HW3

Erik Andersen

2023-10-30

Contents

```
# Load packages
pacman::p_load(tidyverse, here, haven, estimatr, magrittr,ivreg, gmm, ivmodel)

dr_here()
```

```
## here() starts at /Users/johannaallen/Documents/Erik/ECON 587.
## - This directory contains a file matching "[.]Rproj$" with contents matching "^Version: " in the firs
## - Initial working directory: /Users/johannaallen/Documents/Erik/ECON 587/HW3
## - Current working directory: /Users/johannaallen/Documents/Erik/ECON 587/HW3
```

Question 1

```
# In this chunk, were defining a bunch of convenience functions so the code is cleaner below

# True DGP: y_i = b0 + b1x_i + e_i
# Set n and sims
n = 10000
sims = 10000

# I'm learning functional programming, so this may not be the best way but I'm doing it anyways
# This function lets me generate functions that simulate x's with any variance for the error term
simulate = function(error_var){
    function(n){
      # x ~ N(10,4), eps ~ N(0,10)
      x_i = rnorm(n, 10, sqrt(4))
      error = rnorm(n, 0, sqrt(error_var))

      df = matrix(c(x_i, x_i + error), ncol = 2)
      colnames(df) = c("x", "x+u")
      return(df)
    }
}

# Generate our y variables from true date generating process
create_y = function(x, beta0 = 3, beta1 = 1){
  eps = rnorm(length(x), 0, sqrt(10))
  return(y = beta0 + beta1*x + eps)
```

```
}

# This function takes in a variance for measurement error and returns the coefficients of ols regressio
ols_coefs = function(measurement_error){
  # Generate regressors
  x = measurement_error(n)

  # Y needs to be generated with the actual values of x, not x with error
  y = create_y(x = x[,1])

  # Add constant to x
  x_mat = matrix(c(rep(1, nrow(x)), x[,2]), ncol = 2)

  # Run OLS
  coefs = solve(t(x_mat)%*%x_mat)%*%t(x_mat)%*%y
  names(coefs) = c("B0", "B1")
  return(coefs)

}
```

```
# X has no measurement error. Simulate betas a bunch of times
none_var = simulate(0) # zero mean for error means no measurment error

# Simulate ols
betas_none = sapply(1:sims, function(i) ols_coefs(none_var))

# Calculate mean and standard deviation of estimates
rowMeans(betas_none)
```

a)

```
##       B0       B1
## 2.997611 1.000145
```

```
apply(betas_none, 1, sd)
```

```
##         B0         B1
## 0.16122389 0.01575933
```

```
# Now x has measurment error with standard deviation of 1
one_var = simulate(1)

# simulate ols
betas_one = sapply(1:sims, function(i) ols_coefs(one_var))

# Calculate mean and standard deviation of estimates
rowMeans(betas_one)
```

c)

```
##        B0        B1
## 4.9996305 0.8000638
```

```
apply(betas_one, 1, sd)
```

```
##         B0         B1
## 0.15042230 0.01471798
```

```
# Now the measurement error has standard deviation 4
four_var = simulate(16)

betas_four = sapply(1:sims, function(i) ols_coefs(four_var))

# Calculate mean and standard deviation of estimates
rowMeans(betas_four)
```

