ARE213 Problem Set #3

Peter Alstone & Frank Proulx

November 30, 2013

1 Problem 1: Linear models

1.1 Part A: LM results comparison

Using a series of linear models (with heteroskedasticity consistent "robust" standard errors), we find that for a range of model formulations there is a significant effect on housing price from the presence of hazardous waste cleanup sites, but not in the direction one would expect. Instead of a reduction in housing value we estimate an increase in the value, which does not seem likely to be true. The coefficient for the hazardous waste indicator variable (npl2000) takes a wide range of values depending on which additional explanatory variables are included in the model, from 0.04 (i.e., approximately a 4% increase) for the simple model only including 1980 housing values and npl2000 to estimate 2000 housing values, to 0.09 for a model including both housing and demographic characteristics.

Requirements for Unbiased Estimates [add to this]: For our estimates to be unbiased we would need to include all of the potential sources of variation in housing price in a linear model. A particular challenge is that there are very few sites with NPL2000 status (only 2% of sites), so while the overall sample size is large there is very little support for estimates related to NPL2000 status compared to other covariates.

Table 1: Linear models for effect of NPL(2000) on housing value (with many additional state fixed effects omitted)

	simple model	+housing char.	+demographics	+state fixed effects
	(1)	(2)	(3)	(4)
npl2000	0.040*** (0.012)	0.055*** (0.012)	0.090*** (0.010)	0.068*** (0.009)
lnmeanhs8	0.856*** (0.011)	0.866*** (0.018)	0.619*** (0.022)	0.514*** (0.022)
firestoveheat80		0.074*** (0.020)	0.182*** (0.023)	0.230*** (0.033)
nofullkitchen80		−1.776*** (0.176)	−0.751*** (0.164)	−0.559*** (0.152)
zerofullbath80		1.243*** (0.139)	1.044*** (0.124)	0.863*** (0.116)
bedrms1_80occ		0.421* (0.249)	0.404* (0.237)	0.240 (0.234)
bedrms2_80occ		−0.436* (0.229)	0.156 (0.216)	−0.004 (0.214)
bedrms3_80occ		−0.524** (0.230)	−0.147 (0.217)	−0.153 (0.214)
bedrms4_80occ		−0.111 (0.226)	0.004 (0.217)	−0.213 (0.214)
bedrms5_80occ		0.721*** (0.231)	0.732*** (0.222)	0.430* (0.220)
blt0_1yrs80occ		−0.216*** (0.045)	−0.010 (0.044)	0.109** (0.045)
blt2_5yrs80occ		−0.295*** (0.029)	0.011 (0.028)	0.039 (0.026)
blt6_10yrs80occ		−0.271*** (0.021)	−0.048** (0.021)	0.002 (0.021)
blt10_20yrs80occ		−0.242*** (0.017)	−0.136*** (0.015)	−0.123*** (0.014)
blt20_30yrs80occ		−0.191*** (0.017)	−0.181*** (0.014)	−0.156*** (0.013)
blt30_40yrs80occ		−0.190*** (0.026)	−0.121*** (0.025)	−0.104*** (0.023)
occupied80		0.730*** (0.050)	0.242*** (0.046)	−0.093** (0.044)
pop_den8			0.00001*** (0.00000)	0.00001*** (0.00000)
shrblk8			−0.161*** (0.014)	−0.058*** (0.013)
shrhsp8			−0.329*** (0.021)	−0.100*** (0.022)
child8			−0.630*** (0.058)	−0.431*** (0.052)
old8			−0.737*** (0.047)	−0.447*** (0.044)
shrfor8			1.377*** (0.048)	0.567*** (0.041)
ffh8			−0.006 (0.034)	−0.084*** (0.032)
smhse8			0.407*** (0.022)	0.323*** (0.022)
hsdrop8			0.010 (0.025)	0.042* (0.024)
no_hs_diploma8			−0.537*** (0.039)	−0.262*** (0.034)
ba_or_better8			0.112*** (0.034)	0.450*** (0.035)
unemprt8			−0.654*** (0.071)	−1.420*** (0.076)
povrat8			−0.275*** (0.051)	0.118** (0.048)
welfare8			1.271*** (0.070)	0.284*** (0.067)
avhhin8			0.00001*** (0.00000)	0.00001*** (0.00000)
as.factor(statefips)2				−0.129*** (0.027)
as.factor(statefips)4				0.011 (0.015)
as.factor(statefips)5				−0.150*** (0.025)
as.factor(statefips)6				0.340*** (0.017)
as.factor(statefips)8				0.207*** (0.015)
as.factor(statefips)9				0.157*** (0.015)
as.factor(statefips)10				0.230*** (0.018)
as.factor(statefips)11				0.102*** (0.024)
as.factor(statefips)12				−0.005 (0.013)
as.factor(statefips)13				0.182*** (0.015)
as.factor(statefips)15				0.081** (0.038)
as.factor(statefips)16				0.039* (0.020)

