EC 711 Problem Set 1

Samuel Messer

February 2, 2024

Collaborated with: Erin Eidschun, Jimmy Jedras, Alex Kane, Ethan Lewis, Rachel Vogt, Christian Wilson

Question 1: Returns to Schooling (Angrist and Krueger, 1991)

1.

Following Angrist and Kreuger (1991) and Hansen, Hausman, and Newey (2008), we use as instruments indicators for quarter of birth (QOB 4 excluded), interacted with year of birth (YOB 39 excluded), and interacted with state (STATE 56 excluded). This leads to specifications with 3 instruments (S1), 30 instruments (S2), and 180 instruments (S3). Note that the results in S2 and S3 are sensitive to the choice of which group to exclude as we do not use all possible linear combinations. In particular, we never interact the excluded quarter of birth with any year of birth or state. It is unclear why this is the specification chosen in these papers, but in order to replicate the results as closely as possible we will follow that method.

Using 2SLS and LIML, we estimate the coefficient on education for each of the models. Results are presented in Table 1.

Model	S1	S2	S3
$\hat{\beta}_{2SLS}$	0.093 (0.020)	0.020 (0.006)	0.114 (0.002)
$\hat{\beta}_{LIML}$	0.094 (0.021)	0.018 (0.006)	0.154 (0.003)
Bekker (1994) Standard Errors	(0.021)	(0.007)	(0.004)

Table 1: Estimates and standard errors of the returns to schooling on wage. The standard errors below each estimate are calculated using standard asymptotics, while the Bekker standard error is robust to many instruments.

2.

To calibrate the model, we consider the results from estimating LIML using the S1 set of instruments. The LIML estimates provide us with the following values for $\beta = (\beta_0, \beta_1, \beta_2')'$:

$$\beta_0 = 4.543; \quad \beta_1 = 0.094; \quad \beta_{2,1} = 0.239; \quad \beta_{2,2} = -0.219; \quad \beta_{2,3} = -0.161$$

Where $\beta_{2,1}$, $\beta 2, 2$, and $\beta_{2,3}$ are the coefficients on MARRIED, RACE, and SMSA, respectively. To calibrate the vector $\pi = (\pi_0, \pi_1', \pi_2')'$, we fit a linear regression of X on Z and W. This yields the following estimates for π:

$$\pi_0 = 13.067; \quad \pi_{1,1} = 0.146; \quad \pi_{1,2} = -1.702; \quad \pi_{1,3} = -1.163$$
$$\pi_{2,1} = -0.144; \quad \pi_{2,2} = -0.091; \quad \pi_{2,3} = -0.037$$

Where $\pi_{1,1}$, $\pi_{1,2}$, and $\pi_{1,3}$ are the coefficients on the indicators for quarter of birth 1, 2, and 3, respectively, and $\pi_{2,1}$, $\pi_{2,2}$, and $\pi_{2,3}$ are the coefficients on MARRIED, RACE, and SMSA, respectively. Finally, to calibrate the variance-covariance matrix of the errors, we subtract the fitted values in both equations to obtain estimates of the individual errors. From there, we calculate the variance of each and their covariance, yielding the following matrix:

$$\Sigma_{U,V} = \begin{bmatrix} 0.398 & -0.152 \\ -0.152 & 10.353 \end{bmatrix}$$

3.

We generate 1000 synthetic datasets using the model calibrated in part 2. From each dataset, we obtain the point estimates (both 2SLS and LIML) of the effect of education on wage, the standard errors for each estimate (using standard asymptotics), and the standard error calculated using Bekker's formula. Note that the Bekker standard errors had to be implemented by hand, as the default option in `ivmodel` that estimates using standard errors robust to weak and many instruments crashed the software used to simulate here.

4.

From the 1000 datasets generated in part 3, we measure the bias relative to the "true" value of β, the one we used to generate the data. In addition, we measure the mean squared error and the coverage of a 95% confidence interval, using both standard asymptotics and the standard errors of Bekker. Results are reported in table 2

Instruments		Bias / β_0	RMSE	95% CI Coverage
S1	2SLS	0.009	0.021	0.958
	LIML	0.013	0.021	0.956
	Bekker			0.958
S2	2SLS	−0.029	0.018	0.955
	LIML	0.013	0.024	0.913
	Bekker			0.952
S3	2SLS	−0.102	0.015	0.871
	LIML	0.017	0.041	0.739
	Bekker			0.953

Table 2: Bias relative to true value, Root Mean Squared Error, and 95% Confidence Interval coverage from 1000 simulations of the data generating process described in part 2.

This table highlights the usefulness of the Bekker standard errors in getting appropriate size in tests. For example, in the S3 model, the 2SLS CIs cover the true value of β only 87% of the time, and the LIML CIs cover the true value of β 74% of the time. The Bekker standard errors, however, retain the appropriate size even with this large number of instruments.

In addition, the inconsistency of the 2SLS estimator under many instrument asymptotics is demonstrated by the S3 model. We see a strong negative bias in the 2SLS estimator, but a much smaller one in the LIML estimator.

Somewhat puzzling are the results we find in terms of RMSE and coverage for the LIML estimator. The RMSE is increasing for the LIML estimator as the number of instruments grows, and the coverage is worse

than the 2SLS estimator in both the S2 and S3 models, which is not the pattern found in Hansen, Hausman, and Newey.

Question 2: Weak Instrument Distribution (Staiger and Stock, 1997)

1.

See R code at the end of this document for the data generation.

2.

To get the normal and weak asymptotic distributions, we use several approaches. Starting with the normal approximation, we consider two data driven approaches and an analytical one. The first data driven approach takes the variance of the errors of both equations in the 2SLS system (note these are the actual errors, not the sample estimates), their covariance, the variance of X_i, and the R^2 from the first stage from each iteration. We then consider the mean of each of these quantities, and use the asymptotic variance formula:

$$V = \frac{\text{Var}(\epsilon_i)}{\text{Var}(X_i)R^2_{X_i|Z_i}}$$

to calculate the variance for our normal distribution. We then draw from this distribution 200000 times to simulate this normal.