d)

```
##         B0         B1
## 10.9993969  0.2000482
```

```
apply(betas_four, 1, sd)
```

```
##          B0          B1
## 0.088896187 0.008110746
```

```
# Manual two stage least squares. Reports coefficients from 1st and 2nd stage. This function is pretty
tsls = function(yvar, xvar, ivreg = FALSE){
  # Generate common x's
  x = rnorm(n, 10, sqrt(4))

  # Add two measurment errors. We're doing this without my fancy functions so the x's are common to bot
  x_1 = x + rnorm(n, 0, sqrt(yvar))
  x_2 = x + rnorm(n, 0, sqrt(xvar))

  # Generate y
  y = create_y(x)

  # Regress x_2 on x_1
  reg1 = lm(x_1~x_2)

  # Save Coefficients and fitted values
  coefs1 = coef(reg1)
  fitted_values = fitted(reg1)
```

```
  # Run 2sls regression using fitted values
  reg2 = lm(y ~ fitted_values)
  coefs2 = coef(reg2)
  se = summary(reg2)$coefficients[2,2]

  # If we want the ivreg, run that
  if(ivreg == TRUE){
    reg3 = ivreg::ivreg(y~x_1 | x_2)
    # Return coefficients and standard errors
    coefs_iv = coef(reg3)
    se_iv = summary(reg3)$coefficients[2,2] }
  else{
    coefs_iv = NULL
    se_iv = NULL }



  # Make list of things to return
  return(list(coefs_first = coefs1,
       coefs_second = coefs2,
       se = se,
       coefs_iv = coefs_iv,
       se_iv = se_iv))
}
```

```
# Loop over this for x_1 ~ x_2
measurment_error1 = lapply(1:sims, function(i) tsls(1,16))
```

```
# Mean and variance of first stage coefficients
# First we have to extract the coefficients from the list object the function we created makes
delist = function(data, output){
  unlist = sapply(1:length(data), function(i) data[[i]][output])
  sapply(1:length(unlist), function(i) unlist[[i]])
}

coefs1 = delist(measurment_error1, 1)

# This created a matrix, so now we can get means and standard errors
rowMeans(coefs1)
```

e)

```
## (Intercept)         x_2
##   8.0003245   0.1999719
```

```
apply(coefs1, 1, var)
```

```
##  (Intercept)          x_2
## 2.440252e-03 2.032022e-05
```

```
# Now report the mean and variance of 2sls estimator
coefs2 = delist(measurment_error1, 2)

rowMeans(coefs2)
```

f)

```
##   (Intercept) fitted_values
##      2.992671      1.000679
```

```
apply(coefs2, 1, var)
```

```
##   (Intercept) fitted_values
##   0.139213399   0.001381007
```

```
# Loop over generating function for x_2 ~ x_1
measurment_error2 = lapply(1:sims, function(i) tsls(16,1, ivreg = TRUE))
```

```
# Repeart e and f
coefs1 = delist(measurment_error2, 1)

rowMeans(coefs1)
```

g)

```
## (Intercept)         x_2
##   2.0015621   0.7998925
```

```
apply(coefs1, 1, var)
```

```
##  (Intercept)          x_2
## 0.0357491815 0.0003412068
```

```
coefs2 = delist(measurment_error2, 2)

rowMeans(coefs2)
```

```
##   (Intercept) fitted_values
##      2.992743      1.000733
```

```
apply(coefs2, 1, var)
```

```
##   (Intercept) fitted_values
##  0.0857767102  0.0008343966
```

```r
# Report mean of standard errors from each iteration of the loop
se = delist(measurment_error2,3)

mean(se)
```

h)

```
## [1] 0.01838495
```

```r
coefs_iv = delist(measurment_error2, 4)

rowMeans(coefs_iv)
```

i)

```
## (Intercept)         x_1
##    2.992743    1.000733
```

```r
apply(coefs_iv, 1, var)
```

```
##  (Intercept)          x_1
## 0.0857767102 0.0008343966
```

```r
# Calculate average se's from ivreg
se_iv = delist(measurment_error2, 5)
mean(se_iv)
```

```
## [1] 0.02855034
```

Question 2

```r
# Load data
din_df = read_dta(here('HW3', 'data', 'dinkelman_aer2011_Econ587.dta'))

# String of controls
controls = c("kms_to_subs0", "baseline_hhdens0", "base_hhpovrate0", "prop_head_f_a0", "sexratio0_a", "p

# Function to make a formula from our string and anything else
formula_generator = function(y, x, controls){
  lhs = paste(y, "~")
  x = paste(x, collapse = " + ")
  controls = paste(controls, collapse = " + ")
  rhs = ifelse(is.null(x),
               controls,
               paste(x, controls, sep = " + "))
```

```
  formula = paste(lhs, rhs, sep = " ")

  return(formula)
}
```

```
# Generate our formula
form_ols = formula_generator("d_prop_emp_f", "T", controls) |> as.formula()

# Naive ols estimation
(ols = lm_robust(form_ols, din_df, clusters = placecode0))
```

