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
# 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)
}
```

```r
# 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|)  CI Lower  CI Upper
## (Intercept)       -6.69e-02   0.019269 -3.4730 0.000833 -0.105268 -0.028576
## T                 -5.17e-04   0.004866 -0.1062 0.915737 -0.010242  0.009208
## kms_to_subs0       5.52e-05   0.000244  0.2262 0.822173 -0.000438  0.000548
## baseline_hhdens0   5.10e-06   0.000104  0.0492 0.962171 -0.000240  0.000250
## base_hhpovrate0    3.14e-02   0.010061  3.1255 0.002594  0.011376  0.051515
## prop_head_f_a0     4.67e-02   0.020239  2.3079 0.023375  0.006483  0.086937
## sexratio0_a        1.96e-02   0.007593  2.5809 0.011803  0.004471  0.034723
## prop_indianwhite0 -5.25e-01   0.559762 -0.9376 0.402994 -2.097578  1.047863
## kms_to_road0       1.85e-05   0.000123  0.1507 0.881392 -0.000234  0.000271
## kms_to_town0      -1.76e-04   0.000184 -0.9552 0.343236 -0.000543  0.000192
## prop_matric_m0     7.20e-02   0.067133  1.0720 0.287257 -0.061823  0.205750
## prop_matric_f0    -7.20e-02   0.098619 -0.7304 0.467848 -0.269106  0.125041
## d_prop_waterclose  2.18e-02   0.005982  3.6390 0.000514  0.009842  0.033695
## d_prop_flush       6.16e-02   0.048697  1.2656 0.218396 -0.039134  0.162394
## idcc1             -7.15e-03   0.009816 -0.7285 0.471605 -0.027147  0.012844
## idcc2              3.32e-03   0.015494  0.2141 0.832420 -0.028809  0.035444
## idcc3             -3.84e-03   0.010489 -0.3665 0.717581 -0.025640  0.017950
## idcc4             -7.74e-03   0.009236 -0.8377 0.408802 -0.026600  0.011125
## idcc5             -1.46e-02   0.012261 -1.1936 0.319121 -0.053853  0.024584
## idcc6             -7.36e-04   0.008768 -0.0840 0.933842 -0.018934  0.017462
## idcc7              3.09e-03   0.010191  0.3028 0.763735 -0.017548  0.023719
## idcc8              2.91e-03   0.009528  0.3056 0.761960 -0.016523  0.022347
## idcc9             -2.32e-02   0.014823 -1.5647 0.127581 -0.053399  0.007011
##                      DF
## (Intercept)       80.13
## T                 62.50
## kms_to_subs0      40.68
## baseline_hhdens0   6.99
## base_hhpovrate0   69.22
## prop_head_f_a0    87.15
## sexratio0_a       75.16
## prop_indianwhite0  3.88
## kms_to_road0      25.49
## kms_to_town0      61.35
## prop_matric_m0    73.32
## prop_matric_f0    63.02
## d_prop_waterclose 71.38
## d_prop_flush      22.89
## idcc1             31.97
## idcc2             22.08
## idcc3             21.30
## idcc4             29.98
## idcc5              2.97
```

```
## idcc6             21.71
## idcc7             37.84
## idcc8             30.91
## idcc9             31.69
```

