Preliminaries: data generating process

The first part of these section notes is a variation on Max Auffhammer's final exam for ARE212 in 2011. If you don't like this treatment of IV estimation, then blame him. Actually, I like this pattern: if you don't like something I do, then blame Max. He is probably at fault. I am faultless. Consider the following data generating process:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \eta_i \quad \text{with} \quad \eta_i \sim N(0,1) \tag{1}$$

and $\beta = [1\ 2\ \text{-}4\ 1]'$. Assume that the covariance matrix of the covariates, an additional instrument, and the idiosyncratic error (x_{1i}, x_{2i}, x_{3i}, z_i, and η_i) is defined to be

$$\Sigma = \begin{bmatrix} 1 & 0 & \rho_{13} & 0 & 0 \\ 0 & 1 & \rho_{23} & \rho_{2z} & 0 \\ \rho_{13} & \rho_{23} & 1 & 0 & 0 \\ 0 & \rho_{2z} & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where each variable is assumed to have zero mean for simplicity. Note that Σ is consistent with η_i being independently and identically distributed with constant variance. We will explore the properties of IV (e.g., weak instruments and the exclusion restriction) via Monte Carlo simulation.

The first step, then, is to figure out how to generate random data with the appropriate covariance. For this, a useful function is `rmvn.chol`, which returns a random $n \times k$ multivariate normal matrix $\mathbf{X}$, based on the supplied covariance matrix `vcov.mat`. In general, $\mathbf{X} = \mathbf{Q} + \mu'$, where $\mathbf{Q}'\mathbf{Q} = \Sigma$ and μ is a vector of means for each of the k columns of $\mathbf{X}$. We have assumed away $\mu = \mathbf{0}$ and so the problem becomes a simple application of matrix decomposition.

```
rmvn.chol <- function(n, vcov.mat) {
  k <- ncol(vcov.mat)
  Q <- chol(vcov.mat)
  Z <- matrix(rnorm(k*n), nrow=n, ncol=k)
  return(Z %*% Q)
}
```

It will be handy to have a simple function to generate Σ with three arguments representing the three non-zero correlations across the covariates and the instrument.

```
vcov.fn <- function(rho.13, rho.23, rho.2z) {
  mat <- diag(5)
  mat[3,1] <- rho.13; mat[1,3] <- rho.13
  mat[2,3] <- rho.23; mat[3,2] <- rho.23
  mat[2,4] <- rho.2z; mat[4,2] <- rho.2z
  return(mat)
}
```

The result is a way to generate the random data according to the specified data generating process. The following generates the covariance matrix and a random multivariate normal matrix with 500 observations, printing Σ for reference:

```
(vcov <- vcov.fn(0, 0.5, 0.5))
X <- rmvn.chol(500, vcov)
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1  0.0  0.0  0.0    0
[2,]    0  1.0  0.5  0.5    0
[3,]    0  0.5  1.0  0.0    0
[4,]    0  0.5  0.0  1.0    0
[5,]    0  0.0  0.0  0.0    1
```

A quick check to ensure that the variance of each variable is 1, as specified by Σ. There will be some variation around the true variance, but even with 500 observations, it's clear that we are approaching the true variance:

```
apply(X, 2, function(i){var(i)})
```

```
[1] 0.9318116 0.9494840 1.1271233 0.9727111 1.0298838
```

Calculate bias in β and it's standard error

We will now write a couple of functions to help examine the bias of the parameter vector and it's standard error, using an array of estimation techniques, including OLS, 2SLS, and estimation by proxy variable. First, let's put together a very simple function to calculate the parameter vector and it's standard error for both direct regression and two-stage least squares, when a first-stage covariate matrix is provided.

```
ols.results <- function(y, X, first=FALSE) {
  Xt <- t(X)
  xtxi <- solve(Xt %*% X)
  beta <- xtxi %*% Xt %*% y
  if (first == FALSE) {
    e <- y - X %*% beta
  }
  else {
    e <- y - first %*% beta
  }
  s2 <- (t(e) %*% e)/(nrow(X) - ncol(X))
  se <- sqrt(diag(xtxi)*s2)
  return(cbind(beta, se))
}
```

Now comes the serious stuff, specifically, the code that is written specifically to examine IV estimation in this example. Suppose that we do not observe x_{3i}. The composite error is then $x_{3i} + \eta_i$, and we estimate the parameter vector by regressing y_i on x_{1i} and x_{2i}. If $\rho_{13} = \rho_{23} = 0$, then there is no problem: OLS will yield consistent estimates, since the regression utilizes only exogenous variation. If, however, the covariates are correlated with the composite error, the OLS estimates will be biased.