a)

```
##                       Estimate   Std. Error     t value     Pr(>|t|)
## (Intercept)       -6.692178e-02 0.0192691128 -3.47300782 0.0008327141
## T                 -5.169057e-04 0.0048657111 -0.10623437 0.9157366660
## kms_to_subs0       5.521302e-05 0.0002440812  0.22620754 0.8221727196
## baseline_hhdens0   5.097260e-06 0.0001036998  0.04915401 0.9621713209
## base_hhpovrate0    3.144518e-02 0.0100607144  3.12554140 0.0025936759
## prop_head_f_a0     4.671010e-02 0.0202394603  2.30787298 0.0233748041
## sexratio0_a        1.959733e-02 0.0075931837  2.58091106 0.0118027379
## prop_indianwhite0 -5.248575e-01 0.5597616371 -0.93764466 0.4029943971
## kms_to_road0       1.848799e-05 0.0001226735  0.15070892 0.8813917695
## kms_to_town0      -1.755723e-04 0.0001838111 -0.95517782 0.3432364856
## prop_matric_m0     7.196364e-02 0.0671332060  1.07195299 0.2872570707
## prop_matric_f0    -7.203229e-02 0.0986193713 -0.73040709 0.4678478758
## d_prop_waterclose  2.176839e-02 0.0059819965  3.63898342 0.0005141136
## d_prop_flush       6.162984e-02 0.0486971668  1.26557345 0.2183958110
## idcc1             -7.151120e-03 0.0098160866 -0.72851024 0.4716050741
## idcc2              3.317607e-03 0.0154942919  0.21411804 0.8324202084
## idcc3             -3.844713e-03 0.0104893106 -0.36653633 0.7175806302
## idcc4             -7.737164e-03 0.0092356744 -0.83774759 0.4088022824
## idcc5             -1.463429e-02 0.0122610375 -1.19356069 0.3191213287
## idcc6             -7.363315e-04 0.0087679619 -0.08397978 0.9338421733
## idcc7              3.085450e-03 0.0101912483  0.30275486 0.7637347019
## idcc8              2.911778e-03 0.0095281667  0.30559691 0.7619602089
## idcc9             -2.319380e-02 0.0148229262 -1.56472463 0.1275806216
##                        CI Lower      CI Upper        DF
## (Intercept)       -0.1052675463 -0.0285760123 80.134697
## T                 -0.0102417849  0.0092079734 62.497116
## kms_to_subs0      -0.0004378361  0.0005482622 40.681865
## baseline_hhdens0  -0.0002401753  0.0002503698  6.991350
## base_hhpovrate0    0.0113757433  0.0515146155 69.220391
## prop_head_f_a0     0.0064829737  0.0869372332 87.151968
## sexratio0_a        0.0044714550  0.0347232084 75.157900
## prop_indianwhite0 -2.0975781312  1.0478631081  3.883425
## kms_to_road0      -0.0002339180  0.0002708940 25.487512
## kms_to_town0      -0.0005430831  0.0001919384 61.349821
## prop_matric_m0    -0.0618229192  0.2057502007 73.316391
## prop_matric_f0    -0.2691058482  0.1250412715 63.024971
```

```
## d_prop_waterclose  0.0098417195  0.0336950529 71.378384
## d_prop_flush      -0.0391342765  0.1623939597 22.891800
## idcc1             -0.0271466803  0.0128444411 31.965491
## idcc2             -0.0288090225  0.0354442374 22.077809
## idcc3             -0.0256396920  0.0179502652 21.300728
## idcc4             -0.0265995163  0.0111251885 29.977677
## idcc5             -0.0538527231  0.0245841384  2.973385
## idcc6             -0.0189341821  0.0174615191 21.707619
## idcc7             -0.0175484723  0.0237193722 37.842580
## idcc8             -0.0165233319  0.0223468885 30.910293
## idcc9             -0.0533987805  0.0070111852 31.687820
```

```
form_first = formula_generator('T', "mean_grad_new", controls) |> as.formula()

(first_stage = lm_robust(form_first, din_df, clusters = placecode0))
```