The second approach considers the standard errors estimated automatically when estimating the 2SLS model. These use the normal asymptotics, so we should get something close. However, because we have a weak instrument, some of these standard errors are huge. Taking the mean would give us a variance way too high, so we consider the median of these estimated standard errors.

Finally, the analytic approach foregoes the weak instrument asymptotics. Instead of the coefficient on Z_i being $\Delta/\sqrt{(n)}$, it is Δ. Then the true asymptotic variance can be calculated as:

$$V = \frac{\text{Var}(\epsilon_i)}{\text{Var}(X_i)R^2_{X_i|Z_i}} = \frac{1}{1.25(0.2)} = 4$$

Which follows from the following:

- $\beta = 0 \implies Y_i = \epsilon_i \implies \text{Var}(\epsilon_i) = \text{Var}(U_i) = 1.$
- $Z_i \perp V_i \implies \text{Var}(X_i) = \text{Var}(\Delta Z_i) + \text{Var}(V_i) = \Delta^2\text{Var}(Z_i) + \text{Var}(V_i) = 0.25 * 1 + 1 = 1.25.$
- $R^2_{X_i|Z_i} = \frac{\text{Variance Explained by Z}}{\text{Total Variance of X}} = \frac{0.25}{1.25} = 0.2.$

These 3 different approaches give similar patterns, but require a scaling factor for the mean approach that we have been unable to rationalize. Thus, we will proceed with the analytic approach.

For the weak instrument asymptotics approximation, we consider a data driven approach and an analytical one. Here, we look first at the analytical approach. Consider the asymptotic distribution of the difference between $\hat{\beta}$ and β:

$$\hat{\beta} - \beta_0 \underset{d}{\to} \left[(Q_Z\Delta + W_V)' Q_Z^{-1} (Q_Z\Delta + W_V)\right]^{-1} (Q_Z\Delta + W_V)' Q_Z^{-1} W_U$$

Note that $\frac{Z'Z}{n} \underset{p}{\to} Q_Z$. In fact, this is a scalar and is equal to 1 (as it can be expressed as the expectation of $Z'Z$, which is equivalent to the variance of Z as Z is mean 0). Likewise, Δ is a scalar, and W_U and W_V are multivariate normal distributions, with the specified variance covariance matrix. Then in order to draw from this distribution, we can treat every object as a scalar. Thus, we can get a draw from this distribution by drawing U and V from the multivariate normal, combining them using

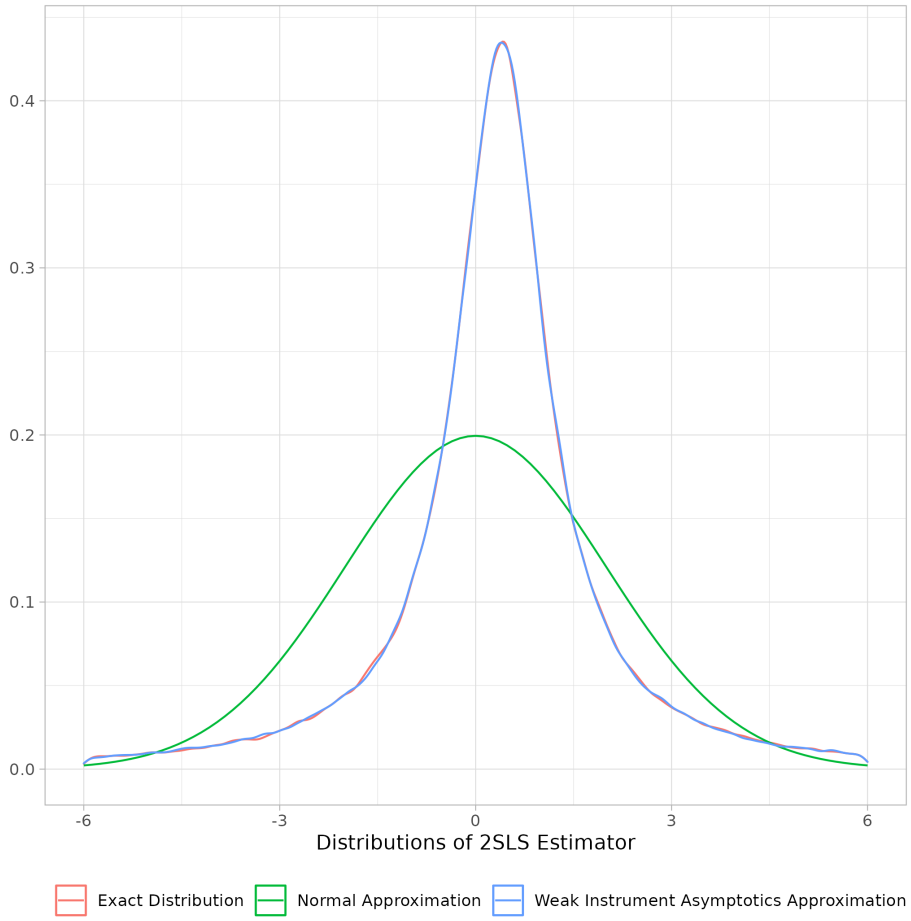

Figure 1: Comparison of Exact, Normal Approximation, and Weak Instrument Asymptotics distributions of the 2SLS estimator.

$$\text{Draw} = \frac{U}{Q_Z\Delta + V} = \frac{U}{0.5 + V}$$

For the data driven approach, we estimate Δ, QZ, and the variance covariance matrix of U and V for each iteration, take their means, and simulate from that exactly as we do in the analytical approach. These lead to almost identical densities.

Following are plots of: 1) the densities of the weak instrument asymptotic distribution, the exact distribution, and the normal approximation, 2) the 3 approaches to the normal approximation and, 3) the two approaches to the weak instrument asymptotics distribution.

3.

Repeating just the full analysis for the LIML estimator, we get the following:

We find the exact same patterns because in the just identified setting, LIML and 2SLS are equivalent. This is because the first stage is essentially unused, as we must use our one instrument exactly.

