Appendices for

Hedonic Markets and Explicit Demands: Bid-Function Envelopes for Public Services and Neighborhood Amenities

John Yinger

The Maxwell School

Syracuse University

June 2013

Appendix A: Data Collection Procedures and Descriptive Statistics

Preliminaries. This appendix describes the procedures used to obtain the variables listed in Tables 2 to 4 of the text. It also presents descriptive statistics for the variables in these tables. Variables taken directly from the Brasington data set are not discussed (but are included in the tables).

In the following discussion, USGS stands for the U.S. Geological Survey. "Lat" stands for latitude. "Long" stands for longitude. Distance to a CBG means distance to the center of the CBG (using the latitude and longitude in the Brasington data set). All distances except commuting distances are translated into miles using the standard formula:

$$D_{Miles} = \sqrt{X^2 + Y^2}$$
$$X = 69.1\left(\text{Lat}_2 - \text{Lat}_1\right)$$
$$Y = 69.1\left(\text{Long}_2 - \text{Long}_1\right)\left(\cos\left\{\frac{\text{Lat}_1}{57.3}\right\}\right)$$

Distances to worksites are calculated using the more accurate spherical law of cosines formula.

Lakefront; Distance to Lake. To obtain the lakefront variables, I divided the shoreline of Lake Erie into seven approximately linear segments. Then I used the USGS website to identify the latitude and longitude of a series of points along each segment. These points were then treated as data for an estimation of the line describing each segment. Each line was Lat = a + b(Long). These variables were expressed in decimals, not minutes and seconds.

Each CBG was linked to the shoreline segment directly to its north. The next step was to find the shortest distance from a CBG to its assigned line segment. Using the standard formula for the distance between two points and the expression for the shoreline with the estimated coefficients, I solved for the values of latitude and longitude on the shoreline that minimized the

distance from the CBG to the shoreline. Plugging these values and the latitude and longitude of the CBG into the standard distance formula yields the desired distance measure. Let Y_i and X_i be latitude and longitude, respectively, for CBG i and let a and b be the estimated coefficients from the above regression. Then the formula for the minimum distance to the lake, $D_{\min}$, is:

$$D_{\min} = \sqrt{\left(\frac{X_i + bY_i - ab}{1+b^2} - X_i \right)^2 + \left(a + b\left(\frac{X_i + bY_i - ab}{1+b^2} \right) - Y_i \right)^2}$$

For some observations near one of the ends of a segment, the point on the "shoreline" identified by the above procedure was not actually on the shoreline but was instead a point on an extension of the line segment into the territory of another segment. This case is illustrated in Figure A-1. For a CBG located at point A, the above formula calculates the distance to point B, which is not correct because B is not on the shoreline. In these cases, the distance to Lake Erie was defined as the distance to the intersection between the two relevant line segments, which is point C in the figure. These cases are easy to identify because point B falls outside the region specified for line segment 1, which is identified here by the vertical line through point C.

In a few other cases, the linear approximation to the shoreline cuts off a piece of land that juts into the lake. These cases are easily identified as CBGs located to the north of their line segment. These CBGs were all assumed to be right on the lake, that is, to have a distance to the lake equal to zero.

Snowbelt 1; Snowbelt 2. Cleveland has an intense snowbelt in its eastern suburbs. The amount and location of the snowfall obviously varies from year to year, but in most years it is concentrated in an area about 10 miles from Lake Erie starting in the town of Pepper Pike and moving to the northeast. Because the shoreline of Lake Erie goes from southwest to northeast in this region, this snowbelt can also be described as being located about 10 miles southeast of Lake

Erie (and east of Pepper Pike). Cleveland snowfall maps are available from the National Weather Service at http://www.erh.noaa.gov/cle/climate/info/snowinfo.html.

I use the "distance to Lake Erie" variable to identify observations in the snowbelt. To be specific, the first snowbelt variable is distance to Lake Erie for all observations east of Pepper Pike (and only for observations within 20 miles of the lake). The second snowbelt variable is the square of the distance to Lake Erie for the same observations. If these variables are indeed capturing a snow belt effect, we would expect this quadratic specification to indicate a maximum effect about 10 miles from the lake. This effect could be positive or negative.

