In this section we'll use simulated data to study the linked concepts of omitted variables bias and instrumental variables. But first, we'll go over a few notes from the last problem set.

Problem set 3 retrospective

In general, the class did well on the last problem set. There were errors here and there but most had to do with simple coding issues, not errors in comprehension of the material. If you find that your answers don't match those on the answer key, consider checking your final results with a friend, ideally one you didn't work with — the probability of you both making identical mistakes is much smaller than the probability of you making a mistake on your own!

Summarizing data with graphs

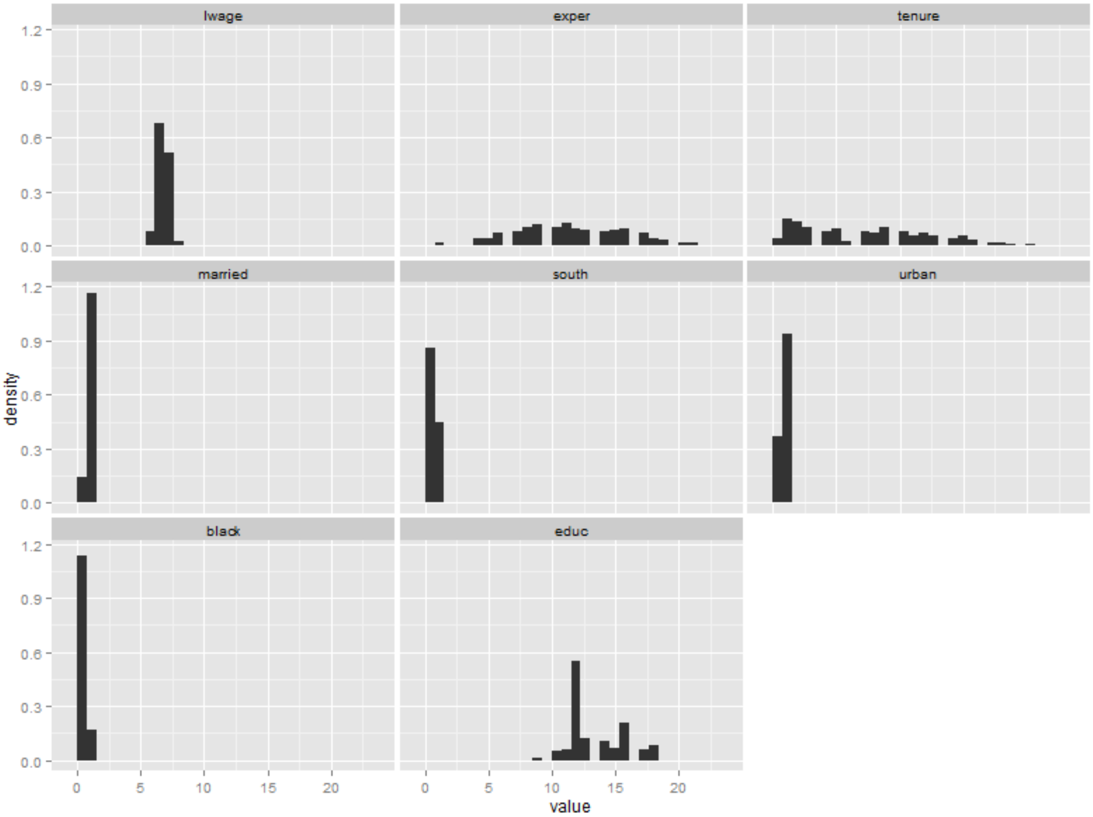
There are many, many ways to view and slice your data. In general, the more the better. Of course, I don't want to see all of them on your problem set, but in true research you want to have a full quiver of different ways to graph your dataset. Here are two small coding examples that will make graphing your data easier. First, we'll "melt" our data so that it is in long, rather than wide format.

```
library(reshape)
data.df <- read.table("nls80.raw")
names(data.df) <- c("wage","hours","iq","kww","educ","exper",
  "tenure","age","married","black","south","urban","sibs",
  "brthord","meduc","feduc","lwage")
data.df <- data.df[c("lwage","exper","tenure","married",
  "south","urban","black","educ")]
histdraw.df <- melt(data.df, measure.vars = c("lwage","exper","tenure","married",
  "south","urban","black","educ"))
histdraw.df[c(1:2,1000:1001,3000:3001), ]
```

```
     variable     value
1       lwage  6.645091
2       lwage  6.694562
1000    exper 15.000000
1001    exper 10.000000
3000  married  1.000000
3001  married  1.000000
```