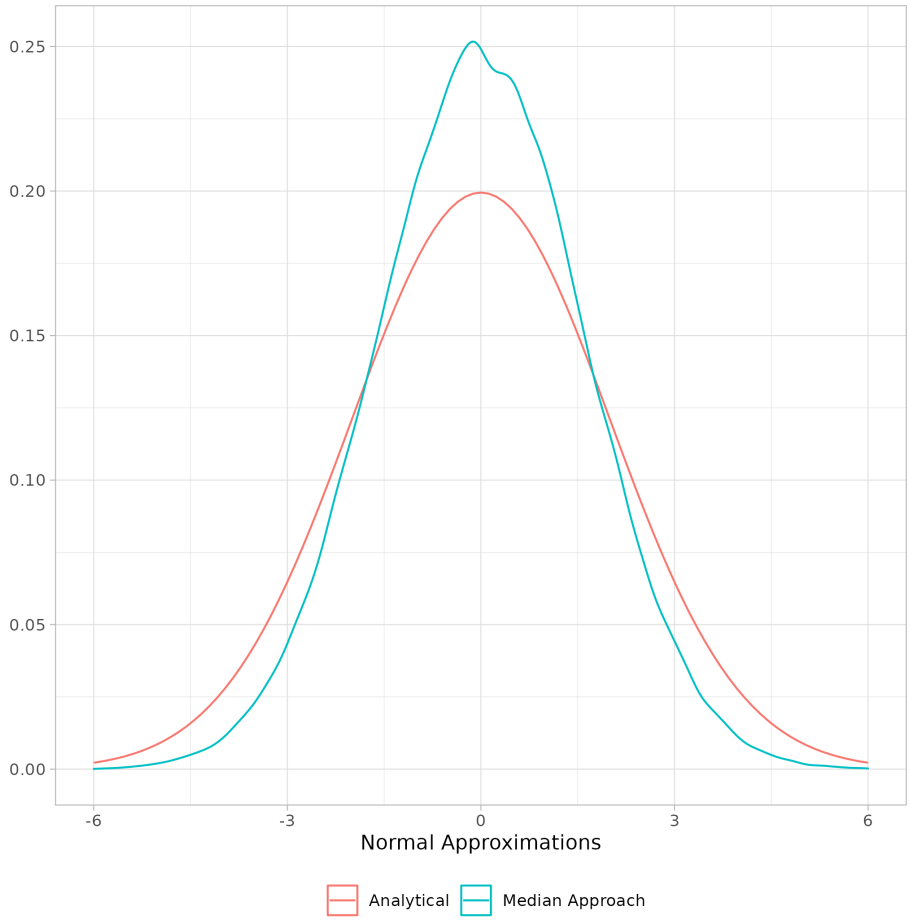

Figure 2: Comparison of median, and analytical approach to the normal asymptotic distribution.

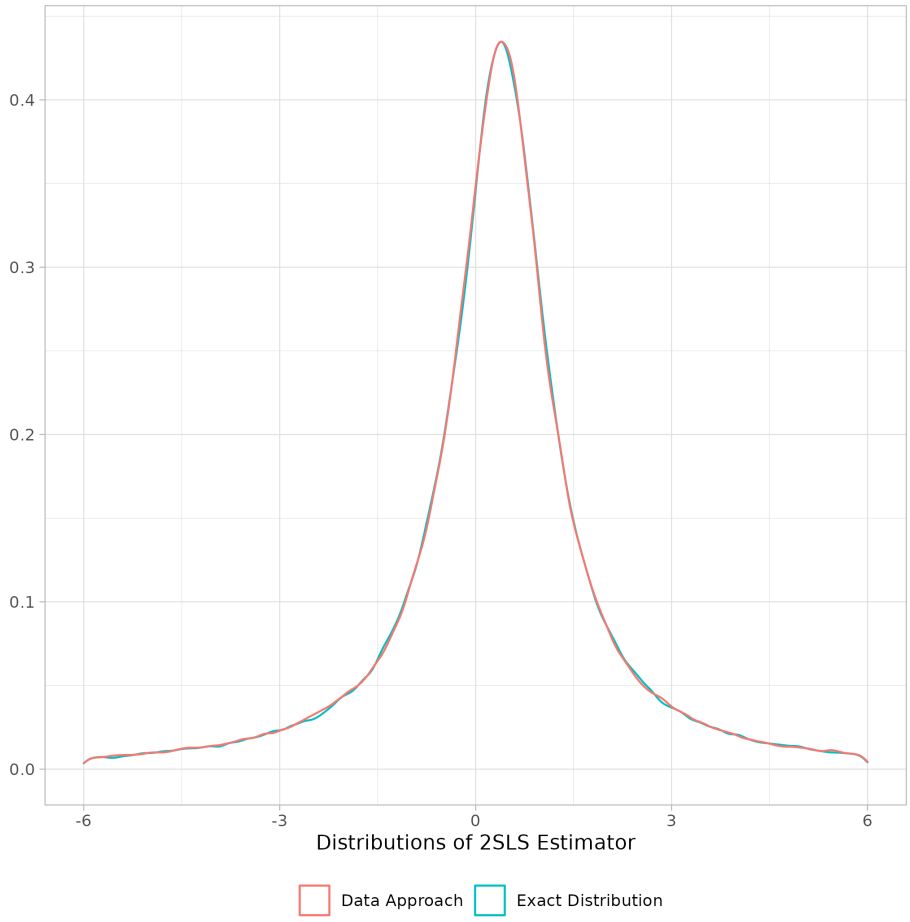

Figure 3: Comparison of analytical and data approach to the weak asymptotic approximation.