```
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|)  CI Lower CI Upper
## (Intercept)        0.312658    0.14123   2.214  0.02969  0.031599  0.59372
## mean_grad_new     -0.007743    0.00278  -2.788  0.00660 -0.013268 -0.00222
## kms_to_subs0      -0.001089    0.00240  -0.454  0.65207 -0.005936  0.00376
## baseline_hhdens0   0.001273    0.00064   1.990  0.08795 -0.000248  0.00279
## base_hhpovrate0    0.017121    0.07208   0.238  0.81294 -0.126667  0.16091
## prop_head_f_a0     0.154581    0.10997   1.406  0.16339 -0.063998  0.37316
## sexratio0_a       -0.121427    0.04335  -2.801  0.00648 -0.207797 -0.03506
## prop_indianwhite0 -1.104952    0.50475  -2.189  0.09593 -2.523944  0.31404
## kms_to_road0      -0.000973    0.00104  -0.933  0.35942 -0.003119  0.00117
## kms_to_town0       0.000765    0.00161   0.475  0.63629 -0.002453  0.00398
## prop_matric_m0    -0.152477    0.42985  -0.355  0.72382 -1.009140  0.70419
## prop_matric_f0     0.984054    0.42352   2.324  0.02342  0.137598  1.83051
## d_prop_waterclose  0.012247    0.05035   0.243  0.80854 -0.088136  0.11263
## d_prop_flush       0.154879    0.10602   1.461  0.15762 -0.064471  0.37423
## idcc1              0.040510    0.11160   0.363  0.71900 -0.186819  0.26784
## idcc2              0.231634    0.13267   1.746  0.09521 -0.044010  0.50728
## idcc3             -0.089873    0.10801  -0.832  0.41464 -0.314375  0.13463
## idcc4              0.203607    0.10265   1.984  0.05663 -0.006130  0.41334
## idcc5              0.274040    0.15491   1.769  0.16820 -0.198936  0.74702
## idcc6              0.036216    0.09969   0.363  0.71980 -0.170354  0.24279
## idcc7             -0.161850    0.11630  -1.392  0.17211 -0.397284  0.07358
## idcc8             -0.141979    0.09966  -1.425  0.16424 -0.345218  0.06126
## idcc9             -0.154762    0.10303  -1.502  0.14299 -0.364723  0.05520
##                      DF
## (Intercept)       80.04
## mean_grad_new     81.53
## kms_to_subs0      40.07
## baseline_hhdens0   6.83
## base_hhpovrate0   69.13
## prop_head_f_a0    87.20
## sexratio0_a       74.81
## prop_indianwhite0  3.88
## kms_to_road0      25.53
## kms_to_town0      61.51
## prop_matric_m0    73.11
## prop_matric_f0    62.54
## d_prop_waterclose 71.90
## d_prop_flush      22.93
## idcc1             31.99
```

```
## idcc2             21.33
## idcc3             21.18
## idcc4             29.64
## idcc5              3.24
## idcc6             22.32
## idcc7             37.98
## idcc8             31.10
## idcc9             31.62
```

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

c)

```
## value
##  8.21
```

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

```
##    value
## 8.08e-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     7.77
```

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