b)

```
##                       Estimate   Std. Error    t value    Pr(>|t|)
## (Intercept)        0.3126579063 0.1412323214  2.2137844 0.029689358
## mean_grad_new     -0.0077427714 0.0027770472 -2.7881310 0.006595224
## kms_to_subs0      -0.0010893734 0.0023979251 -0.4542984 0.652065753
## baseline_hhdens0   0.0012733298 0.0006398868  1.9899299 0.087945424
## base_hhpovrate0    0.0171213803 0.0720785787  0.2375377 0.812942285
## prop_head_f_a0     0.1545813215 0.1099748161  1.4056065 0.163393833
## sexratio0_a       -0.1214267291 0.0433542926 -2.8008006 0.006484161
## prop_indianwhite0 -1.1049521143 0.5047505152 -2.1891055 0.095927354
## kms_to_road0      -0.0009733132 0.0010428915 -0.9332833 0.359417124
## kms_to_town0       0.0007649052 0.0016094567  0.4752567 0.636287425
## prop_matric_m0    -0.1524773372 0.4298477363 -0.3547241 0.723817455
## prop_matric_f0     0.9840538950 0.4235190994  2.3235172 0.023415990
## d_prop_waterclose  0.0122466340 0.0503547759  0.2432070 0.808537607
## d_prop_flush       0.1548787500 0.1060178441  1.4608744 0.157616497
## idcc1              0.0405096480 0.1116014647  0.3629849 0.719003891
## idcc2              0.2316341199 0.1326704214  1.7459364 0.095211081
## idcc3             -0.0898734378 0.1080101519 -0.8320832 0.414644231
## idcc4              0.2036066026 0.1026455167  1.9835898 0.056631883
## idcc5              0.2740403682 0.1549147074  1.7689758 0.168204425
## idcc6              0.0362164111 0.0996896079  0.3632917 0.719804531
## idcc7             -0.1618498832 0.1162965479 -1.3916998 0.172111284
## idcc8             -0.1419788282 0.0996644780 -1.4245680 0.164238942
## idcc9             -0.1547622031 0.1030284978 -1.5021301 0.142985115
##                       CI Lower     CI Upper        DF
## (Intercept)        0.0315989079  0.593716905 80.042265
## mean_grad_new     -0.0132676777 -0.002217865 81.531952
## kms_to_subs0      -0.0059355096  0.003756763 40.066648
## baseline_hhdens0  -0.0002476245  0.002794284  6.825818
## base_hhpovrate0   -0.1266667304  0.160909491 69.127180
## prop_head_f_a0    -0.0639982981  0.373160941 87.204735
## sexratio0_a       -0.2077965587 -0.035056899 74.807132
## prop_indianwhite0 -2.5239443127  0.314040084  3.877864
```

```
## kms_to_road0      -0.0031189299  0.001172303 25.529984
## kms_to_town0      -0.0024528662  0.003982677 61.506866
## prop_matric_m0    -1.0091400458  0.704185371 73.114465
## prop_matric_f0     0.1375977949  1.830509995 62.544238
## d_prop_waterclose -0.0881362456  0.112629514 71.896515
## d_prop_flush      -0.0644713074  0.374228807 22.933043
## idcc1             -0.1868186822  0.267837978 31.987135
## idcc2             -0.0440095660  0.507277806 21.329844
## idcc3             -0.3143745060  0.134627631 21.183378
## idcc4             -0.0061299310  0.413343136 29.641324
## idcc5             -0.1989356448  0.747016381  3.240032
## idcc6             -0.1703535307  0.242786353 22.323656
## idcc7             -0.3972843514  0.073584585 37.978311
## idcc8             -0.3452183434  0.061260687 31.103941
## idcc9             -0.3647231391  0.055198733 31.620773
```

```
first_stage$fstatistic[1]
```

c)

```
##    value
## 8.211135
```

```
# Pvalue
pf(first_stage$fstatistic[1], first_stage$fstatistic[2], first_stage$fstatistic[3], lower.tail = FALSE)
```

```
##        value
## 8.084736e-20
```

```
# Extract t statistic
coefs = tidy(first_stage)[2,c(2:3,5)]

# Calculate t stat and square it
(f = (coefs[1]/coefs[2])^2)
```