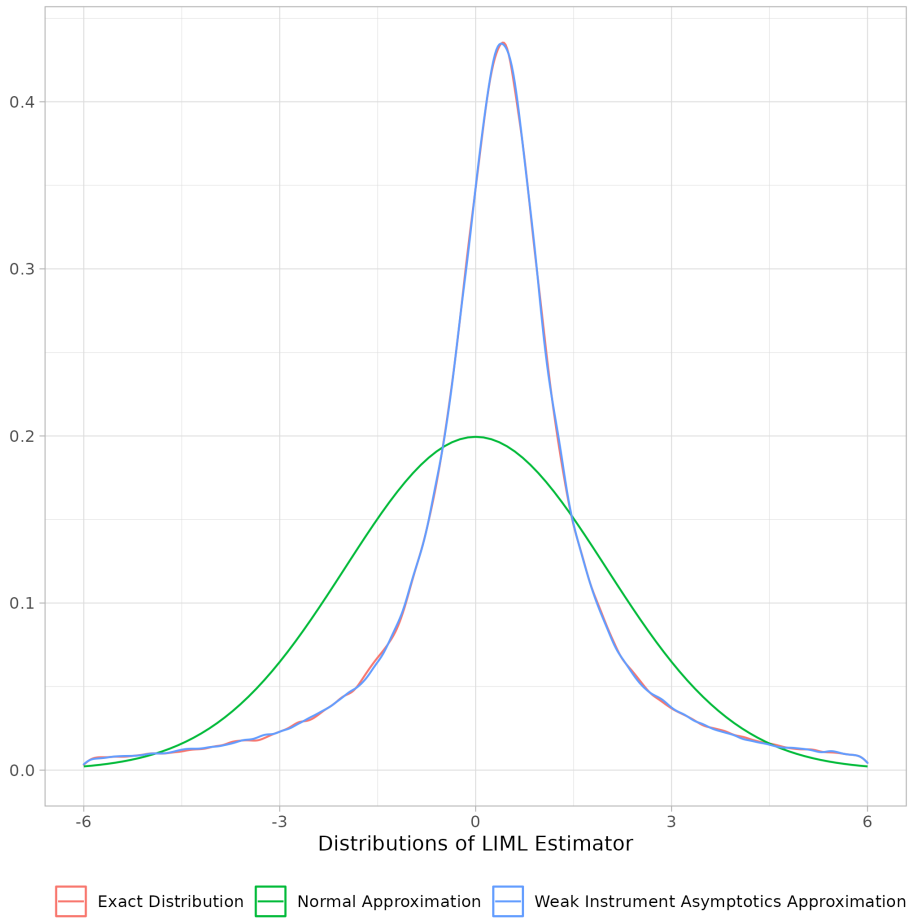

Figure 4: Comparison of Exact, Normal Approximation, and Weak Instrument Asymptotics distributions of the LIML estimator.

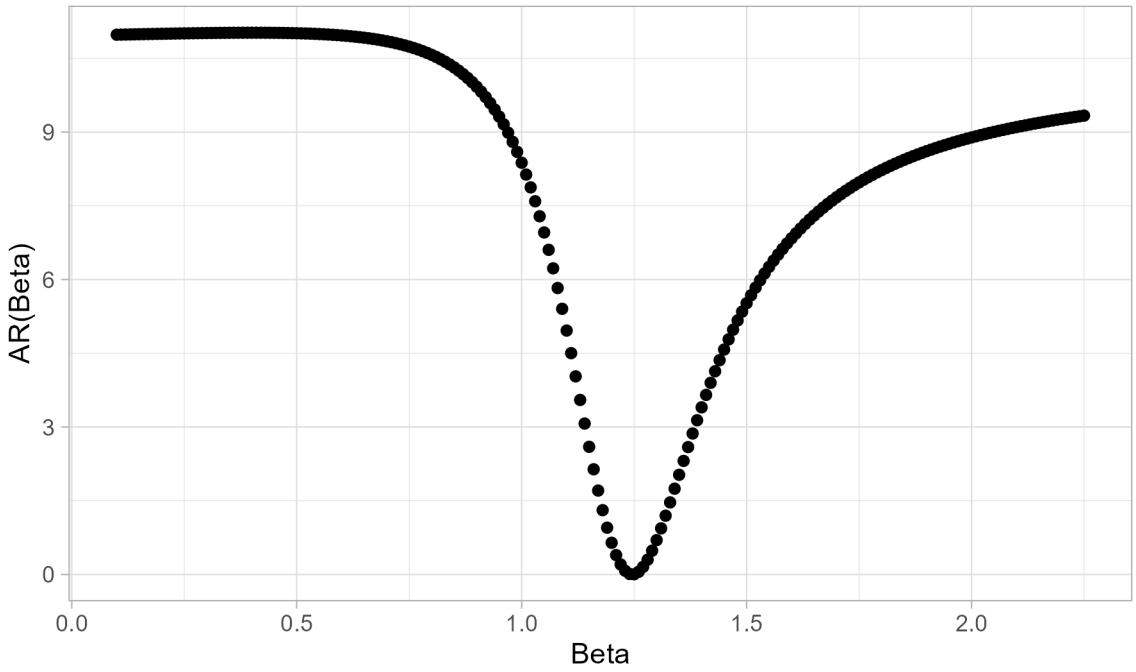

Figure 5: AR Statistic at a grid of width 0.01 between 0.1 and 2.25.

Question 3: Impact of Institutions on Economic Growth (Acemoglu, Johnson, and Robinson, 2001)

1.

See R code at the end of this document for the regressions. Estimates and standard errors will be reported in part 4.

2.

Figure 5 shows the value of the AR statistic calculated on a grid of width 0.01 between 0.1 and 2.25.

3.

The confidence intervals, point estimates, and standard errors of OLS, 2SLS, and AR are reported in table 3.

	$\hat{\beta}$	Standard Error	95% CI
OLS	0.487	0.064	(0.361, 0.614)
2SLS	1.262	0.542	(0.2, 2.324)
AR	1.27	0.071	(1.13, 1.41)

Table 3: Point estimates, standard errors, and 95% Confidence Intervals for the effect of institutions on GDP using OLS, 2SLS, and the Anderson-Rubin Statistic

The AR estimate is the center of the range of values of β for which the Anderson-Rubin test would not reject. The standard errors are created such that creating a 95% confidence interval using them will result in exactly the set of β's for which we would not reject the test.

4.

2SLS and AR have very similar point estimates, and the AR confidence interval is actually narrower. The fact that these two intervals overlap suggests that the results of Acemoglu, Johnson, and Robinson (2001) are indeed robust to the weak instrument problem.