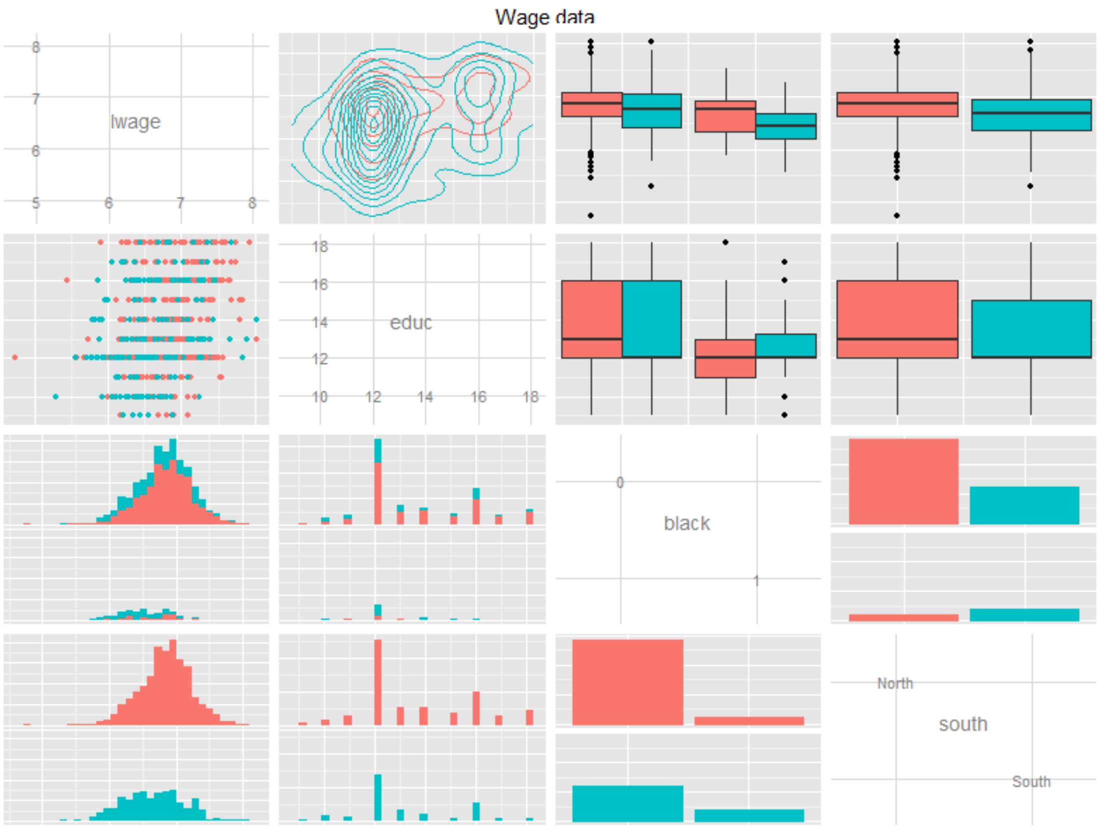
Normally, we would need to tell `melt` which of our variables are "id" variables and which are "measured" variables. Here we call all of our variables measured, giving us the preferred format for running `ggplot()`.

```
library(ggplot2)
ggplot(data = histdraw.df, aes(x = value)) +
  geom_histogram(aes(y = ..density..), binwidth = diff(range(histdraw.df$value))/30) +
  facet_wrap(˜ variable)
```



You are all familiar with ggpairs(), but you may not know that it is capable of far more than just simple scatterplots. In fact, ggpairs() is pretty smart — if you pass it factor variables, for example, it will treat them as discrete rather than continuous and graph appropriately. You can also easily customize what appears in the upper and lower sections of the graph.

```
library(GGally)
plots.df <- data.df
plots.df$married <- as.factor(plots.df$married)
plots.df$south <- as.factor(plots.df$south)
plots.df$urban <- as.factor(plots.df$urban)
plots.df$black <- as.factor(plots.df$black)
levels(plots.df$south) <- c("North", "South")
ggpairs(data = plots.df,
  columns = c("lwage","educ","black", "south"),
  upper = list(continuous = "density", combo = "box"),
  colour = "south",
  title = "Wage data"
)
```



Creating interaction variables

Many of you went to heroic lengths to create interaction terms by hand. Here is an easier way:

```
interactions <- t(apply(data.df[ ,2:8], 1, combn, 2, prod))
colnames(interactions) <- paste(combn(c("exper","tenure","married","south","urban","black","educ"), 2, p
print(interactions[1:6,3:7])
```

```
     experVsouth experVurban experVblack experVeduc tenureVmarried
[1,]           0          11           0        132              2
[2,]           0          11           0        198             16
[3,]           0          11           0        154              9
[4,]           0          13           0        156              7
[5,]           0          14           0        154              5
[6,]           0          14          14        224              2
```

Now, onto the good stuff!

