("Invalid lasso object passed. It must be the return value of gamlr, rlasso, glmnet or cv.glmnet.") |
|  | } |
|  | rows = co[,1] != 0 |
|  | rows[1] = keep.intercept |
|  | rows |
|  | co = co[rows,] |
|  | co |
|  | } |
|  |  |
|  | # Get the (non-zero) post lasso coefficients |
|  | post\_lasso\_coef = function(lasso, x,y, add.var=NULL, keep.intercept=FALSE) { |
|  | restore.point("post\_lasso\_coef") |
|  | co = lasso\_coef(lasso) |
|  | vars = unique(c(add.var,names(co))) |
|  | x = as.matrix(x[,vars]) |
|  | reg = lm.fit(x = cbind(1,x),y=y) |
|  | co = coef(reg) |
|  | if (!keep.intercept) co = co[-1] |
|  | co |
|  | } |
|  |  |
|  | # Perform double selection lasso |
|  | # similar to hdm::rlassoEffects |
|  | double\_selection = function(d,x,y,..., lasso.fun=c("rlasso","gamlr")[1],dvar="d", keep.intercept=FALSE, just.d.coef = FALSE) { |
|  | args = list(...) |
|  | if (lasso.fun == "rlasso") { |
|  | library(hdm) |
|  | lasso1 = rlasso(x=x,y=d,...) |
|  | lasso2 = rlasso(x=x,y=y,...) |
|  | } else if (lasso.fun=="gamlr") { |
|  | library(gamlr) |
|  | lasso1 = gamlr(x=x,y=d,...) |
|  | lasso2 = gamlr(x=x,y=y,...) |
|  | } |
|  | #restore.point("double\_selection") |
|  | vars1 = names(lasso\_coef(lasso1)) |
|  | vars2 = names(lasso\_coef(lasso2)) |
|  | vars = union(vars1,vars2) |
|  | X = cbind(1,d,x[,vars,drop=FALSE]) |
|  | if (NCOL(X)>2) { |
|  | colnames(X)[1:2] = c("(Intercept)",dvar) |
|  | } else { |
|  | colnames(X) = c("(Intercept)",dvar) |
|  | } |
|  | co = coef(lm.fit(x=X,y=y)) |
|  | if (!keep.intercept) co = co[-1] |
|  | if (just.d.coef) co = co[1] |
|  | list(coef = co,vars=vars, vars1=vars1, vars2=vars2) |
|  | } |
|  |  |
|  |  |
|  | # Call cv.glmnet with free variables that will not be penalized |
|  | cv.glmnet.free = function(x,y,free=NULL, ...) { |
|  | library(glmnet) |
|  | if (is.null(free)) { |
|  | cv.glmnet(x=x,y=y,...) |
|  | } else { |
|  | nvars = NCOL(x) |
|  | penalty.factor = rep(1, nvars) |
|  | if (is.character(free)) |
|  | free = match(free, colnames(x)) |
|  | penalty.factor[free] = 0 |
|  | cv.glmnet(x=x,y=y,penalty.factor = penalty.factor) |
|  | } |
|  | } |
|  |  |
|  | # Call glmnet with free variables that will not be penalized |
|  | glmnet.free = function(x,y,free=NULL, ...) { |
|  | library(glmnet) |
|  | if (is.null(free)) { |
|  | glmnet(x=x,y=y,...) |
|  | } else { |
|  | nvars = NCOL(x) |
|  | penalty.factor = rep(1, nvars) |
|  | if (is.character(free)) |
|  | free = match(free, colnames(x)) |
|  | penalty.factor[free] = 0 |
|  | glmnet(x=x,y=y,penalty.factor = penalty.factor) |
|  | } |
|  | } |
|  |  |
|  | # Call an expression with a specific random seed and |
|  | # afterwards restore the previous state of the |
|  | # pseudo-random number generator |
|  | with\_random\_seed = function(expr, seed = 1234567890) |
|  | { |
|  | old.seed = get(".Random.seed", .GlobalEnv) |
|  | set.seed(seed) |
|  | ret = eval(expr) |
|  | assign(".Random.seed", old.seed, .GlobalEnv) |
|  | runif(1) |
|  | return(ret) |
|  | } |
|  |  |
|  | quick\_df = function(...) { |
|  | as\_tibble(list(...)) |
|  | } |

set.seed(1)

ma = lasso\_sim(alpha=1, n=1000,Kc=50,Ke=1,Ky=50,Ku=1000,return.what = "data") dim(mat) # more variables than observations