Ghetto; Near Ghetto. To define the ghetto variables, I plotted all the CBGs in which the population was at least 80 percent black. I then identified two sub-sets of these CBGs that were closely clustered together and defined ghetto tracts to be tracts that were both at least 80 percent black and in one of these clusters. The resulting set of tracts is identified in Figure A-2. I then found the population weighted centroid of each sub-set and the distance from each CBG outside the ghetto to the nearest of these two centroids. CBGs were identified as near the ghetto if this distance was less than five miles. Note that this distance is to the center of the ghetto, not to its outer edge.

Near Airport; Airport Distance. The space occupied by the Cleveland Hopkins Airport is shaped approximately like a circle with a radius of two miles. I used the USGS site to find the (approximate) center of this circle and then measured the distance from each CBG to this center.

Near Public; Elementary Score. The data for elementary schools comes from two sources: the 2001 report card file posted by the Ohio Department of Education (at http://ilrc.ode.state.oh.us/Downloads.asp) and the latitudes and longitudes available at the U.S. Geological Survey Site (www.usgs.gov). The Ohio DOE file gives the name, district, and address

of each elementary school that existed in 2001. (It lists other schools, too, but I did not collect any of that information.) In addition, this source gives the fourth-grade passing rates for each school for 1998-1999, 1999-2000, and 2000-2001 for each state test (citizenship, mathematics, reading, writing, and science). Following my treatment of the district level tests, I recorded the overall average passing rate averaged over the first two of the above years (except in a handful of schools for which only one of these years was available). I also recorded the average passing rate on the math, reading, and writing tests for these two years.

The Ohio DOE site identified 378 elementary schools in the school districts that appear in the Cleveland area subsample of the Brasington data. (None of these were charter schools, which are called community schools in Ohio. Ohio's charter school law passed in 1997, and there were few such schools in 2000. See the 2002-2003 *Annual Report on Community Schools*, http://education.ohio.gov/GD/Templates/Pages/ODE/ODEDetail.aspx?page=3&TopicRelationID=662&ContentID=42095&Content=61111.) Test scores are not available for 11.9 percent of these elementary schools, most of which were in small districts and none of which were in Cleveland, and missing scores were set to the district average. To obtain a location for each school, I first downloaded the list of schools in each county, complete with latitude and longitude, from the USGS site. This list is for 2009, so it includes some schools that did not exist in 2000 and does not include some schools that did exist. This source provided locations for about 70 percent of the 2001 schools. I searched for the addresses of the remaining schools on the 2000-2001 list on the USGS maps. Placing the cursor on each school location reveals its latitude and longitude.

The Ohio Department of Education web site provides information on the ethnicity of students and teachers in each school district. I collected information on the share of teachers and pupils who were African-American and the share who belonged to any minority group in 2000-

2001. These variables are all highly correlated. In preliminary regressions, the share of teachers in a minority group had the most explanatory power, so I decided to use it exclusively.

The other school variables are based on variables in the Brasington data set. The value-added measure is a school district's 6th grade passing rate on all 5 state tests in 2000-2001 minus its 4th grade passing rate on the same tests in 1998-99. These two test scores apply to the same student cohort, so the difference between them can be interpreted as a value-added measure for late elementary school. This value-added measure obviously does not correct for movement in and out of a cohort.

Corrected School Assignments. When I first calculated the distance to the nearest elementary school in an observation's school district, I found that some of the distances were unreasonably large (almost 50 miles!). As a result, I plotted the latitude and longitude of each observation with a different color for each assigned school district. This approach revealed a few observations that were assigned to the wrong school district in the original Brasington data (as indicated by an observation with one color code surrounded by observations of a different color). On the basis of these plots I corrected the school-district assignments for about 120 observations. Ambiguities were resolved with http://tax.ohio.gov/online_services/thefinderschooldistrict.stm, which indicates the school district associated with any latitude and longitude.

Crime Rates, Municipality Type, and City Population. Police and other municipal services are delivered through a variety of governments in Ohio. Most locations receive their services from a city, village, or township, but a few also receive police services from the county sheriff. Crime rates for municipalities are provided by the Ohio Department of Public Safety (at http://www.ocjs.ohio.gov/crime_stats_reports.stm). I collected crime data from this source for

1999, namely, violent and property crime per 1,000 residents. (Overall crime rates for 1997 are available in the Brasington data set, but I did not use this variable.)