Simulating omitted variables bias

This part of these section notes is a variation on Max Auffhammer's final exam for ARE212 in 2011. In short, our goal is to generate a framework where we can simulate bias, and then examine possible solutions to that bias.

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 $\beta = [1\ 2\ \text{-}4\ 3]'$. 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, we'll write the 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`. We'll need to use the fact that premultiplying an $n \times k$ matrix filled with draws from a standard normal distribution with the Cholesky decomposition of Σ (i.e. $\mathbf{Q}$, where $\mathbf{Q}'\mathbf{Q} = \Sigma$) will give us the matrix we want[1].

```
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 also 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:

```
set.seed(42)
(vcov <- vcov.fn(rho.13 = 0, rho.23 = 0.5, rho.2z = 0.5))
X <- rmvn.chol(500, vcov)
```

[1] I'm not going to prove this, but I'm more than willing to wave my hands at it. To start, note that the Cholesky decomposition is the linear algebra equivalent of taking a square root.

```
     [,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
```

Now we also want to see that the estimated variance covariance matrix of `X` does, in fact, mirror the Σ matrix we specified:

```
(vcov.data <- var(X))
```

```
             [,1]         [,2]          [,3]          [,4]        [,5]
[1,]  0.945130845 -0.008457707  0.0285443272 -0.0151609602 -0.03685374
[2,] -0.008457707  1.066945223  0.5340984979  0.5182452952 -0.01498788
[3,]  0.028544327  0.534098498  0.9684988460  0.0006668004 -0.07744696
[4,] -0.015160960  0.518245295  0.0006668004  1.0188597996  0.01454024
[5,] -0.036853744 -0.014987876 -0.0774469608  0.0145402406  0.99199352
```

Estimate and fix bias with instrumental variables

We will now write a couple of functions to help examine the bias of the parameter vector and its 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 its 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) {
  XpXinv <- solve(t(X) %*% X)
  b <- XpXinv %*% t(X) %*% y

  if (is.matrix(first)) {
    e <- y - first %*% b
  } else {
    e <- y - X %*% b
  }

  s2 <- (t(e) %*% e) / (nrow(X) - ncol(X))
  se <- sqrt(diag(XpXinv) * s2)
  return(list(b = b, se = se))
}
```

We'll use the `first` flag later, to calculate the correct residuals for IV/2SLS[2].

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 the omitted variable is uncorrelated with the covariaties, i.e. $\rho_{13} = \rho_{23} = 0$, then there is no problem: OLS will yield consistent estimates, since the regression utilizes only exogenous variation. If, however, the covariates

[2] See Max's notes for a refresher on this. In short — we don't want to use the predicted values of the endogenous variable to calculate the residuals; failing to correct here would give us the wrong standard errors.

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`. 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 = 10000, two.stage = FALSE) {
  true.beta <- c(1, 2, -4, 3)
  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 {
      exog <- cbind(1, data[,1:2])
      limited.ols <- ols.results(y, exog)
    }

    res.beta[ , i] <- limited.ols$b - true.beta[1:3]
    res.se[ , i]   <- limited.ols$se - full.ols$se[1:3]
  }

  results <- cbind(rowMeans(res.beta), rowMeans(res.se))
  colnames(results) <- c("beta bias", "se chg")
  print(results)
}
```

A couple notes on this code. First, `full.ols` is our model *if we were able to observe* x_{3i}. It is our basis of comparison for this exercise, but wouldn't be estimatable under normal conditions. Second, let's take a slightly closer look at the `if-else` statement in the middle of this code. If the flag `two.stage` has been set to `TRUE`, then `est.bias` performs 2SLS. First, it regresses the endogenous variable (x_{2i}) on the exogenous variables (the covariate x_{1i} and the instrument z_i). Then, it regresses y_i on the exogenous variable x_{2i} and the (also exogenous) predicted values for x_{2i}. That's it!

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 $\Sigma = \mathbf{I}_5$ and runs the following regression $10,000$ times:

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \varepsilon_i \quad \text{with} \quad \varepsilon_i = x_{3i} + \eta_i \tag{2}$$

```
vcov <- vcov.fn(rho.13 = 0, rho.23 = 0, rho.2z = 0)
est.bias(vcov)

        beta bias     se chg
[1,] 0.0012811636 0.09687554
[2,] 0.0009925163 0.09703724
[3,] 0.0001443709 0.09698733
```

The parameter estimates seem fairly consistent. The simulated bias is negligible. The standard errors in the simulation are somewhat larger, however, simply because there is more variation in the composite error term. Now suppose that $\rho_{23} = \rho_{2z} = 0.5$, so that $\mathbb{E}[\boldsymbol{\varepsilon}_i | x_{1i}, x_{2i}] \neq 0$. The coefficient β_2 is biased upwards by 1.5 in the simulation below.