## [1] 1000 1103

ma [1:3,1:5] # show excerpt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ## | y | d | xc1 | xc2 | xc3 |
| ## | [1,] -1.319487 | -3.570383 | -0.6264538 | 1.1349651 | -0.8861496 |
| ## | [2,] 15.526450 | 5.809369 | 0.1836433 | 1.1119318 | -1.9222549 |
| ## | [3,] 21.525964 | 1.011645 | -0.8356286 | -0.8707776 | 1.6197007 |

Let us first estimate three regressions via OLS:

y = mat[,1]

X = cbind(1,mat[,-1]); colnames(X)[1] = "const"

# Using all x does give non-sensible result # because we have too many variables coef(lm.fit(y=y,x=X))[2]

## d

## 3.112387

# Short regression y ~ d yields positive bias # due to omitted confounders coef(lm.fit(y=y,x=X[,1:2]))[2]

## d

## 1.94576

# Controlling for all confounders eliminates bias coef(lm.fit(y=y,x=X[,1:52]))[2]

## d

## 1.015087

If we know which variables are the confounders, we can easily consistently estimate the causal effect

$\alpha=1$. But let’s assume we don’t know which of the 1101 potential control variables are the confounders.

# Lasso, Post-lasso and Post Double Selection

From the machine learning toolbox, lasso regressions seem natural candidates to select relevant control variables. Consider a linear regression with $K$ standardized explanatory variables with corresponding estimators and residuals denoted by $\hat \beta$ and $\hat \varepsilon(\hat \beta)$, respectively. The lasso estimator solves the following optimization problem:

\[\min\_{\hat \beta} \sum\_{i=1}^n {\hat \varepsilon\_i(\hat \beta)^2} + \lambda \sum\_{k=1}^K {|\hat \beta\_k|}\]

The first term is just the sum of squared residuals, which the OLS estimator minimizes. The second term penalizes larger absolute values of the estimated coefficients. The penalty parameter $\lambda \gt 0$ will be chosen in an outer loop e.g. by cross-validation or using a criterion like the corrected AIC. Lasso estimates typically have many coefficients $\hat \beta\_k$ equal to zero. In this sense the lasso estimator selects a subset of explanatory variables whose estimated coefficients are non-zero.

But also the coefficients of the selected variables will be typically attenuated towards 0 because of the penalty term. The *post-lasso* estimator avoids this attenuation by simply performing an OLS estimation using all the selected variables from the lasso estimation.

The following code estimates a lasso and post-lasso regression for our dependent variable $y$ in our simulated data set and examines the corresponding estimator $\hat \alpha$ of our causal effect of interest.

The gamlr function uses by default the corrected Akaike Information Criterion to select the penalty parameter $\lambda$. This avoids the random fluctuation from cross validation (used in the popular cv.glmnet function) and makes the gamlr function blazingly fast.

# Lasso estimator library(gamlr)

lasso = gamlr(x=X,y=y)

# lasso\_coef is a helper function in lasso\_tools.R # because coef(lasso) returns a sparse matrix coefs = lasso\_coef(lasso,keep.intercept = FALSE) coefs[1] # still biased

## d

## 1.955174

# Post lasso estimator vars = names(coefs)

post.lasso = lm.fit(y=y,x=X[,c("const",vars)]) coef(post.lasso)[2] # post-lasso estimator also still biased