Code Appendix

There were 4 R scripts used to perform the analysis in this problem set, each presented here in its own subsection. Some of the longer lines will be cut off. The scripts themselves can be provided if needed.

Question 1

question1.R

```
library(tidyverse)
library(readstata13)
library(fastDummies)
library(ivmodel)
library(doParallel)
library(foreach)
library(doRNG)
library(mvtnorm)
library(data.table)
library(stargazer)

rm(list = ls())

time = proc.time()

source("src/bekker_se.R")

hhn <- read.dta13("data/hhn.dta")

hhn_factors <- hhn %>%
  mutate(QOB = as.factor(QOB), STATE = as.factor(STATE), YOB = as.factor(YOB))


# Suppose instead we take last of every group to be excluded
hhn_factors$QOB <- relevel(hhn_factors$QOB, ref = tail(levels(hhn_factors$QOB), 1))
hhn_factors$YOB <- relevel(hhn_factors$YOB, ref = tail(levels(hhn_factors$YOB), 1))
hhn_factors$STATE <- relevel(hhn_factors$STATE, ref = tail(levels(hhn_factors$STATE), 1))

# This generates all of the interaction columns
interacts <- model.matrix(~ QOB + QOB:YOB + QOB:STATE, data = hhn_factors)

hhn_full <- cbind(hhn_factors, interacts)

# QOB1 is the excluded group, so we want it removed everywhere
cols_to_remove <- c(grep("QOB4", colnames(hhn_full)), 4, 7, 8)
hhn_all_inst <- hhn_full[,-cols_to_remove]
hhn_clean <- hhn_all_inst[,c(1:2, 6, 3:5, 7:186)]

# Write subset of the file so we can skip time consuming steps when we parallelize
hhn_export <- hhn_clean[,c(3:186)]
fwrite(hhn_export, file = "intermediate_data/hhn_export.csv")

# Remove extraneous datasets
rm(hhn, hhn_all_inst, hhn_factors, hhn_full, interacts, hhn_export, cols_to_remove)

#### Do some sort of validation exercise w/ Jimmy/Erin
```

```
# subse <- hhn_clean %>%
#   filter(`QOB2:YOB31` == 1) %>%
#   select(QOB, YOB, QOB2, `QOB2:YOB31`)

# The same across specifications
wage <- hhn_clean$LWKLYWGE
educ <- hhn_clean$EDUC
controls <- hhn_clean[,3:6] # Including intercept

# Different instrument specifications
instr_s1 <- hhn_clean[,7:9] # QOB only
instr_s2 <- hhn_clean[,7:36] # QOB + QOB*YOB
instr_s3 <- hhn_clean[,7:186] # QOB + QOB*YOB + QOB*STATE

# IVmodels
mod_s1 <- ivmodel(Y=wage, D=educ, Z=instr_s1, X=controls, intercept = FALSE)
mod_s2 <- ivmodel(Y=wage, D=educ, Z=instr_s2, X=controls, intercept = FALSE)
mod_s3 <- ivmodel(Y=wage, D=educ, Z=instr_s3, X=controls, intercept = FALSE)

tsls_est = c(coef(mod_s1)["TSLS","Estimate"], coef(mod_s2)["TSLS","Estimate"], coef(mod_s3)["TSLS","Esti
tsls_se = c(coef(mod_s1)["TSLS","Std. Error"], coef(mod_s2)["TSLS","Std. Error"], coef(mod_s3)["TSLS","S
liml_est = c(coef(mod_s1)["LIML","Estimate"], coef(mod_s2)["LIML","Estimate"], coef(mod_s3)["LIML","Esti
liml_se = c(coef(mod_s1)["LIML","Std. Error"], coef(mod_s2)["LIML","Std. Error"], coef(mod_s3)["LIML","S
bekker = c(bekker_se(mod_s1, wage, educ, instr_s1, controls), bekker_se(mod_s2, wage, educ, instr_s2, co

base_results = rbind(tsls_est, tsls_se, liml_est, liml_se, bekker)
colnames(base_results) = c("S1", "S2", "S3")

stargazer(base_results, out = "out/results_last_excluded.tex")


######### Part (b)
# Collect parameters for simulation

# Betas
non_edg_coefs = coefOther(mod_s1)$LIML[, "Estimate"]
b0 = non_edg_coefs[1] #Intercept
b2 = non_edg_coefs[2:4] #Control estimates
b1 = coef(mod_s1)["LIML","Estimate"] #Endogenous beta
b_ = as.matrix(c(b0, b2, b1))

# To get the parameters for the equation defining X, we run OLS with X on
# instruments and controls
# Pis
mod_x <- lm(educ ~ hhn_clean$MARRIED + hhn_clean$RACE + hhn_clean$SMSA
            + hhn_clean$QOB1 + hhn_clean$QOB2 + hhn_clean$QOB3)
p_ = as.matrix(mod_x$coefficients)

# Errors
# U
X_ = cbind(1, hhn_clean$MARRIED, hhn_clean$RACE, hhn_clean$SMSA, educ)
u_ = wage - X_ %*% b_
sig_u = sd(u_)
```

```
#V
Z_ = cbind(1, hhn_clean$MARRIED, hhn_clean$RACE, hhn_clean$SMSA,
           hhn_clean$QOB1, hhn_clean$QOB2, hhn_clean$QOB3)
v_ = educ - Z_ %*%  p_
sig_v = sd(v_)

# Cov(u, v)
sig_uv = cor(u_, v_)

# Sigma for errors
sigma_hhn = c(sig_u^2, sig_uv, sig_uv, sig_v^2)

fwrite(list(b_), file = "intermediate_data/betas.txt")
fwrite(list(p_), file = "intermediate_data/pis.txt")
fwrite(list(sigma_hhn), file = "intermediate_data/sigma_hhn.txt")

num_datasets = 1000

ncore <- detectCores()
cl <- makeCluster(10, type = "PSOCK") #ncore - 1 usually
registerDoParallel(cl)

# From random.org
set.seed(191116266)
time_parallel <- system.time({
  liml_bekker <- foreach(i = 1:num_datasets
             , .combine = 'rbind'
             , .packages = c("ivmodel", "mvtnorm", "data.table")
             ) %dorng% {

               cat("Currently processing dataset", i, "\n")

               source("src/bekker_se.R")
               hhn_export <- fread("intermediate_data/hhn_export.csv")
               betas <- as.matrix(fread("intermediate_data/betas.txt"))
               pis <- as.matrix(fread("intermediate_data/pis.txt"))
               sigma_hhn <- matrix(as.matrix(fread("intermediate_data/sigma_hhn.txt")), nrow = 2, ncol =

               data_length = 329509
               errors = rmvnorm(data_length, sigma = sigma_hhn)

               Z_ = as.matrix(hhn_export[,1:7])

               X_fit = Z_ %*% pis + errors[,2]

               X_ = as.matrix(cbind(hhn_export[,1:4], X_fit))

               Y_fit = X_ %*% betas + errors[,1]

               # Name columns
               colnames(X_fit)[1] = "D"
               colnames(Y_fit)[1] = "Y"

               controls <- hhn_export[,1:4]
```

```
                # Different instrument specifications
                instr_s1 <- hhn_export[,5:7] # QOB only
                instr_s2 <- hhn_export[,5:34] # QOB + QOB*YOB
                instr_s3 <- hhn_export[,5:184] # QOB + QOB*YOB + QOB*STATE

                mod_s1 <- ivmodel(Y=Y_fit, D=X_fit, Z=instr_s1, X=controls, intercept = FALSE)
                mod_s2 <- ivmodel(Y=Y_fit, D=X_fit, Z=instr_s2, X=controls, intercept = FALSE)
                mod_s3 <- ivmodel(Y=Y_fit, D=X_fit, Z=instr_s3, X=controls, intercept = FALSE)

                tsls_est_s1 = coef(mod_s1)["TSLS","Estimate"]
                tsls_est_s2 = coef(mod_s2)["TSLS","Estimate"]
                tsls_est_s3 = coef(mod_s3)["TSLS","Estimate"]
                tsls_se_s1 = coef(mod_s1)["TSLS","Std. Error"]
                tsls_se_s2 = coef(mod_s2)["TSLS","Std. Error"]
                tsls_se_s3 = coef(mod_s3)["TSLS","Std. Error"]
                liml_est_s1 = coef(mod_s1)["LIML","Estimate"]
                liml_est_s2 = coef(mod_s2)["LIML","Estimate"]
                liml_est_s3 = coef(mod_s3)["LIML","Estimate"]
                liml_se_s1 = coef(mod_s1)["LIML","Std. Error"]
                liml_se_s2 = coef(mod_s2)["LIML","Std. Error"]
                liml_se_s3 = coef(mod_s3)["LIML","Std. Error"]
                bekker_s1 = bekker_se(mod_s1, Y_fit, X_fit, instr_s1, controls)
                bekker_s2 = bekker_se(mod_s2, Y_fit, X_fit, instr_s2, controls)
                bekker_s3 = bekker_se(mod_s3, Y_fit, X_fit, instr_s3, controls)

                out = c(tsls_est_s1, tsls_se_s1, liml_est_s1, liml_se_s1, bekker_s1, tsls_est_s2, tsls_s

                out
  }
})
stopCluster(cl)

cat("Time used for processing:", time_parallel[3], "seconds. \n")

simulated_estimates <- as.data.frame(liml_bekker)
colnames(simulated_estimates) <- c("tsls_est_s1", "tsls_se_s1", "liml_est_s1", "liml_se_s1", "bekker_s1

write.csv(simulated_estimates, file = "intermediate_data/sim_ests.csv")

cat("All done! :) \n", "Total analysis time:", (proc.time() - time)[3], " seconds. \n")

#####
# Assume all prior was run on cluster, need to load data here!
sim_est <- read.csv("intermediate_data/sim_ests.csv")
true_beta <- b1

sim_errors <- sim_est[,c(1, 3, 6, 8, 11, 13)] - true_beta

squared_errors <- sim_errors^2

ci_coverage <- between(0, sim_errors[,c(1:6, 2, 4, 6)] - qnorm(0.975)*sim_est[,c(2, 4, 7, 9, 12, 14, 5,
                          sim_errors[,c(1:6, 2, 4, 6)] + qnorm(0.975)*sim_est[,c(2, 4, 7, 9, 12, 14, 5,
)
```

```
bias <- c(mean(sim_errors[,1]), mean(sim_errors[,2]), NA, mean(sim_errors[,3]),
          mean(sim_errors[,4]), NA, mean(sim_errors[,5]), mean(sim_errors[,6]), NA)
rmse <- c(sqrt(mean(squared_errors[,1])), sqrt(mean(squared_errors[,2])), NA,
          sqrt(mean(squared_errors[,3])), sqrt(mean(squared_errors[,4])), NA,
          sqrt(mean(squared_errors[,5])), sqrt(mean(squared_errors[,6])), NA)
cove <- c(mean(ci_coverage[,1]), mean(ci_coverage[,2]), mean(ci_coverage[,7]),
          mean(ci_coverage[,3]), mean(ci_coverage[,4]), mean(ci_coverage[,8]),
          mean(ci_coverage[,5]), mean(ci_coverage[,6]), mean(ci_coverage[,9]))

results_out <- cbind(bias, rmse, cove)
rownames(results_out) <-  c("tsls_s1", "liml_s1", "bekker_s1",
                            "tsls_s2", "liml_s2", "bekker_s2",
                            "tsls_s3", "liml_s3", "bekker_s3")

stargazer(results_out, out = "out/simulation_est.tex")
write.csv(results_out, "out/simulation_results_qob4.csv")
```