Notes:

***Significant at the 1 percent level.
**Significant at the 5 percent level.
*Significant at the 10 percent level.

Table 2: Contingency table for a range of factors by nbr

	N	0	1
		$N = 42540$	$N = 5705$
npl1990 : 0	48245	99% (42137)	94% (5362)
1		1% (403)	6% (343)
npl2000 : 0	48245	99% (41989)	92% (5271)
1		1% (551)	8% (434)
pop_den8	48245	674.4158 2887.1943 6473.8845	194.1788 961.2832 3344.8530
shrblk8	48245	0.002638 0.015964 0.082081	0.001940 0.011711 0.058053
shrhsp8	48245	0.0064900 0.0185855 0.0634445	0.0049900 0.0131040 0.0369590
child8	48245	0.2328310 0.2846455 0.3289673	0.2587950 0.2974520 0.3323570
shrfor8	48245	0.01912050 0.04256600 0.08668875	0.01689000 0.03570600 0.06528300
ffh8	48245	0.09759375 0.15094300 0.24082675	0.08582000 0.12909600 0.19591101
smhse8	48245	0.4122337 0.5306680 0.6283828	0.4513610 0.5558160 0.6362780
hsdrop8	48245	0.054545 0.111739 0.192307	0.056338 0.107438 0.182572
no_hs_diploma8	48245	0.1816847 0.2916233 0.4290175	0.2093909 0.3010590 0.4139572
ba_or_better8	48245	0.07740766 0.13831653 0.24082869	0.07717157 0.12516019 0.20234250
unemprt8	48245	0.03652375 0.05492750 0.08234525	0.04026800 0.05844100 0.08421600
povrat8	48245	0.0452915 0.0800220 0.1442960	0.0455540 0.0744300 0.1235710
welfare8	48245	0.0286895 0.0510035 0.0952380	0.0317910 0.0513860 0.0844040
favinc8	48245	18678.15 22858.37 27771.77	19537.96 22877.64 26884.79
avhhin8	48245	16223.85 20316.22 25271.48	17278.15 20688.77 24637.19
meanrnt80	48115	222.7207 268.2003 324.8214	224.3905 264.1910 313.5771
mdvalhs9	48245	43600 69000 125600	47300 72200 126112
meanrnt9	48190	389.6826 491.0426 635.9742	387.5005 487.7268 628.6454
mdvalhs0	48245	81900 120300 179700	87000 123100 167700
meanrnt0	48127	520.2832 645.7830 823.5361	520.2703 638.1877 800.3922
tothsun8	48245	873 1280 1739	891 1278 1718
ownocc8	48245	438 740 1080	544 831 1156
owner_occupied80	48245	0.4729324 0.6618154 0.7839525	0.5725154 0.7147436 0.8039216
bltlast5yrs80	48245	0.01945995 0.08701160 0.22249195	0.04517272 0.11816839 0.22092116
bltlast10yrs80	48245	0.06065490 0.21716790 0.45006267	0.12627292 0.26502535 0.42857143
firestoveheat80	48245	0.002498829 0.011919458 0.042651323	0.005777961 0.021126760 0.060995184
noaircond80	48245	0.1612903 0.3921197 0.6600715	0.2464638 0.4505289 0.6935139
nofullkitchen80	48245	0.004088785 0.009900990 0.020574207	0.004839685 0.010771993 0.020997375
zerofullbath80	48245	0.004034873 0.011940299 0.028044170	0.005586592 0.013818182 0.029411765
northeast : 0	48245	79% (33483)	66% (3779)
1		3 21% (9057)	34% (1926)
midwest : 0	48245	77% (32659)	78% (4449)
1		23% (9881)	22% (1256)
south : 0	48245	68% (28851)	76% (4327)
1		32% (13689)	24% (1378)
west : 0	48245	77% (32627)	80% (4560)
1		23% (9913)	20% (1145)