d)

```
##   estimate
## 2 7.773675
```

```
# P-value
coefs[3]
```

```
##       p.value
## 2 0.006595224
```

```r
# Get fitted values from first stage
t_hat = fitted.values(first_stage)

# Re-estimate part 1 with fitted t
form_second = formula_generator("d_prop_emp_f", "T_hat", controls) |> as.formula()

din_df %<>% mutate(T_hat = t_hat)

(second_stage = lm_robust(form_second, din_df, clusters = placecode0))
```

e)

```
##                       Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)       -0.0903677371 0.0215613137 -4.1911981 6.384444e-05
## T_hat              0.0950786454 0.0450853776  2.1088577 3.802573e-02
## kms_to_subs0       0.0001659306 0.0002425745  0.6840398 4.983406e-01
## baseline_hhdens0  -0.0001320400 0.0001260123 -1.0478341 3.032156e-01
## base_hhpovrate0    0.0305119306 0.0101051623  3.0194399 3.542099e-03
## prop_head_f_a0     0.0329809020 0.0214444109  1.5379719 1.277286e-01
## sexratio0_a        0.0323338877 0.0095195114  3.3965911 1.053960e-03
## prop_indianwhite0 -0.4210682776 0.5637596052 -0.7468933 4.868337e-01
## kms_to_road0       0.0001021605 0.0001201909  0.8499852 4.019717e-01
## kms_to_town0      -0.0002286730 0.0001826035 -1.2522922 2.151995e-01
## prop_matric_m0     0.0795869237 0.0674954076  1.1791458 2.421322e-01
## prop_matric_f0    -0.1654147424 0.1070382717 -1.5453794 1.263357e-01
## d_prop_waterclose  0.0211736299 0.0058987094  3.5895360 6.052418e-04
## d_prop_flush       0.0455801782 0.0489005478  0.9320995 3.584941e-01
## idcc1             -0.0105944591 0.0099426993 -1.0655516 2.942448e-01
## idcc2             -0.0175431336 0.0188411547 -0.9311071 3.551607e-01
## idcc3              0.0027253593 0.0108972277  0.2500966 8.043937e-01
## idcc4             -0.0296128999 0.0133326644 -2.2210789 2.996705e-02
## idcc5             -0.0469944563 0.0197953654 -2.3740131 2.867810e-02
## idcc6             -0.0061099840 0.0088703218 -0.6888120 4.969652e-01
## idcc7              0.0136592059 0.0106117301  1.2871799 2.037543e-01
## idcc8              0.0156349862 0.0113680222  1.3753480 1.741069e-01
## idcc9             -0.0080396896 0.0156839453 -0.5126063 6.104744e-01
##                       CI Lower      CI Upper        DF
## (Intercept)       -0.1331924607 -0.0475430135 91.663699
## T_hat              0.0053817579  0.1847755330 81.531952
## kms_to_subs0      -0.0003260750  0.0006579362 35.912139
## baseline_hhdens0  -0.0003895627  0.0001254828 29.531311
## base_hhpovrate0    0.0103545794  0.0506692818 69.365726
## prop_head_f_a0    -0.0096494815  0.0756112855 85.956827
## sexratio0_a        0.0133969411  0.0512708343 82.114159
## prop_indianwhite0 -1.8447690052  1.0026324499  5.315446
## kms_to_road0      -0.0001431567  0.0003474776 30.430174
## kms_to_town0      -0.0005937429  0.0001363969 61.570463
## prop_matric_m0    -0.0549093835  0.2140832310 73.710674
## prop_matric_f0    -0.3785392619  0.0477097771 77.361871
## d_prop_waterclose  0.0094122319  0.0329350279 71.102724
## d_prop_flush      -0.0541565134  0.1453168698 30.974202
## idcc1             -0.0308114272  0.0096225090 33.506308
```

```
## idcc2             -0.0551543036  0.0200680363 66.608088
## idcc3             -0.0196271641  0.0250778826 27.175143
## idcc4             -0.0562587890 -0.0029670107 62.677253
## idcc5             -0.0885218108 -0.0054671019 18.377592
## idcc6             -0.0243318377  0.0121118696 26.337171
## idcc7             -0.0076368084  0.0349552203 51.793598
## idcc8             -0.0071016688  0.0383716412 60.358165
## idcc9             -0.0395385060  0.0234591268 50.213270
```

```
# the function takes a different type of object than lm, so I can't use my fancy formula generator :(
iv_reg = ivreg(d_prop_emp_f ~ T +kms_to_subs0 + baseline_hhdens0 + base_hhpovrate0 + prop_head_f_a0 + s

summary(iv_reg)
```