```
vcov <- vcov.fn(rho.13 = 0, rho.23 = 0.5, rho.2z = 0.5)
est.bias(vcov)

         beta bias     se chg
[1,]  0.0029008827 0.07992574
[2,] -0.0008977518 0.08003102
[3,]  1.5014989328 0.07305445
```

Why 1.5? Remember equation 6.5 in Max's notes: the bias is equal to the product correlation between x_{2i} and x_{3i} (0.5) and the true effect of x_{3i} on y_i (3). Since we don't actually observe x_{3i}, neither of these are observable, but if we could make a convincing argument that they were both positive we could at least suggest that the sign of our bias is positive[3].

That's a lot of bias. Fortunately, we have a suitable instrument, by design, that we can use to cordon off exogenous variation for estimation. Note that $Cov(z_i, \boldsymbol{\varepsilon}_i) = 0$ since z_i is uncorrelated with the full, composite error, but that $Cov(z_i, x_{2i}) = 0.5 \neq 0$, satisfying both properties of a suitable instrument. We can now get a consistent estimate for β:

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

Note, however, that the standard errors are much higher. We can prove that, in general, the variance of the IV estimator is larger than the variance of an OLS estimator. Let $\mathbf{Z}$ be a matrix of instruments and $\mathbf{P} = \mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'$. 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 $\mathbf{M} = (\mathbf{I}_n - \mathbf{P})$. Note that for any $\mathbf{X}$ and any symmetric, idempotent matrix $\mathbf{W}$, we have the following use linear algebra fact: [4]:

$$\mathbf{X}'\mathbf{W}\mathbf{X} = \mathbf{X}'\mathbf{W}\mathbf{W}\mathbf{X} = \mathbf{X}'\mathbf{W}'\mathbf{W}\mathbf{X} = (\mathbf{W}\mathbf{X})'(\mathbf{W}\mathbf{X}) \geq \mathbf{0} \tag{3}$$

Note also that $\mathbb{V}(\hat{\beta}_{ols}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$ and $\mathbb{V}(\hat{\beta}_{iv}) = \sigma^2(\mathbf{X}'\mathbf{P}\mathbf{X})^{-1}$. It suffices to show that $\mathbf{X}'\mathbf{X} \geq \mathbf{X}'\mathbf{P}\mathbf{X}$ to complete the proof. It follows from Equation (3) that:

$$\mathbf{X}'\mathbf{M}\mathbf{X} = \mathbf{X}'(\mathbf{I}_n - \mathbf{P})\mathbf{X} \geq 0 \Rightarrow \mathbf{X}'\mathbf{X} - \mathbf{X}'\mathbf{P}\mathbf{X} \geq 0 \Rightarrow \mathbf{X}'\mathbf{X} \geq \mathbf{X}'\mathbf{P}\mathbf{X},$$

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

[3] Which, depending on our research setting, may or may not be helpful...

[4] This is because of another useful linear algebra fact: any idempotent and symmetric matrix is positive semi-definite.

Weak instruments in practice

Instrumental variables are by no means a cure-all, however — far from it. First of all, an argument must be made for the exclusion restriction, that indeed the instrument is uncorrelated with the composite error term. There is no way to check this empirically. Second, even if the exclusion restriction holds, the instrument must be highly correlated with the covariates. Otherwise, we run into the problem of weak instruments, which is a *big* problem, as illustrated below. The set up is exactly the same, except that $\rho_{2z} = 0.01$. Technically, both properties of a suitable instrument are satisfied, but the instrumental projection of x_{2i} leaves very little variation to work with. The result is a crazy amount of variance, yielding an almost certainly biased estimate of β. The first simulation shows the standard bias induced by a violation of strict exogeneity:

```
vcov <- vcov.fn(rho.13 = 0, rho.23 = 0.5, rho.2z = 0.01)
est.bias(vcov)
```

```
        beta bias     se chg
[1,] -0.002298768 0.07980712
[2,] -0.001249993 0.07991724
[3,]  1.501399486 0.07296658
```

The second simulation displays the results of trying to correct the endogeneity with a weak instrument. Not good. Way worse.

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

```
       beta bias     se chg
[1,]  0.05645526   997.0063
[2,] -0.01069822  1824.4216
[3,]  5.34220154 48177.2764
```