These data required adjustments in a few cases. First, 1999 data were not available for about one-third of the seventy-five municipalities in the Cleveland area, so I used the closest year available, usually 2000 or 2001. Second, crime data were not available from this source for Rocky River, so I used an alternative source for the overall crime rate there (http://www.neighborhoodscout.com/oh/rocky-river/crime/) and divided this crime rate into sub-categories using other medium-sized cities in Cuyahoga County. Third, crime data were not available for five villages in Cuyahoga County, so I filled in the crime rates using data from four other villages in the county. All of these villages had populations below 4,000. For three small cities (Garfield Heights, Bedford, Middleberg Heights), data for 1999 were available but implausibly low, so I used later years in the first two case and rates from a nearby city of similar size (Brook Park) in the other.

Crime rates were also available by census tract within Cleveland from the "Neo Cando" web site maintained by Case-Western University (http://neocando.case.edu/cando/index.jsp). The crime rates used in the regression combine these two sources, so they reflect tracts in Cleveland and municipalities outside Cleveland.

Following Bowes and Ihlandfelt (2001), the crime variables are expressed per unit area, not per 1,000 people. Because property and violent crime are highly correlated, I accounted for them by defining four dummy variables: below-average levels of both types of crime (the omitted category), below-average property crime and above-average violent crime, above-average property crime and below-average violent crime, and above-average levels of both types of crime. The impact of crime on property values is complicated because both local crime and distance from

high-crime areas seem to matter. To separate these to effects, I first identified all the locations in which both property and violent crime exceed the 95th percentile of the observed distribution for these variables. These locations all turned out to be census tracts in Cleveland located in three clusters that were within about five miles of each other. See Figure A-2. I found the population-weighted centroid of each cluster, which I call a crime hot-spot, and then defined rings around each centroid. More specifically, I defined:

Crime Hotspot1 = CBG within ½ mile of the center of a crime hot-spot,
Crime Hotspot2 = CBG between ½ and 1 mile from the center of a crime hot-spot,
Crime Hotspot3 = CBG between 1 and 2 miles from the center of a crime hot-spot, and
Crime Hotspot4 = CBG between 2 and 5 miles from the center of a crime hot-spot.

I also determined from this source the type of government that provided police protection in 2000. For cities, villages, and townships, I also recorded the population of this government. Because larger governments tend to provide a wider range of services and because of possible economies or diseconomies of scale in the delivery of local government services, the level of police and other services may vary by population, even controlling for the city tax rate. The regressions include a quartic function of municipal population to control for these possibilities. The regressions also include dummy variables for villages, for locations that receive police services from a township, and for places that receive police services from the county sheriff. For the last type of place, the municipal population is entered as zero.

Smog; Smog Distance. The Brasington data set indicates the total release (in pounds) of pollutants into the air from facilities within each CBG. This figure includes emissions through "confined air streams," such as stacks, vents , and pipes as well as "fugitive" emissions. These data come from the U.S. Environmental Protection Agency's Toxic Release Inventory. To obtain the Smog variables, I plotted the 34 CBGs with at least 25,000 pounds of air pollutants per year.

This plot indicated that these CBGs all fell into one of 7 clusters. I then looked at the total pollution from each of these clusters and selected 3 clusters with more than 800,000 pounds of air pollutants. (The other 4 clusters had air emissions below 250,000 pounds.) Then I found the effluent-weighted centroid of each of these 3 clusters. The Smog variable identifies CBGs within 20 miles of one of these centroids. The Smog Distance variables indicate the distance from the CBG to the nearest of these centroids (given that the distance is less than 20). To account for wind patterns and perhaps other factors that influence the distribution of these emissions across space, I originally interacted the Smog and Smog Distance variables with direction indicators. Preliminary regressions indicated that direction did not matter, however, except that property values did not decline when moving away from a high-smog site to the northwest. This result reflects the particular geography of Cleveland. Two of the three sites are on Lake Erie, where moving to the northwest places one in the lake. The other site, which is the only one with sales to its northwest, is to the southeast of another site, so that moving to the northwest does not move one away from a smog concentration. Hence, housing prices are assumed not to decline with distance as one moves away from a smog concentration to the northwest. (If a variable for "distance to the northwest is included, its coefficient is close to zero and it is not close to statistically significant.)