bekker_se.R

```
##' bekker_se.R
##'
##' This procedure calculates standard errors as in Bekker (1994)
##'
##' @title Bekker standard error calculation
##'
##' @param ivmodel A fit IV model using the ivmodel package
##' @param Y The outcome variable
##' @param D The endogenous variable description
##' @param Z The instruments
##' @param X The control variables (must include intercept) description
##'
##' @return The standard error robust to many instruments
##'
##' @author Samuel Messer
##' @export
##'


bekker_se <- function(ivmodel, Y, D, Z, X)
{
  # Estimates from LIML
  b_tilde = as.vector(c(coefOther(ivmodel)$LIML[, "Estimate"], coef(ivmodel)["LIML","Estimate"]))
  # Vars used to estimate Y
  X_ = as.matrix(cbind(X, D))
  # Vars used to estimate X
  Z_ = as.matrix(cbind(X, Z))

  # Estimated errors
  u_tilde = Y - X_ %*% b_tilde

  # Useful quantities
```

```
  UU = as.numeric(t(u_tilde) %*% u_tilde)

  # Components of Bekker's SE
  sig_u_sq = UU/nrow(u_tilde)
  a_tilde = as.numeric(t(u_tilde) %*% Z_ %*% solve(t(Z_) %*% Z_) %*% t(Z_) %*% u_tilde / UU)
  X_tilde = X_ - (u_tilde %*% (t(u_tilde) %*% X_)) / UU
  H_hat = t(X_) %*% Z_ %*% solve(t(Z_) %*% Z_) %*% t(Z_) %*% X_ - a_tilde * (t(X_) %*% X_)
  Sig_hat = sig_u_sq * ((1 - a_tilde)^2 * t(X_tilde) %*% Z_ %*% solve(t(Z_) %*% Z_) %*% t(Z_) %*% X_til

  bekker_var = solve(H_hat) %*% Sig_hat %*% solve(H_hat)

  bekker_se = as.numeric(sqrt(diag(bekker_var))["D"])

  return(bekker_se)
}
```