g)

```
##
## Call:
## ivreg(formula = d_prop_emp_f ~ T + kms_to_subs0 + baseline_hhdens0 +
##     base_hhpovrate0 + prop_head_f_a0 + sexratio0_a + prop_indianwhite0 +
##     kms_to_road0 + kms_to_town0 + prop_matric_m0 + prop_matric_f0 +
##     d_prop_waterclose + d_prop_flush + idcc1 + idcc2 + idcc3 +
##     idcc4 + idcc5 + idcc6 + idcc7 + idcc8 + idcc9 | mean_grad_new +
##     kms_to_subs0 + baseline_hhdens0 + base_hhpovrate0 + prop_head_f_a0 +
##     sexratio0_a + prop_indianwhite0 + kms_to_road0 + kms_to_town0 +
##     prop_matric_m0 + prop_matric_f0 + d_prop_waterclose + d_prop_flush +
##     idcc1 + idcc2 + idcc3 + idcc4 + idcc5 + idcc6 + idcc7 + idcc8 +
##     idcc9, data = din_df, method = "OLS")
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.513706 -0.035613  0.003398  0.036802  0.499317
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -0.0903677  0.0206631  -4.373 1.29e-05 ***
## T                  0.0950786  0.0546590   1.739  0.08212 .
## kms_to_subs0       0.0001659  0.0002426   0.684  0.49409
## baseline_hhdens0  -0.0001320  0.0001001  -1.319  0.18730
## base_hhpovrate0    0.0305119  0.0102666   2.972  0.00300 **
## prop_head_f_a0     0.0329809  0.0218354   1.510  0.13111
## sexratio0_a        0.0323339  0.0113548   2.848  0.00446 **
## prop_indianwhite0 -0.4210683  0.1523836  -2.763  0.00578 **
## kms_to_road0       0.0001022  0.0001295   0.789  0.43018
## kms_to_town0      -0.0002287  0.0001665  -1.373  0.16983
## prop_matric_m0     0.0795869  0.0712653   1.117  0.26424
## prop_matric_f0    -0.1654147  0.0840251  -1.969  0.04915 *
## d_prop_waterclose  0.0211736  0.0067650   3.130  0.00178 **
## d_prop_flush       0.0455802  0.0242682   1.878  0.06052 .
## idcc1             -0.0105945  0.0091916  -1.153  0.24922
## idcc2             -0.0175431  0.0160559  -1.093  0.27471
```

```
## idcc3              0.0027254  0.0114313   0.238  0.81159
## idcc4             -0.0296129  0.0156434  -1.893  0.05852 .
## idcc5             -0.0469945  0.0232922  -2.018  0.04378 *
## idcc6             -0.0061100  0.0093967  -0.650  0.51563
## idcc7              0.0136592  0.0122960   1.111  0.26678
## idcc8              0.0156350  0.0114970   1.360  0.17403
## idcc9             -0.0080397  0.0133444  -0.602  0.54693
##
## Diagnostic tests:
##                   df1  df2 statistic  p-value
## Weak instruments    1 1793    13.524 0.000242 ***
## Wu-Hausman          1 1792     3.998 0.045702 *
## Sargan              0   NA        NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07352 on 1793 degrees of freedom
## Multiple R-Squared: -0.1845, Adjusted R-squared: -0.1991
## Wald test: 6.011 on 22 and 1793 DF,  p-value: < 2.2e-16
```

```r
# generated reduced form formula
form_reduced = formula_generator('d_prop_emp_f', 'mean_grad_new', controls)

# Run regression
(reduced_reg = lm(form_reduced, din_df))
```