```
##   p.value
## 2  0.0066
```

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

# Re-estimate part 1 with fitted t
```

```r
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|)  CI Lower  CI Upper
## (Intercept)       -0.090368   0.021561  -4.191 6.38e-05 -0.133192 -0.047543
## T_hat              0.095079   0.045085   2.109 3.80e-02  0.005382  0.184776
## kms_to_subs0       0.000166   0.000243   0.684 4.98e-01 -0.000326  0.000658
## baseline_hhdens0  -0.000132   0.000126  -1.048 3.03e-01 -0.000390  0.000125
## base_hhpovrate0    0.030512   0.010105   3.019 3.54e-03  0.010355  0.050669
## prop_head_f_a0     0.032981   0.021444   1.538 1.28e-01 -0.009649  0.075611
## sexratio0_a        0.032334   0.009520   3.397 1.05e-03  0.013397  0.051271
## prop_indianwhite0 -0.421068   0.563760  -0.747 4.87e-01 -1.844769  1.002632
## kms_to_road0       0.000102   0.000120   0.850 4.02e-01 -0.000143  0.000347
## kms_to_town0      -0.000229   0.000183  -1.252 2.15e-01 -0.000594  0.000136
## prop_matric_m0     0.079587   0.067495   1.179 2.42e-01 -0.054909  0.214083
## prop_matric_f0    -0.165415   0.107038  -1.545 1.26e-01 -0.378539  0.047710
## d_prop_waterclose  0.021174   0.005899   3.590 6.05e-04  0.009412  0.032935
## d_prop_flush       0.045580   0.048901   0.932 3.58e-01 -0.054157  0.145317
## idcc1             -0.010594   0.009943  -1.066 2.94e-01 -0.030811  0.009623
## idcc2             -0.017543   0.018841  -0.931 3.55e-01 -0.055154  0.020068
## idcc3              0.002725   0.010897   0.250 8.04e-01 -0.019627  0.025078
## idcc4             -0.029613   0.013333  -2.221 3.00e-02 -0.056259 -0.002967
## idcc5             -0.046994   0.019795  -2.374 2.87e-02 -0.088522 -0.005467
## idcc6             -0.006110   0.008870  -0.689 4.97e-01 -0.024332  0.012112
## idcc7              0.013659   0.010612   1.287 2.04e-01 -0.007637  0.034955
## idcc8              0.015635   0.011368   1.375 1.74e-01 -0.007102  0.038372
## idcc9             -0.008040   0.015684  -0.513 6.10e-01 -0.039539  0.023459
##                      DF
## (Intercept)       91.66
## T_hat             81.53
## kms_to_subs0      35.91
## baseline_hhdens0  29.53
## base_hhpovrate0   69.37
## prop_head_f_a0    85.96
## sexratio0_a       82.11
## prop_indianwhite0  5.32
## kms_to_road0      30.43
## kms_to_town0      61.57
## prop_matric_m0    73.71
## prop_matric_f0    77.36
## d_prop_waterclose 71.10
## d_prop_flush      30.97
## idcc1             33.51
## idcc2             66.61
## idcc3             27.18
## idcc4             62.68
## idcc5             18.38
## idcc6             26.34
```

```
## idcc7             51.79
## idcc8             60.36
## idcc9             50.21
```

```
# the function takes a different type of object than lm, so I can't use my fancy formula generator :(
ivreg(d_prop_emp_f ~ T +kms_to_subs0 + baseline_hhdens0 + base_hhpovrate0 + prop_head_f_a0 + sexratio0_
```

g)

```
##
## Call:
## ivreg(formula = d_prop_emp_f ~ T + kms_to_subs0 + baseline_hhdens0 +     base_hhpovrate0 + prop_head
##
## Coefficients:
##       (Intercept)                  T       kms_to_subs0   baseline_hhdens0
##         -0.090368           0.095079           0.000166          -0.000132
##   base_hhpovrate0     prop_head_f_a0        sexratio0_a  prop_indianwhite0
##          0.030512           0.032981           0.032334          -0.421068
##      kms_to_road0       kms_to_town0     prop_matric_m0     prop_matric_f0
##          0.000102          -0.000229           0.079587          -0.165415
## d_prop_waterclose       d_prop_flush              idcc1              idcc2
##          0.021174           0.045580          -0.010594          -0.017543
##             idcc3              idcc4              idcc5              idcc6
##          0.002725          -0.029613          -0.046994          -0.006110
##             idcc7              idcc8              idcc9
##          0.013659           0.015635          -0.008040
```

```
# 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
##         -6.06e-02          -7.36e-04           6.24e-05          -1.10e-05
##   base_hhpovrate0     prop_head_f_a0        sexratio0_a  prop_indianwhite0
##          3.21e-02           4.77e-02           2.08e-02          -5.26e-01
##      kms_to_road0       kms_to_town0     prop_matric_m0     prop_matric_f0
##          9.62e-06          -1.56e-04           6.51e-02          -7.19e-02
## d_prop_waterclose       d_prop_flush              idcc1              idcc2
```

```
##          2.23e-02           6.03e-02          -6.74e-03           4.48e-03
##             idcc3              idcc4              idcc5              idcc6
##         -5.82e-03          -1.03e-02          -2.09e-02          -2.67e-03
##             idcc7              idcc8              idcc9
##         -1.73e-03           2.14e-03          -2.28e-02
```