1.2 Part B: Comparing covariates

2 Problem 2: RDD setup

3 Problem 3: RDD First Stage

4 Problem 4: RDD Second Stage

5 Problem 5: Synthesis

6 Appendix: Code Listings

```
# Econometrics helper functions for [R]
#
# Peter Alstone and Frank Proulx
# 2013
# version 1
# contact: peter.alstone AT gmail.com

# Category: Data Management -------------


# Category: Data Analysis -----------------

# Function: Find adjusted R^2 for subset of data
# This requires a completed linear model...pull out the relevant y-values and residuals and feed them to function
# [TODO @Peter] Improve function so it can simply evaluate lm or glm object, add error handling, general clean up.
adjr2 <- function(y,resid){
  r2 <- 1-sum(resid^2) / sum((y-mean(y))^2)
  return(r2)
} #end adjr2


# Category: Plots and Graphics -------------

## Function for arranging ggplots. use png(); arrange(p1, p2, ncol=1); dev.off() to save.
require(grid)
vp.layout <- function(x, y) viewport(layout.pos.row=x, layout.pos.col=y)
arrange_ggplot2 <- function(..., nrow=NULL, ncol=NULL, as.table=FALSE) {
  dots <- list(...)
  n <- length(dots)
  if(is.null(nrow) & is.null(ncol)) { nrow = floor(n/2) ; ncol = ceiling(n/nrow)}
  if(is.null(nrow)) { nrow = ceiling(n/ncol)}
  if(is.null(ncol)) { ncol = ceiling(n/nrow)}
  ## NOTE see n2mfrow in grDevices for possible alternative
```

```
  grid.newpage()
  pushViewport(viewport(layout=grid.layout(nrow,ncol) ) )
  ii.p <- 1
  for(ii.row in seq(1, nrow)){
    ii.table.row <- ii.row
    if(as.table) {ii.table.row <- nrow - ii.table.row + 1}
    for(ii.col in seq(1, ncol)){
      ii.table <- ii.p
      if(ii.p > n) break
      print(dots[[ii.table]], vp=vp.layout(ii.table.row, ii.col))
      ii.p <- ii.p + 1
    }
  }
}

robust <- function(model){ #This calculates the Huber-White Robust standard
    errors -- code from http://thetarzan.wordpress.com/2011/05/28/
    heteroskedasticity-robust-and-clustered-standard-errors-in-r/
    s <- summary(model)
    X <- model.matrix(model)
    u2 <- residuals(model)^2
    XDX <- 0

    for(i in 1:nrow(X)) {
        XDX <- XDX +u2[i]*X[i,]%*%t(X[i,])
    }

# inverse(X'X)
    XX1 <- solve(t(X)%*%X)

#Compute variance/covariance matrix
    varcovar <- XX1 %*% XDX %*% XX1

# Degrees of freedom adjustment
    dfc <- sqrt(nrow(X))/sqrt(nrow(X)-ncol(X))

    stdh <- dfc*sqrt(diag(varcovar))

    t <- model$coefficients/stdh
    p <- 2*pnorm(-abs(t))
    results <- cbind(model$coefficients, stdh, t, p)
    dimnames(results) <- dimnames(s$coefficients)
    results
}

## Two functions for clustered standard errors below from: http://people.su.
    se/~ma/clustering.pdf -------

clx <-
  function(fm, dfcw, cluster){
    # R-codes (www.r-project.org) for computing
    # clustered-standard errors. Mahmood Arai, Jan 26, 2008.

    # The arguments of the function are:
    # fitted model, cluster1 and cluster2
    # You need to install libraries 'sandwich' and 'lmtest'