h)

```
##
## Call:
## lm(formula = form_reduced, data = din_df)
##
## Coefficients:
##       (Intercept)      mean_grad_new       kms_to_subs0   baseline_hhdens0
##         -6.064e-02         -7.362e-04          6.235e-05         -1.097e-05
##   base_hhpovrate0     prop_head_f_a0        sexratio0_a  prop_indianwhite0
##          3.214e-02          4.768e-02          2.079e-02         -5.261e-01
##      kms_to_road0       kms_to_town0     prop_matric_m0     prop_matric_f0
##          9.619e-06         -1.559e-04          6.509e-02         -7.185e-02
## d_prop_waterclose       d_prop_flush              idcc1              idcc2
##          2.234e-02          6.031e-02         -6.743e-03          4.480e-03
##              idcc3              idcc4              idcc5              idcc6
##         -5.820e-03         -1.025e-02         -2.094e-02         -2.667e-03
##              idcc7              idcc8              idcc9
##         -1.729e-03          2.136e-03         -2.275e-02
```

```r
# Wald estimator: reduced form coefficient/first stage coefficient
reduced_reg$coefficients[2]/first_stage$coefficients[2]
```

i)

```
## mean_grad_new
##    0.09507865
```

```
# Gmm estimator
(gmm_est = gmm(d_prop_emp_f ~ T +kms_to_subs0 + baseline_hhdens0 + base_hhpovrate0 + prop_head_f_a0 + s
```

j)

```
## Method
##  twoStep
##
## Objective function value:  3.103102e-30
##
##       (Intercept)                  T       kms_to_subs0   baseline_hhdens0
##       -0.09036774         0.09507865         0.00016593       -0.00013204
##   base_hhpovrate0     prop_head_f_a0        sexratio0_a  prop_indianwhite0
##         0.03051193         0.03298090         0.03233389       -0.42106828
##      kms_to_road0       kms_to_town0     prop_matric_m0     prop_matric_f0
##         0.00010216        -0.00022867         0.07958692       -0.16541474
## d_prop_waterclose       d_prop_flush              idcc1              idcc2
##         0.02117363         0.04558018        -0.01059446       -0.01754313
##              idcc3              idcc4              idcc5              idcc6
##         0.00272536        -0.02961290        -0.04699446       -0.00610998
##              idcc7              idcc8              idcc9
##         0.01365921         0.01563499        -0.00803969
```

```
# SE
gmm_est$vcov |> diag() |> sqrt()
```

```
##       (Intercept)                  T      kms_to_subs0  baseline_hhdens0
##      0.0221562080      0.0542875361      0.0002509053      0.0001229219
##   base_hhpovrate0    prop_head_f_a0       sexratio0_a prop_indianwhite0
##      0.0108722562      0.0231748242      0.0105514423      0.4691551708
##      kms_to_road0      kms_to_town0    prop_matric_m0    prop_matric_f0
##      0.0001261899      0.0001833220      0.0746456027      0.1009393179
## d_prop_waterclose      d_prop_flush             idcc1             idcc2
##      0.0068730313      0.0352748683      0.0105028140      0.0185503463
##             idcc3             idcc4             idcc5             idcc6
##      0.0116779640      0.0150186621      0.0223062243      0.0095543835
##             idcc7             idcc8             idcc9
##      0.0127528252      0.0123351377      0.0165952357
```

```
# LIML estimator
# Make an ivmodel object
form_liml = with(din_df, ivmodel(Y=as.numeric(d_prop_emp_f), D = as.numeric(T), Z = as.numeric(mean_gra

# Run LIML
liml = LIML(form_liml)
```

```r
# Point estimate and se
liml$point.est
```

```
##        Estimate
## [1,] 0.09507865
```

```r
liml$std.err
```

```
##      Std. Error
## [1,] 0.05465903
```

```r
# Add residuals to data
din_df %<>% mutate(resids = din_df$`T` - t_hat)

# We already calculated the fitted values from the first stage and added them to din_df as t_hat
form_exog = formula_generator("d_prop_emp_f", c("T", "resids"), controls) |> as.formula()

(exog = lm_robust(form_exog, din_df, cluster = placecode0))
```