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

i)

```
## mean_grad_new
##        0.0951
```

```
# 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:  3.1e-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
##          0.022156           0.054288           0.000251           0.000123
##   base_hhpovrate0    prop_head_f_a0       sexratio0_a prop_indianwhite0
##          0.010872           0.023175           0.010551           0.469155
##      kms_to_road0      kms_to_town0    prop_matric_m0    prop_matric_f0
##          0.000126           0.000183           0.074646           0.100939
## d_prop_waterclose      d_prop_flush              idcc1              idcc2
```

```
##          0.006873          0.035275          0.010503          0.018550
##             idcc3             idcc4             idcc5             idcc6
##          0.011678          0.015019          0.022306          0.009554
##             idcc7             idcc8             idcc9
##          0.012753          0.012335          0.016595
```

```
# 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)

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

```
##      Estimate
## [1,]   0.0951
```

```
liml$std.err
```

```
##      Std. Error
## [1,]     0.0547
```

```
# 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", "t_hat"), controls) |> as.formula()

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

k)

```
##                   Estimate Std. Error t value Pr(>|t|)  CI Lower  CI Upper
## (Intercept)       -0.090368   0.021550  -4.193 6.33e-05 -0.133170 -0.047566
## T                 -0.001238   0.004834  -0.256 7.99e-01 -0.010897  0.008421
## t_hat              0.096317   0.045181   2.132 3.60e-02  0.006425  0.186208
## kms_to_subs0       0.000166   0.000242   0.685 4.98e-01 -0.000325  0.000657
## baseline_hhdens0  -0.000132   0.000126  -1.045 3.05e-01 -0.000390  0.000126
## base_hhpovrate0    0.030512   0.010093   3.023 3.51e-03  0.010379  0.050645
## prop_head_f_a0     0.032981   0.021417   1.540 1.27e-01 -0.009596  0.075558
## sexratio0_a        0.032334   0.009503   3.402 1.03e-03  0.013429  0.051239
## prop_indianwhite0 -0.421068   0.564194  -0.746 4.87e-01 -1.845841  1.003704
## kms_to_road0       0.000102   0.000120   0.851 4.02e-01 -0.000143  0.000347
## kms_to_town0      -0.000229   0.000183  -1.252 2.15e-01 -0.000594  0.000136
## prop_matric_m0     0.079587   0.067491   1.179 2.42e-01 -0.054900  0.214074
## prop_matric_f0    -0.165415   0.107113  -1.544 1.27e-01 -0.378688  0.047859
## d_prop_waterclose  0.021174   0.005901   3.588 6.08e-04  0.009407  0.032940
## d_prop_flush       0.045580   0.048913   0.932 3.59e-01 -0.054182  0.145342
## idcc1             -0.010594   0.009960  -1.064 2.95e-01 -0.030847  0.009658
## idcc2             -0.017543   0.018860  -0.930 3.56e-01 -0.055191  0.020105
```

```
## idcc3              0.002725   0.010886   0.250 8.04e-01 -0.019604  0.025054
## idcc4             -0.029613   0.013378  -2.213 3.05e-02 -0.056350 -0.002876
## idcc5             -0.046994   0.019902  -2.361 2.94e-02 -0.088746 -0.005243
## idcc6             -0.006110   0.008897  -0.687 4.98e-01 -0.024386  0.012166
## idcc7              0.013659   0.010601   1.288 2.03e-01 -0.007615  0.034934
## idcc8              0.015635   0.011359   1.376 1.74e-01 -0.007085  0.038355
## idcc9             -0.008040   0.015680  -0.513 6.10e-01 -0.039530  0.023451
##                      DF
## (Intercept)       91.66
## T                 62.96
## t_hat             81.32
## kms_to_subs0      35.91
## baseline_hhdens0  29.54
## base_hhpovrate0   69.36
## prop_head_f_a0    85.95
## sexratio0_a       82.11
## prop_indianwhite0  5.32
## kms_to_road0      30.41
## kms_to_town0      61.56
## prop_matric_m0    73.71
## prop_matric_f0    77.36
## d_prop_waterclose 71.10
## d_prop_flush      30.97
## idcc1             33.51
## idcc2             66.61
## idcc3             27.17
## idcc4             62.69
## idcc5             18.37
## idcc6             26.34
## idcc7             51.79
## idcc8             60.35
## idcc9             50.21
```