```

```
    # reweighting the var-cov matrix for the within model
    library(sandwich);library(lmtest)
    M <- length(unique(cluster))
    N <- length(cluster)
    K <- fm$rank
    dfc <- (M/(M-1))*((N-1)/(N-K))
    uj  <- apply(estfun(fm),2, function(x) tapply(x, cluster, sum));
    vcovCL <- dfc*sandwich(fm, meat=crossprod(uj)/N)*dfcw
    coeftest(fm, vcovCL) }

mclx <-
  function(fm, dfcw, cluster1, cluster2){
    # R-codes (www.r-project.org) for computing multi-way
    # clustered-standard errors. Mahmood Arai, Jan 26, 2008.
    # See: Thompson (2006), Cameron, Gelbach and Miller (2006)
    # and Petersen (2006).
    # reweighting the var-cov matrix for the within model

    # The arguments of the function are:
    # fitted model, cluster1 and cluster2
    # You need to install libraries 'sandwich' and 'lmtest'

    library(sandwich);library(lmtest)
    cluster12 = paste(cluster1,cluster2, sep="")
    M1  <- length(unique(cluster1))
    M2  <- length(unique(cluster2))
    M12 <- length(unique(cluster12))
    N   <- length(cluster1)
    K   <- fm$rank
    dfc1  <- (M1/(M1-1))*((N-1)/(N-K))
    dfc2  <- (M2/(M2-1))*((N-1)/(N-K))
    dfc12 <- (M12/(M12-1))*((N-1)/(N-K))
    u1j   <- apply(estfun(fm), 2, function(x) tapply(x, cluster1,  sum))
    u2j   <- apply(estfun(fm), 2, function(x) tapply(x, cluster2,  sum))
    u12j  <- apply(estfun(fm), 2, function(x) tapply(x, cluster12, sum))
    vc1   <-  dfc1*sandwich(fm, meat=crossprod(u1j)/N )
    vc2   <-  dfc2*sandwich(fm, meat=crossprod(u2j)/N )
    vc12  <- dfc12*sandwich(fm, meat=crossprod(u12j)/N)
    vcovMCL <- (vc1 + vc2 - vc12)*dfcw
    coeftest(fm, vcovMCL)}

## Function to compute ols standard errors , robust, clustered...
## Based on http://diffuseprior.wordpress.com/2012/06/15/standard-robust-and
    -clustered-standard-errors-computed-in-r/
ols.hetero <- function(form, data, robust=FALSE, cluster=NULL,digits=3){
  r1 <- lm(form, data)
  if(length(cluster)!=0){
    data <- na.omit(data[,c(colnames(r1$model),cluster)])
    r1 <- lm(form, data)
  }
  X <- model.matrix(r1)
  n <- dim(X)[1]
  k <- dim(X)[2]
  if(robust==FALSE & length(cluster)==0){
    se <- sqrt(diag(solve(crossprod(X)) * as.numeric(crossprod(resid(r1))/(n
        -k))))
    res <- cbind(coef(r1),se)
```

```
143   }
144   if(robust==TRUE){
145     u <- matrix(resid(r1))
146     meat1 <- t(X) %*% diag(diag(crossprod(t(u)))) %*% X
147     dfc <- n/(n-k)
148     se <- sqrt(dfc*diag(solve(crossprod(X)) %*% meat1 %*% solve(crossprod(X)
          )))
149     res <- cbind(coef(r1),se)
150   }
151   if(length(cluster)!=0){
152     clus <- cbind(X,data[,cluster],resid(r1))
153     colnames(clus)[(dim(clus)[2]-1):dim(clus)[2]] <- c(cluster,"resid")
154     m <- dim(table(clus[,cluster]))
155     dfc <- (m/(m-1))*((n-1)/(n-k))
156     uclust  <- apply(resid(r1)*X,2, function(x) tapply(x, clus[,cluster],
          sum))
157     se <- sqrt(diag(solve(crossprod(X)) %*% (t(uclust) %*% uclust) %*% solve
          (crossprod(X)))*dfc)
158     res <- cbind(coef(r1),se)
159   }
160   res <- cbind(res,res[,1]/res[,2],(1-pnorm(abs(res[,1]/res[,2])))*2)
161   res1 <- matrix(as.numeric(sprintf(paste("%.",paste(digits,"f",sep=""),sep=
        "",res)),nrow=dim(res)[1])
162   rownames(res1) <- rownames(res)
163   colnames(res1) <- c("Estimate","Std. Error","t value","Pr(>|t|)")
164   return(res1)
165 }
```

../util/are213–func.R