Bootstrapping and Instrumental Variables

Drew Dimmery drewd@nyu.edu

March 8, 2014

Structure

- Rubber-duck debugging
- Bootstrapping
  - I'll give some examples of the more complicated bootstrapping methods
- Instrumental Variables
  - Basic 2SLS
  - Sensitivity Analysis for IV

Debugging

- Rubber duck debugging:
  - Use when you can't figure out why your code doesn't work right
  - Find something inanimate to talk to
  - Explain what your code does, line by excruciating line
  - If you can't explain it, that's probably where the problem is.
  - This works ridiculously well.
  - You should also be able to tell your duck exactly what is stored in each variable at all times.
- Check individual elements of your code on small data such that you know what the right answer *should* be.

A note on the Bootstrap

- I'm going to talk about the non-parametric bootstrap (which Cyrus talked about)

- The key to this is that we don't want to make an assumption about what our data looks like.
- So what looks more like our data than anything else we know of?
- Our data.
- We sample repeatedly *from the empirical distribution of our data.*
- This is why we resample with replacement. Each draw is an independent draw from the empirical distribution of the data we've collected.

Advanced bootstrapping

- There are more advanced uses of the bootstrap in the construction of estimators. One example is in so called *bias reduction.* (digression ahead)
- If b is the bias of an estimator $\hat{\theta}$ of θ, we could obtain an unbiased estimator by subtracting b from $\hat{\theta}$.
- Bootstrapping let's us estimate b with $\frac{1}{B}\sum_{i=1}^{B}[\hat{\theta}_i^b - \hat{\theta}]$ where $\hat{\theta}_i^b$ is an estimate after resampling from the empirical distribution.
- Not guaranteed to do better, but it's an interesting approach.
- Talk to me for more info.

Bootstrap Example

- We're going to start with Lalonde again.
- We will do the IPW procedure as before, but this time get the "right" SEs.

. . .

```
require(MatchIt)
data(lalonde, package = "MatchIt")
p.model <- glm(treat ~ age + educ + black + hispan + married + nodegree + re74 +
    re75, lalonde, family = "binomial")
pscore <- predict(p.model, type = "response")
ipw <- lalonde$treat + (1 - lalonde$treat)/(1 - pscore)
ipw.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
    re74 + re75, lalonde, weights = ipw)
N <- nrow(lalonde)
bootstrapLalonde <- function(out = "coef") {
    b.samp <- sample(N, replace = TRUE)
    b.p.model <- glm(treat ~ age + educ + black + hispan + married + nodegree +
        re74 + re75, lalonde[b.samp, ], family = "binomial")
    b.pscore <- predict(b.p.model, type = "response")
    b.ipw <- lalonde$treat[b.samp] + (1 - lalonde$treat[b.samp])/(1 - b.pscore)
```

```
    b.ipw.mod <- lm(re78 ~ treat + age + educ + black + hispan + married + nodegree +
        re74 + re75, lalonde[b.samp, ], weights = b.ipw)
    if (out == "coef") {
        out <- coef(b.ipw.mod)[2]
    } else if (out == "t") {
        out <- summary(b.ipw.mod)$coefficients[2, 3]
    } else {
        stop("Unknown output statistic.")
    }
    out
}
b.samps.coef <- replicate(500, bootstrapLalonde())
b.samps.t <- replicate(500, bootstrapLalonde(out = "t"))
```

Check bootstrap output

- We should expect a larger SE since we're including additional first stage variation.

. . .

```
c(coef(ipw.mod)[2], mean(b.samps.coef))
```

```
## treat
##  1332  1389
```

```
c(summary(ipw.mod)$coefficients[2, 2], sd(b.samps.coef))
```

```
## [1] 697.9 767.1
```

```
c(summary(ipw.mod)$coefficients[2, 3], mean(b.samps.t))
```

```
## [1] 1.909 1.904
```

. . .

- How do we reconcile this? Is it a problem?

Cluster Bootstrapping

- When treatment was assigned with clustering, our bootstrap must account for that.
- So we adjust the resampling procedure.
- We now move away from Lalonde, and we'll look at a replication of (and extension to) the Green, Vavreck (2008) PA paper which examines the performance of cluster robust SEs in an experimental context.
- Treatment was a GOTV advertising campaign
- Outcome was 2004 turnout.
- 23869 individuals 19 or under in 85 clusters.