k)

```
##                        Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept)       -0.0903677371 0.0215499011 -4.1934177 6.332297e-05
## T                  0.0950786454 0.0451055646  2.1079139 3.811006e-02
## resids            -0.0963166181 0.0451813456 -2.1317784 3.604326e-02
## kms_to_subs0       0.0001659306 0.0002421915  0.6851212 4.976660e-01
## baseline_hhdens0  -0.0001320400 0.0001264061 -1.0445696 3.046957e-01
## base_hhpovrate0    0.0305119306 0.0100930581  3.0230610 3.505083e-03
## prop_head_f_a0     0.0329809020 0.0214174678  1.5399067 1.272567e-01
## sexratio0_a        0.0323338877 0.0095034802  3.4023207 1.034816e-03
## prop_indianwhite0 -0.4210682776 0.5641943085 -0.7463178 4.871515e-01
## kms_to_road0       0.0001021605 0.0001201033  0.8506048 4.016369e-01
## kms_to_town0      -0.0002286730 0.0001826368 -1.2520640 2.152828e-01
## prop_matric_m0     0.0795869237 0.0674905050  1.1792314 2.420985e-01
## prop_matric_f0    -0.1654147424 0.1071131014 -1.5442998 1.265969e-01
## d_prop_waterclose  0.0211736299 0.0059011235  3.5880676 6.081603e-04
## d_prop_flush       0.0455801782 0.0489129738  0.9318627 3.586144e-01
## idcc1             -0.0105944591 0.0099603128 -1.0636673 2.950849e-01
## idcc2             -0.0175431336 0.0188597641 -0.9301884 3.556325e-01
## idcc3              0.0027253593 0.0108856132  0.2503634 8.041900e-01
## idcc4             -0.0296128999 0.0133783975 -2.2134863 3.051158e-02
## idcc5             -0.0469944563 0.0199016231 -2.3613379 2.943964e-02
## idcc6             -0.0061099840 0.0088968274 -0.6867599 4.982360e-01
## idcc7              0.0136592059 0.0106009712  1.2884863 2.033037e-01
## idcc8              0.0156349862 0.0113594604  1.3763846 1.737884e-01
## idcc9             -0.0080396896 0.0156797196 -0.5127445 6.103785e-01
##                        CI Lower      CI Upper        DF
## (Intercept)       -0.1331697961 -0.0475656780 91.663224
## T                  0.0053414370  0.1848158538 81.522357
```

```
## resids            -0.1862079621 -0.0064252741 81.319948
## kms_to_subs0      -0.0003252980  0.0006571591 35.912952
## baseline_hhdens0  -0.0003903652  0.0001262853 29.537559
## base_hhpovrate0    0.0103786721  0.0506451891 69.355600
## prop_head_f_a0    -0.0095959477  0.0755577516 85.952911
## sexratio0_a        0.0134288231  0.0512389522 82.111667
## prop_indianwhite0 -1.8458405622  1.0037040069  5.315796
## kms_to_road0      -0.0001429846  0.0003473055 30.410292
## kms_to_town0      -0.0005938106  0.0001364646 61.560725
## prop_matric_m0    -0.0548997331  0.2140735805 73.706770
## prop_matric_f0    -0.3786884088  0.0478589240 77.358374
## d_prop_waterclose  0.0094074001  0.0329398597 71.096299
## d_prop_flush      -0.0541817657  0.1453421220 30.974903
## idcc1             -0.0308471890  0.0096582708 33.508631
## idcc2             -0.0551914183  0.0201051511 66.611332
## idcc3             -0.0196036414  0.0250543599 27.167262
## idcc4             -0.0563501160 -0.0028756837 62.685886
## idcc5             -0.0887459811 -0.0052429315 18.369688
## idcc6             -0.0243861694  0.0121662013 26.340694
## idcc7             -0.0076152926  0.0349337044 51.785903
## idcc8             -0.0070846288  0.0383546011 60.347246
## idcc9             -0.0395300390  0.0234506598 50.212002
```