The function `est.bias` returns the simulated bias in the parameter estimates and standard errors from a Monte Carlo simulation with $B = 10,000$ repetitions. The arguments are `vcov` which is the variance-covariance matrix generated by `vcov.fn`; `n` which specifies the number of observations for each iteration, defaulted at 500; `B` is the number of iterations in the MC simulation, defaulted at 10,000; `two.stage` is a boolean argument indicating whether the simulation should employ two-stage least squares with z_i as the instrument for x_{3i}, defaulted to `FALSE`; `proxy` is a boolean indicating whether z_i should be treated as a proxy for x_{3i}, defaulted to `FALSE`. The default behavior, then, is to run a simulation where x_{3i} is left out of the OLS regression, relegated to the error term.

```
est.bias <- function(vcov, n=500, B=1000, two.stage=FALSE, proxy=FALSE) {
  true.beta <- c(1, 2, -4, 1)
```

```
  res.beta <- mat.or.vec(3,B); res.se <- mat.or.vec(3,B)

  for (i in 1:B) {

    data <- rmvn.chol(n, vcov)

    X <- cbind(1, data[,1:3]); eta <- data[,5]
    y <- X %*% true.beta + eta
    full.ols <- ols.results(y, X)

    if (two.stage==TRUE) {
      endog <- data[,2]
      first  <- cbind(1, data[,c(1,4)])
      predict <- first %*% solve(t(first) %*% first) %*% t(first) %*% endog
      exog <- cbind(1, data[,1], predict)
      limited.ols <- ols.results(y, exog, first=first)
    }
    else if (proxy==TRUE) {
      exog <- cbind(1, data[,c(1, 4)])
      limited.ols <- ols.results(y, exog)
    }
    else {
      exog <- cbind(1, data[,1:2])
      limited.ols <- ols.results(y, exog)
    }

    res.beta[,i] <- limited.ols[,1] - true.beta[1:3]
    res.se[,i]   <- limited.ols[,2] - full.ols[1:3,2]
  }
  results <- cbind(rowMeans(res.beta), rowMeans(res.se))
  colnames(results) <- c("beta bias", "se bias")
  print(results)
}
```

We can check `est.bias` by first setting $\rho_{13} = \rho_{23} = 0$ and ensuring that the bias is very low with $n = 500$ and $B = 10,000$. The following MC simulation sets $\Sigma = \mathbb{I}_5$ and runs the following regression $10,000$ times: $y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon_i$, where $\epsilon_i = x_{3i} + \eta_i$.

```
vcov <- vcov.fn(0, 0, 0)
est.bias(vcov)
```

```
         beta bias    se bias
[1,]  3.025788e-05 0.01850461
[2,]  7.504312e-05 0.01853065
[3,] -2.155886e-04 0.01853233
```

```
vcov <- vcov.fn(0, 0.5, 0.5)
est.bias(vcov)
```

```
         beta bias     se bias
[1,] -9.313589e-05 0.014414161
[2,] -3.125962e-04 0.014437845
[3,]  5.000045e-01 0.007484851
```

```
est.bias(vcov, two.stage=TRUE)
```

```
        beta bias   se bias
[1,]  0.0006683202 0.1239639
[2,] -0.0005169007 0.1240761
[3,] -0.0042907956 0.2877558
```

We can prove that, in general, the variance of the IV estimator is larger than the variance of an OLS estimator. Let be a matrix of instruments and $\mathbf{P} =$. Then the IV estimator is $\hat{\beta}_{iv} = (\mathbf{X}'\mathbf{P}\mathbf{X})^{-1}\mathbf{X}'\mathbf{P}\mathbf{y}$. The projection matrix $\mathbf{P}$ is symmetric and idempotent, as is the residual maker $(\mathbb{I}_N - \mathbf{P})$. Note that for any $\mathbf{X}$ and any symmetric, idempotent matrix, the following fact is true:

$$\mathbf{X}'\mathbf{P}\mathbf{X} = \mathbf{X}'\mathbf{P}\mathbf{P}\mathbf{X} = \mathbf{X}'\mathbf{P}\mathbf{X} = (\mathbf{P}\mathbf{X})'(\mathbf{P}\mathbf{X}) \geq 0 \tag{2}$$

Note also that $\mathbb{V}(\hat{\beta}_{ols}) = (\mathbf{X}'\mathbf{X})^{-1}$ and $\mathbb{V}(\hat{\beta}_{iv}) = (\mathbf{X}'\mathbf{P}\mathbf{X})^{-1}$. It suffices to show that $\mathbf{X}'\mathbf{X} \geq \mathbf{X}'\mathbf{P}\mathbf{X}$ to complete the proof. It follows from Equation (2) that:

$$\mathbf{X}'(\mathbb{I}_N - \mathbf{P})\mathbf{X} \geq 0 \Rightarrow \mathbf{X}'\mathbf{X} - \mathbf{X}'\mathbf{P}\mathbf{X} \geq 0 \Rightarrow \mathbf{X}'\mathbf{X} \geq \mathbf{X}'\mathbf{P}\mathbf{X},$$

thus proving that $\mathbb{V}(\hat{\beta}_{iv}) \geq \mathbb{V}(\hat{\beta}_{ols})$. The intuition is a bit more clear. The IV estimator by design restricts the use of covariation between $\mathbf{y}$ and $\mathbf{X}$ to *only* exogenous variation and therefore greater variance of the estimator.

```
vcov <- vcov.fn(0, 0.5, 0.5)
est.bias(vcov, proxy=TRUE)
```

```
        beta bias    se bias
[1,] -0.0000945208 0.09679044
[2,]  0.0002856399 0.09692606
[3,]  2.0001688753 0.08998506
```

```
vcov <- vcov.fn(0.5, 0.5, 0.5)
est.bias(vcov)
```

```
        beta bias       se bias
[1,] -0.0006474492  9.996362e-03
[2,]  0.5005109401 -1.006042e-04
[3,]  0.5004461721 -8.279249e-05
```

```
est.bias(vcov, two.stage=TRUE)
```

```
        beta bias   se bias
[1,]  0.0005228898 0.1224915
[2,]  0.4996544539 0.1125726
[3,] -0.0030798629 0.2818936
```

```
vcov <- vcov.fn(0, 0.5, 0.01)
est.bias(vcov)
```

```
        beta bias     se bias
[1,] -5.845659e-06 0.014421682
[2,]  9.731249e-05 0.014443940
[3,]  5.000803e-01 0.007497558
```

```
est.bias(vcov, two.stage=TRUE)
```

```
       beta bias    se bias
[1,]  0.18013841   982.6684
[2,] -0.02440537   745.3372
[3,]  2.06182132 13433.0801
```