Read in G/V data

```
dat <- read.csv(file = "GreenVavreck_PolAnalysis_2008_PA_Replication.csv", head = TRUE,
    sep = ",")
f1 <- paste("tout1 ~ treat + ", paste(names(dat)[11:49], collapse = " + "))
ols1 <- lm(f1, dat)
summary(ols1)$coefficients[2, ]
```

```
##   Estimate Std. Error    t value   Pr(>|t|)
##  0.0241120  0.0070322  3.4288223  0.0006072
```

Try Cluster Robust SEs

- $\frac{H}{H-1}\frac{N-1}{N-K}(X'X)^{-1}\left(\sum_{h=1}^{H} X_h'\hat{e}_h\hat{e}_h'X_h\right)(X'X)^{-1}$
- The "estimating function" evaluates $(Y_i - X_i'\beta)X_i$ that is, $\hat{e}_iX_i$. It is $N \times p$.

. . .

```
robust.se <- function(model, cluster) {
    require(sandwich)
    require(lmtest)
    M <- length(unique(cluster))
    N <- length(cluster)
    K <- model$rank
    dfc <- (M/(M - 1)) * ((N - 1)/(N - K))
    uj <- apply(estfun(model), 2, function(x) tapply(x, cluster, sum))
    rcse.cov <- dfc * sandwich(model, meat = crossprod(uj)/N)
    rcse.se <- coeftest(model, rcse.cov)
```

```
    return(list(rcse.cov, rcse.se))
}
robust.se(ols1, dat$syscode)[[2]][2, ]

## Note: no visible binding for global variable 'lmtest'

##   Estimate Std. Error    t value   Pr(>|t|)
##    0.02411    0.01405    1.71577    0.08622

ses <- c(model = summary(ols1)$coefficients[2, 2], cl.robust = robust.se(ols1,
    dat$syscode)[[2]][2, 2])
```

Bootstrap SEs

- We should expect the cluster robust sandwich estimator to do pretty well here
- We have an actual experiment, and thus pretty plausible "true" clustering.

. . .

```
H <- length(unique(dat$syscode))
clus <- unique(dat$syscode)
obs.in.clus <- sapply(clus, function(h) which(dat$syscode == h))
clus <- as.character(clus)
names(obs.in.clus) <- clus
pairsBoot.GV <- function(out = "coef") {
    b.samp <- sample(clus, replace = TRUE)
    b.obs <- unlist(sapply(b.samp, function(h) obs.in.clus[[h]]))
    b.ols <- lm(f1, dat[b.obs, ])
    if (out == "coef") {
        out <- coef(b.ols)[2]
    } else if (out == "t") {
        out <- summary(b.ols)$coefficients[2, 3]
    } else {
        stop("Unknown output type.")
    }
    out
}
b.samps.coef <- replicate(500, pairsBoot.GV())
# b.samps.t <- replicate(500, pairsBoot.GV(coef='t'))
ses <- c(ses, pairs.boot = sd(b.samps.coef))
round(ses, 3)

##      model  cl.robust pairs.boot
##      0.007      0.014      0.024
```

Wild Cluster Bootstrap

- I'm not showing the residual cluster bootstrap
- Clusters are not of equal size in this data
- (And its probably rare that they will be)

. . .

```
ols.fit <- predict(ols1)
ols.res <- residuals(ols1)
dat$syscode <- as.character(dat$syscode)
f.b <- paste("newY ~ treat + ", paste(names(dat)[11:49], collapse = " + "))
wildBoot.GV <- function(out = "coef") {
    toflip <- rbinom(H, 1, 0.5)
    names(toflip) <- clus
    toflip <- names(toflip)[toflip == 1]
    b.resid <- ifelse(dat$syscode %in% toflip, -ols.res, ols.res)
    newY <- ols.fit + b.resid
    b.ols <- lm(f.b, cbind(newY, dat))
    if (out == "coef") {
        out <- coef(b.ols)[2]
    } else if (out == "t") {
        out <- summary(b.ols)$coefficients[2, 3]
    } else {
        stop("Unknown output type.")
    }
    out
}
b.samps.coef <- replicate(500, wildBoot.GV())
ses <- c(ses, wild.boot = sd(b.samps.coef))
round(ses, 3)
```

```
##      model  cl.robust pairs.boot  wild.boot
##      0.007      0.014      0.024      0.015
```