## d

## 1.979171

We see that both the lasso and post-lasso estimators of the causal effect $\alpha$ are still biased. Both are roughly the same size as the estimate in the short OLS regression of $y$ on just $d$. To understand why, let’s have a look at the selected variables:

vars

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ## | [1] | "d" | "xc3" | "xc39" | "xe1" | "xu314" | "xu700" | "xu734" | "xu763" | "xu779" |
| ## | [10] | "xu831" | "xu870" | "xy1" | "xy2" | "xy3" | "xy4" | "xy5" | "xy6" | "xy7" |
| ## | [19] | "xy8" | "xy9" | "xy10" | "xy11" | "xy12" | "xy13" | "xy14" | "xy15" | "xy16" |
| ## | [28] | "xy17" | "xy18" | "xy19" | "xy20" | "xy21" | "xy22" | "xy23" | "xy24" | "xy25" |
| ## | [37] | "xy26" | "xy27" | "xy28" | "xy29" | "xy30" | "xy31" | "xy32" | "xy33" | "xy34" |
| ## | [46] | "xy35" | "xy36" | "xy37" | "xy38" | "xy39" | "xy40" | "xy41" | "xy42" | "xy43" |
| ## | [55] | "xy44" | "xy45" | "xy46" | "xy47" | "xy48" | "xy49" | "xy50" |  |  |

# A helper function to count selected variables by type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| vars\_counts(vars)  ## num.vars | xc | xe | xu | xy |
| ## 60 | 2 | 1 | 7 | 50 |

While the lasso regression selects all 50 xy variables that only affect $y$, it only selects 2 of the 50 confounders xc. So 48 confounders remain uncontrolled and create massive bias. This quite asymmetric selection may be a bit surprising given that in our simulation the confounders xc affect y as strongly as the xy variables ($\beta\_k^{cy} = \beta\_k^{y} = 1$). My intuition is that our (also selected) variable of interest d already captures some effect of the confounders on y. Thus the confounders are less important to predict y and are therefore not selected.

The *post double selection* method by Belloni et. al. (2014) selects the control variables as follows. We run two lasso regressions. The first regresses d on all potential controls. The second regresses y on all potential controls (excluding d). Then we use the union of the selected variables from both lasso regressions for our post-lasso OLS regression. An intuition for this approach is that confounders are variables that affect both d and y. To ensure that very few confounders are omitted, it thus seems not implausible to use as a broad control set all variables that relevantly affect d or y.

Let us apply the post double selection method:

# post double selection

# 1. run the two lasso regressions d = mat[,2]

lasso.dx = gamlr(y=d,x=X[,-2]) lasso.yx = gamlr(y=y,x=X[,-2])

# 2. compute union of selected variables from # both lasso regressions

vars1 = names(lasso\_coef(lasso.dx)) vars2 = names(lasso\_coef(lasso.yx)) vars = union(vars1,vars2)

# 3. Run OLS estimation with d and all selected variables post.lasso = lm.fit(y=y,x=X[,c("const","d",vars)]) coef(post.lasso)[2] # looks fairly unbiased

## d

## 0.9643243

Now our estimator looks fairly unbiased. Let us look at the selected variables:

vars\_counts(vars1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## num.vars | xc | xe | xu | xy |
| ## 148 | 50 | 1 | 96 | 1 |
| vars\_counts(vars2) | | | | |
| ## num.vars | xc | xe | xu | xy |
| ## 207 | 50 | 1 | 106 | 50 |
| vars\_counts(vars) |  |  |  |  |
| ## num.vars | xc | xe | xu | xy |
| ## 271 | 50 | 1 | 170 | 50 |

We see that the post double selection method selects all 50 confounders xc. Interestingly the confounders are found in both the first and second lasso regression. This means if we don’t add the variable of interest d when regressing y on all potential controls, we seem more likely to pick up confounders.

You may think: “But we selected many more variables: 271 instead of only 60 before. In particular, we wrongly picked up 170 unrelated variables. Therefore it is no surprise that we now also select the 50 confounders.”