Question 2

question2.R

```
#### Question 2
library(mvtnorm)
library(ivmodel)
library(ggplot2)
library(doParallel)
library(foreach)
library(doRNG)

rm(list = ls())

# We will run 200000 simulations
n_sims = 200000

ncore <- detectCores()
cl <- makeCluster(ncore - 2, type = "PSOCK") #ncore - 1 usually
registerDoParallel(cl)

# From random.org
set.seed(859255299)
time_parallel <- system.time({
  simulation_data <- foreach(i = 1:n_sims
                             , .combine = 'rbind'
                             , .packages = c("ivmodel", "mvtnorm")
  ) %dorng% {

    # Specified parameters
    beta = 0
    delta = 0.5
    n = 100
    sig_mat = matrix(c(1, 0.5, 0.5, 1), nrow = 2, ncol = 2)

    # Generate data
    z_i = rnorm(n)
    uv = rmvnorm(n, sigma = sig_mat)
    x_i = (delta/(sqrt(n))) * z_i + uv[,2]
```

```
    y_i = beta * x_i + uv[,1]

    # Get tsls and liml estimates
    mod <- ivmodel(Y = y_i, D = x_i, Z = z_i)

    b_tsls <- coef(mod)["TSLS", "Estimate"]
    b_liml <- coef(mod)["LIML", "Estimate"]

    # For 2SLS normal asymptotics, need var(x), var(u), R^2(x~z)
    # var(x)
    var_x = var(x_i)

    # var(u)
    u_i_hat = y_i - b_tsls * x_i
    var_u_hat = var(u_i_hat)
    var_u = var(uv[,1])

    # R^2(x~z)
    rsq = summary(lm(x_i ~ z_i))$r.squared

    # For weak asymptotic approx, need var(u), var(v), cov(u, v), delta, qz
    # var(u) from above

    # QZ
    Z_ = as.matrix(z_i)
    wa_qz = t(Z_) %*% Z_ / n

    # Delta is coef of regression of x~z multiplied by sqrt(n)

    wa_delta = summary(lm(x_i ~ z_i))$coef["z_i", "Estimate"] * sqrt(n)

    # var(v)
    v_i_hat = x_i - (wa_delta / sqrt(n)) * z_i
    var_v_hat = var(v_i_hat)
    var_v = var(uv[,2])

    # cov(u, v)
    cov_uv_hat = cov(u_i_hat, v_i_hat)
    cov_uv = cov(uv[,1], uv[,2])

    # The standard error calculated when we run 2SLS should be based off the asymptotic model
    tsls_se <- coef(mod)["TSLS", "Std. Error"]

    simulated_data = c(b_tsls, b_liml, var_x, var_u, rsq, var_v, cov_uv, wa_qz, wa_delta, tsls_se)

    simulated_data
  }
})
stopCluster(cl)

cat("Time used for processing:", time_parallel[3], "seconds. \n")

simulation_data = as.data.frame(simulation_data)
colnames(simulation_data) <- c("b_tsls", "b_liml", "var_x", "var_u", "rsq", "var_v", "cov_uv", "wa_qz",
```

```
# 3 different approaches to normal approximation

# First, simulate using estimates from data
simu_norm_var = mean(simulation_data$var_u) / (mean(simulation_data$var_x) * mean(simulation_data$rsq))
# But, we have to shrink by dividing by sqrt(n), which divides variance by n
simu_asym_var = simu_norm_var / 100
simulation_data$normal_data = rnorm(n_sims, 0, simu_norm_var)

# Second, consider standard errors from fitting TSLS model
simulation_data$normal_med = rnorm(n_sims, 0, median(simulation_data$tsls_se))

# Finally, use N(0, 4) from analytic calculation

# 2 Approaches for weak asymptotic
# First, data. We estimate v-cov matrix and QZ*delta
sig_hat = matrix(c(mean(simulation_data$var_u), mean(simulation_data$cov_uv),
                   mean(simulation_data$cov_uv), mean(simulation_data$var_v)),
                 nrow = 2, ncol = 2)
qz_delt = mean(simulation_data$wa_qz) * mean(simulation_data$wa_delta)

# And we can simulate using the weak asymptotic distribution
uv_draws = rmvnorm(n_sims, sigma = sig_hat)

simulation_data$wa_app = uv_draws[,1] / (qz_delt + uv_draws[,2])


# Second, analytical given we know everything
true_sig = matrix(c(1, 0.5, 0.5, 1), nrow = 2, ncol = 2)
true_delt_qz = 0.5

exact_draws = rmvnorm(n_sims, sigma = true_sig)
simulation_data$wa_exact =  exact_draws[,1] / (0.5 + exact_draws[,2])


ggplot(simulation_data) +
  geom_density(aes(x = b_tsls, color = "Exact Distribution")) +
  stat_function(fun = dnorm, args = list(mean = 0, sd = 2), aes(color = "Normal Approximation")) +
  geom_density(aes(x = wa_app, color = "Weak Instrument Asymptotics Approximation")) +
  scale_x_continuous(name = "Distributions of 2SLS Estimator", limits = c(-6, 6)) +
  scale_y_continuous(name = "") +
  #scale_color_manual(values = c("green", "blue", "red"), labels = c("Exact Distribution", "Normal Appro
  theme_light() +
  theme(legend.position = "bottom", legend.title = element_blank())
ggsave("out/tsls_asymptotics.png")

ggplot(simulation_data) +
  geom_density(aes(x = b_liml, color = "Exact Distribution")) +
  stat_function(fun = dnorm, args = list(mean = 0, sd = 2), aes(color = "Normal Approximation")) +
  geom_density(aes(x = wa_app, color = "Weak Instrument Asymptotics Approximation")) +
  scale_x_continuous(name = "Distributions of LIML Estimator", limits = c(-6, 6)) +
  scale_y_continuous(name = "") +
  #scale_color_manual(values = c("green", "blue", "red"), labels = c("Exact Distribution", "Normal Appro
  theme_light() +
```

```
  theme(legend.position = "bottom", legend.title = element_blank())
ggsave("out/liml_asymptotics.png")

ggplot(simulation_data) +
  geom_density(aes(x = wa_exact, color = "Exact Distribution")) +
  geom_density(aes(x = wa_app, color = "Data Approach")) +
  scale_x_continuous(name = "Distributions of 2SLS Estimator", limits = c(-6, 6)) +
  scale_y_continuous(name = "") +
  #scale_color_manual(values = c("red", "green"), labels = c("Exact Solution", "Data Approach")) +
  theme_light() +
  theme(legend.position = "bottom", legend.title = element_blank())
ggsave("out/wa_approximations.png")

ggplot(simulation_data) +
  stat_function(fun = dnorm, args = list(mean = 0, sd = 2), aes(color = "Analytical")) +
  geom_density(aes(x = normal_med, color = "Median Approach")) +
  scale_x_continuous(name = "Normal Approximations", limits = c(-6, 6)) +
  scale_y_continuous(name = "") +
  #scale_color_manual(values = c("green", "blue", "red"), labels = c("Data Approach", "Analytical Appro
  theme_light() +
  theme(legend.position = "bottom", legend.title = element_blank())
ggsave("out/normal_approximations_update.png")
```