Instrumental Variables

- $\rho = \frac{\text{Cov}(Y_i, Z_i)}{\text{Cov}(S_i, Z_i)} = \frac{\frac{\text{Cov}(Y_i, Z_i)}{\text{Var}(Z_i)}}{\frac{\text{Cov}(S_i, Z_i)}{\text{Var}(Z_i)}} = \frac{\text{Reduced form}}{\text{First stage}}$
- If we have a perfect instrument, this will be unbiased.
- But bias is a function of both violation of exclusion restriction and of strength of first stage.
- 2SLS has finite sample bias. (Cyrus showed this, but didn't dwell on it)

- In particular, it can be shown that this bias "is":
$\frac{\sigma_{\eta\xi}}{\sigma_{\xi}^{2}} \frac{1}{F+1}$
where η is the error in the structural model and ξ is the error in the first stage.
- With an irrelevant instrument ($F = 0$), the bias is equal to that of OLS (regression of Y on X).
- There are some bias corrections for this, we might talk about this next week.

Setup IV example

- For our example with IV, we will look at AJR (2001) - Colonial Origins of Comparative Development
- Treatment is average protection from expropriation
- Exogenous covariates are dummies for British/French colonial presence
- Instrument is settler mortality
- Outcome is log(GDP) in 1995

. . .

```
require(foreign)
dat <- read.dta("maketable5.dta")
dat <- subset(dat, baseco == 1)
```

Estimate IV via 2SLS

```
require(AER)
first <- lm(avexpr ~ logem4 + f_brit + f_french, dat)
iv2sls <- ivreg(logpgp95 ~ avexpr + f_brit + f_french, ~logem4 + f_brit + f_french,
    dat)
require(car)
linearHypothesis(first, "logem4", test = "F")
```

```
## Linear hypothesis test
##
## Hypothesis:
## logem4 = 0
##
## Model 1: restricted model
## Model 2: avexpr ~ logem4 + f_brit + f_french
##
```

```
##   Res.Df RSS Df Sum of Sq    F  Pr(>F)
## 1     61 117
## 2     60  94  1        23 14.7 0.00031 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Examine Output

```r
summary(iv2sls)
```

```
##
## Call:
## ivreg(formula = logpgp95 ~ avexpr + f_brit + f_french | logem4 +
##     f_brit + f_french, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2716 -0.7488  0.0728  0.7544  2.4004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.372      1.388    0.99    0.327
## avexpr         1.078      0.218    4.95  6.3e-06 ***
## f_brit        -0.778      0.354   -2.20    0.032 *
## f_french      -0.117      0.355   -0.33    0.743
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.04 on 60 degrees of freedom
## Multiple R-Squared: 0.0483,  Adjusted R-squared: 0.000748
## Wald test: 10.1 on 3 and 60 DF,  p-value: 1.82e-05
```

Sensitivity Analysis

- Conley, Hansen and Rossi (2012)
- Suppose that the exclusion restriction does NOT hold, and there exists a direct effect from the instrument to the outcome.
- That is, the structural model is:
  $Y = X\beta + Z\gamma + \epsilon$
- If γ is zero, the exclusion restriction holds (we're in a structural framework)
- We can assume a particular value of γ, take $\tilde{Y} = Y - Z\gamma$ and estimate our model, gaining an estimate of β.

- This defines a sensitivity analysis on the exclusion restriction.
- Subject to an assumption about the support of γ, they suggest estimating in a grid over this domain, and then taking the union of the confidence intervals for each value of γ as the combined confidence interval (which will cover).

. . .

```
gamma <- seq(-1, 1, 0.25)
ExclSens <- function(g) {
    newY <- dat$logpgp95 - g * dat$logem4
    coef(ivreg(newY ~ avexpr + f_brit + f_french, ~logem4 + f_brit + f_french,
        cbind(dat, newY)))[2]
}
sens.coefs <- sapply(gamma, ExclSens)
names(sens.coefs) <- gamma
round(sens.coefs, 3)
```

```
##     -1  -0.75   -0.5  -0.25      0   0.25    0.5   0.75      1
## -0.793 -0.326  0.142  0.610  1.078  1.546  2.013  2.481  2.949
```