However, Belloni et. al. (2014) actually use a different method to select the penalty parameter $\lambda$, not the AICc critierion used in gamlr. The method is implemented in the R package hdm authored by Martin Spindler, Victor Chernozhukov and Chris Hansen. Let us repeat the post double selection procedure using the function rlasso from the hdm package:

library(hdm)

# post double selection

# 1. run the two lasso regressions lasso.dx = rlasso(y=d,x=X[,-2]) lasso.yx = rlasso(y=y,x=X[,-2])

# 2. compute union of selected variables from # both lasso regressions

vars1 = names(lasso\_coef(lasso.dx)) vars2 = names(lasso\_coef(lasso.yx)) vars = union(vars1,vars2)

# 3. Run OLS estimation with d and all selected variables post.lasso = lm.fit(y=y,x=X[,c("const","d",vars)]) coef(post.lasso)[2] # looks unbiased

## d

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## 1.013494 |  |  |  |  |
| # Var counts vars\_counts(vars) |  |  |  |  |
| ## num.vars | xc | xe | xu | xy |
| ## 113 | 50 | 1 | 12 | 50 |

We now only select 12 unrelated variables but still all confounders. Correspondingly, our resulting OLS estimate is pretty close to the true causal effect.

The function rlassoEffect is a convenient wrapper to directly apply the double selection method via rlasso: rlassoEffect(x=X[,-2],d=d,y=y,method="double selection")

##

## Call:

## rlassoEffect(x = X[, -2], y = y, d = d, method = "double selection") ##

## Coefficients:

## d1 ## 1.013

The hdm package also implements an alternative “partialling out” method based on an idea related to regression anatomy and the FWL theorem. Here we find:

rlassoEffect(x=X[,-2],d=d,y=y,method="partialling out")

##

## Call:

## rlassoEffect(x = X[, -2], y = y, d = d, method = "partialling out") ##

## Coefficients: ## [1] 0.9402

While the estimated $\hat \alpha$ differs slightly, in all simulations where I checked either both approaches seemed to worked well or both failed.

# Cases were post double selection fails but selection with single lasso regression works well

While in the previous example post double selection worked very well, I believe there are cases in which selection with a single lasso regression on y works better for selecting variables.

We now simulate data from our model with the main difference that we now also add $K\_e=50$ variables that are sources of exogenous variation, i.e. they only affect $d$ but not $y$.

set.seed(1)

ma = lasso\_sim(alpha=1,n=700,Kc=50,Ke=50,Ky=50,Ku=700,return.what = "data") y = mat[,1]

d = mat[,2]

X = cbind(1,mat[,-1]); colnames(X)[1] = "const"

# Short OLS regression y on d: biased coef(lm.fit(y=y,x=X[,1:2]))[2]

## d

## 1.472816

Now let’s compare two linear regressions where in both we select 49 of the 50 confounders. In one regression we also control for all 50 observable sources of exogenous variation xe in the other we don’t add any xe as control variable. Make a guess about the biases of $\hat \alpha$ in the two regressions before you look at the results here:

xc.cols = 3:52; xe.cols = 53:102

# Controlling for 49 of 50 of confounders,

# not controlling sources of exogenous variation coef(lm.fit(y=y,x=X[,c(1:2, xc.cols[1:49])]))[2]

## d

## 1.028753

# Controlling for 49 of 50 of confounders

# and controlling for all sources of exogenous variation coef(lm.fit(y=y,x=X[,c(1:2, xc.cols[1:49], xe.cols)]))[2]

## d

## 1.478361

If we don’t add any observable source of exogenous variation, we find an estimate pretty close to the true causal effect $\alpha=1$, but if we control for the exogenous variation our estimate looks substantially biased.

OK, one probably should not infer a bias from a single simulation run. Therefore, I have run a proper Monte- Carlo simulation, where I repeated the data generation and both estimations a 1000 times.

sim = readRDS("control\_exo\_sim.Rds") library(ggplot2)

ggplot(sim, aes(x=alpha.hat, fill=reg)) + geom\_density() + facet\_wrap(~reg) + geom\_vline(xintercept=1, alpha=0.7)



sim %>% group\_by(reg) %>% summarize(

bias = mean(alpha.hat-alpha), se = sd(alpha.hat)

)

## # A tibble: 2 x 3

## reg bias se

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ## |  | | | |
| ## | 1 | control\_xe | 0.507 | 0.205 |
| ## | 2 | dont\_control\_xe | 0.0250 | 0.0437 |

We now properly see that we have a substantial bias in our causal effect estimate $\hat \alpha$ if we control for 49 out of 50 confounders and also add all sources of exogenous variation xe as control variables. The standard error is also high. If we don’t control for the sources of exogenous variation both the bias and standard error of $\hat \alpha$ are very small instead.