Question 3

question3.R

```
#### Question 3
library(ggplot2)
library(ivreg)

# Read in data
ajr <- read.table("data/ajr.txt", header = T)

# OLS model
OLS_mod = lm(GDP ~ Exprop + Latitude, data = ajr)

# store estimates
OLS_beta = summary(OLS_mod)$coefficients[2,1]
OLS_se = summary(OLS_mod)$coefficients[2,2]

OLS_est = c(OLS_beta, OLS_se)

# 2SLS model
TSLS_mod = ivreg(GDP ~ Latitude + Exprop | Mort + Latitude, data = ajr)

# store estimates
TSLS_beta = summary(TSLS_mod)$coefficients[3,1]
TSLS_se = summary(TSLS_mod)$coefficients[3,2]

TSLS_est = c(TSLS_beta, TSLS_se)

part1_out = cbind(OLS_est, TSLS_est)

# Report estimates
```

```
stargazer(part1_out, out = "out/ajr_ols_tsls.tex")

# AR statistic function
AR_calc <- function(Z, Y, X, beta) {
  n = length(Y)

  PZ <-  Z %*% solve(t(Z) %*% Z) %*% t(Z)
  num <- t(Y - X * beta) %*% PZ %*% (Y - X * beta)
  denom <- t(Y - X * beta) %*% (Y - X * beta)
  AR_stat <- num/(denom/n)

  return(AR_stat)
}

# Setup grid
ar_grid = seq(from = 0.10, to = 2.25, by = 0.01)

# Calculate AR statistic on the grid
ar_stat = rep(NA, length(ar_grid))
for (i in 1:length(ar_grid)) {
  ar_stat[i] = AR_calc(Z = ajr$Mort, Y = ajr$GDP, X = ajr$Exprop, beta = ar_grid[i])
}

grid_stat = as.data.frame(cbind(ar_grid, ar_stat))

# Plot AR stats
ggplot(grid_stat, aes(x = ar_grid, y = ar_stat)) +
  geom_point() +
  theme_light() +
  scale_x_continuous(name = "Beta") +
  scale_y_continuous(name = "AR(Beta)")
ggsave("out/ar_stats.png", width = 6, height = 3.6)

# In AR range
grid_stat$not_rej = ifelse(grid_stat$ar_stat <= qchisq(0.95, 1), 1, 0)

non_rej_b = filter(grid_stat, not_rej == 1)
ar_min = min(non_rej_b$ar_grid)
ar_max = max(non_rej_b$ar_grid)

# AR CI
ar_center = (ar_min + ar_max) / 2
ar_se = (ar_max - ar_min) / (2 * qnorm(0.975))

# Calculate confidence intervals
ar_ci = paste0("(", round(ar_min, 3), ", ", round(ar_max, 3), ")")
tsls_ci = paste0("(", round(TSLS_beta - (qnorm(0.975) * TSLS_se), 3), ", ", round(TSLS_beta + (qnorm(0.9
ols_ci = paste0("(", round(OLS_beta - (qnorm(0.975) * OLS_se), 3), ", ", round(OLS_beta + (qnorm(0.975)

cis = c(ols_ci, tsls_ci, ar_ci)
betas = c(round(OLS_beta, 3), round(TSLS_beta, 3), round(ar_center, 3))
ses = c(round(OLS_se, 3), round(TSLS_se, 3), round(ar_se, 3))

part3_out = cbind(betas, ses, cis)
```

```
# Report estimates
stargazer(part3_out, out = "out/confidence_intervals.tex")
```