The following graph thus summarizes the goal in selecting control variables:



Unless we can manually guarantee that enough relevant sources of exogenous variation are excluded from the pool of candidate control variables, there seems to be a drawback of the post-double selection procedure: The lasso regression of $d$ on all potential control variables is likely to select the sources of exogenous variation, which we don’t want to use as control variables.

In contrast, if we would just select variables with a single lasso regression of $y$ on $d$ and all potential control variables, we seem much less likely to select sources of exogenous variation, since by definition they only affect $y$ via $d$. Let’s run both approaches on our simulated data set:

# Post double selection

double\_sel = rlassoEffect(x=X[,-2],y=y,d=d, method="double selection") coef(double\_sel)

## d1

## 1.440466

# Simple gamlr post-lasso estimator simple\_lasso = gamlr(x=X,y=y)

post\_lasso\_coef(simple\_lasso,x=X,y=y,keep.intercept = FALSE)[1]

## d

## 1.023805

Indeed, the simple lasso approach yields here a much better estimate of the causal effect than double selection. The hdm package also allows to compute confidence intervals:

confint(double\_sel,level = 0.999)

## 0.05 % 99.95 %

## d1 1.334566 1.546366

The computed 99.9% confidence interval of the post double selection estimator is erroneously far away from the true causal effect $\alpha=1$.

The following code estimates both models again and also returns statistics about the selected variables:

set.seed(1) lasso\_sim(alpha=1,n=700,Kc=50,Ke=50,Ky=50,Ku=700,

models=c("gamlr\_simple","rlasso\_double\_sel"))

## model coef num.vars xc xe xu xy ## 1 gamlr\_simple 1.023805 179 50 28 51 50

## 2 rlasso\_double\_sel 1.440466 7 5 2 0 0

We see here that the reason why the post double selection algorithm does not work in our example is not that too many xe are selected, but that overall only 7 variables are selected including just 5 of the 50 confounders. The default settings of rlasso seem to select relatively few variables.

Below we estimate two variants of the post double selection where we reduce a parameter c in the determination of the penalty term. This causes rlasso to select more variables:

set.seed(1) model = list(

rlasso\_double\_sel\_c106 = list(lasso.fun="rlasso",type="double\_sel", args=list(penalty=list(c=1.06))),

rlasso\_double\_sel\_c100 = list(lasso.fun="rlasso",type="double\_sel", args=list(penalty=list(c=1)))

)

lasso\_sim(alpha=1,n=700,Kc=50,Ke=50,Ky=50,Ku=700,return.what = "details", models = models)

## model coef num.vars xc xe xu xy ## 1 rlasso\_double\_sel\_c106 1.489470 76 32 34 8 2

## 2 rlasso\_double\_sel\_c100 1.214481 140 50 50 34 6

Now more variables are selected, but never more confounders xc than sources or exogenous variation xe. Let’s load the results of a proper Monte-Carlo simulation with 1000 simulation runs:

sim = readRDS("lasso\_sel\_sim.Rds") sim %>%

group\_by(model) %>% summarize(

bias = mean(coef-1), se = sd(coef)

)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ##  ## | # | A tibble: 4 x  model | 3 | bias | se |
| ## |  |  | |  |  |
| ## | 1 | gamlr\_simple | | 0.0242 | 0.00951 |
| ## | 2 | rlasso\_double\_sel | | 0.431 | 0.101 |
| ## | 3 | rlasso\_double\_sel\_c100 | | 0.176 | 0.292 |
| ## | 4 | rlasso\_double\_sel\_c106 | | 0.365 | 0.192 |

We see how the simple lasso model yields the lowest bias and also has a lower standard error than the 3 specifications of the post double selection approach.

# Are confounders that strongly affect y worse than those that strongly affect d?

Finally, let us look at a simulation where both the xe and the xc variables are confounders. Yet, there is a difference. The confounders xe strongly affect the explanatory variable d with coefficients $\beta\_k^{ed}=10$ and only weakly affect y with $\beta\_k^{ey}=0.5$. The coefficients shall be reversed for the confounders xc that strongly affect y with $\beta\_k^{cy}=10$ but only weakly affect d with $\beta\_k^{cd}=0.5$.

Let’s compare post double selection with a simple gamlr lasso selection in this setting:

set.seed(1)

res = lasso\_sim(alpha=1,n=100,Kc=10,Ke=10,Ky=5,Ku=20, beta.ed = 10, beta.ey = 0.5, beta.cd = 0.5, beta.cy = 10, models=c("gamlr\_simple","rlasso\_double\_sel"))

select(res, - coef)

## model num.vars xc xe xu xy ## 1 gamlr\_simple 16 10 1 0 5

## 2 rlasso\_double\_sel 17 7 10 0 0

I have not yet shown the estimated coefficients. But we see that the simple lasso regression has selected all confounders that strongly affect y but only 1 confounder that strongly affects d. In contrast, double selection picks 7 out of the 10 confounders that strongly affect y and all 10 that strongly affect d.

Why not make a guess about the two estimates $\hat \alpha$ of the two methods… Loading…

Ok, let’s look at the complete result:

res

## model coef num.vars xc xe xu xy ## 1 gamlr\_simple 1.051267 16 10 1 0 5

## 2 rlasso\_double\_sel 8.150202 17 7 10 0 0

While the simple lasso estimate is pretty close to the true causal effect, the double selection estimate is pretty far away, even though it selects many more confounders. It seems as if omitting confounders that strongly affect the dependent variable is worse than omitting those that strongly affect the explanatory variable.

If you find this result surprising, it may be because the simple omitted variable bias formula is often presented in a way that could wrongly suggest that an omitted confounder that strongly affects the explanatory variable d is equally bad as one that strongly affects the dependent variable y. However, even with a single omitted variable it is worse if it strongly affects y rather than d (see my previous blog post for more details).

A thousand simulation runs also verify that the single run above is representative:

sim = readRDS("lasso\_sim3.Rds") sim %>%

group\_by(model) %>% summarize(

bias = mean(coef-1), se = sd(coef),

num.vars = mean(num.vars), xe = mean(xe),

xc = mean(xc)

)

## # A tibble: 2 x 6

## model bias se num.vars xe xc ##

## 1 gamlr\_simple 0.0489 0.00393 16.6 0.685 10

## 2 rlasso\_double\_sel 6.85 4.04 17.8 10 6.65

To sum up, this simulation provides another related example, in which control variable selection with a simple lasso regression works better than the post double selection method.

# Conclusion

We first have illustrated a setting in which post double selection performed considerably better in selecting

control variables than just a simple single lasso regression. Post double selection is probably most save to apply in settings where an exogenous source of variation is known and manually taken care that it is excluded from the set of potential control variables. Intuitively, the procedure seems also reasonable in quasi-experimental settings where d is a dummy variable of a single policy change. Belloni et. al. (2014) indeed mostly refer to such settings.

Yet, we also have illustrated that in situations in which many important sources of exogenous variation are part of the set of potential control variables, a simple lasso regression can outperform post double selection. We also exemplified in a setting without observable sources of exogenous variation that a simple lasso regression was better able to select the worse confounders (i.e. those that affect y strongly).

Perhaps a method will surface that works well across all situations were either double selection or selection with a simple lasso fails. Alas, I imagine that the selection of control variables can be automatized only up to a certain extend and will continue to require considerable subjective assessments. For example, in addition to the issues discussed in this post, one must also ensure that the set of potential control variables does not contain colliders or channel variables (unless one wants to explicitly ignore the mediated effect from certain channels).