Local Amenities; Freeway; Railroad; Shopping; Hospital; Small Airport; Big Park.

To obtain these variables, I copied the latitude and longitude of each CBG into the search command on the website for Google Maps. All of these features are highlighted on the maps and are therefore easy to identify around each CBG location. Many features were verified on the satellite pictures also available through Google Maps.

- The local amenities include neighborhood parks, golf courses, rivers, and lakes.
- Freeways were defined as limited access highways.

- Railroads with a visible end were checked on the satellite pictures. Many of these segments had been removed and were therefore obviously not counted.
- Small streams were not counted as rivers; all rivers were verified on the satellite pictures.
- All lakes were verified on the satellite pictures; industrial ponds were not counted as lakes.
- Shopping includes malls, shopping centers, and business districts as identified on the maps; all of them were verified on the satellite pictures.
- Hospitals were defined to include medical centers, but not nursing homes, mental hospitals, rehabilitation centers, or the hospital for the Merchant Marines.
- Small Airports exclude Cleveland Hopkins Airport and small "air parks."
- Big Parks were defined as county, metro, regional, state, and national parks, plus the Cleveland zoo.

For neighborhood amenities, freeways, and railroads, I determined whether the geographic feature was within one-quarter mile of the CBG. This distance was selected because one-quarter mile was thought to be a reasonable walking distance to neighborhood amenities and a reasonable indicator of the distance over which the noise and pollution from a freeway or railroad might be seen as a problem. Shopping, Hospital, Small Airport, and Big Park are switched on if the relevant geographic feature is within one mile of the CBG. One mile was selected as a reasonable indicator of convenient access or, in the case of airports, of noticeable air traffic. In all cases distance was measured as the crow flies, not along streets.

Historic District. Historic districts in the five-county Cleveland MSA are listed by the National Register of Historic Places (http://www.nationalregisterofhistoricplaces.com). This list indicates the name, address, size (in acres), and date of designation for the district; it also indicates whether the district contains any single family houses. I identified 38 registered historic districts

containing single family houses and then looked up their latitudes and longitudes using either http://ohio.hometownlocator.com or the USGS site. I included 5 districts designated in 2001 and 2002 under the assumption that it takes several years to obtain a designation, so that these districts were well-known historic sites in 2000; only two sales, both in Geauga County, took place in one of these sites. To link CBGs and historic districts, I calculated the radius of a circle that would enclose an area the size of the district and then placed a CBG in the district if the distance between the center of the CBG and the center of the historic district was less than or equal to this radius.

Public Housing. All public housing projects in Ohio in 2000 were identified in a spreadsheet posted by the Development Department of the Ohio State Government (www.development.ohio.gov/cdd/ohcp/PublicHousingProjects.xls). This spreadsheet also indicated the location of the project (latitude and longitude), whether the project was for the elderly, and the number of units the project contained. Based on this information, I created three categorical variables: whether the center of the CBG is (1) within one-half mile of an elderly housing project, (2) within one-half mile of a small (fewer than 200 units) family housing project, or (3) within one-half mile of a large (at least 200 units) family housing project.

Taxes. The original Brasington data set includes the 2000 school property tax rate and the 2000 school income tax rate, which is usually zero. I also collected data on city property taxes and on property tax exemptions. Although property is supposed to be assessed at market value in Ohio, assessments are not updated annually, and deviations from market value may arise. I found data on assessment-sales ratios for most of the jurisdictions in the Cleveland area in 2005. These data, which covered about 88 percent of my observations, were used this to convert the nominal tax rates in the Brasington data set into effective rates using the well-known relationship: $t = m(A/\mathrm{V})$. In words, the effective rate is the nominal rate multiplied by the assessment-sales ratio.

As it turns out, 98 percent of the observations with A/V data have an A/V ratio between 0.83 and 1.08, so this correction does not have a dramatic influence on the property tax rate variable. I also included a dummy variable for locations for which I did not have assessment/sales data and a dummy variable for locations that are not in a city, for which I did not have non-school property tax data. These data all come from the State of Ohio, and in particular from: http://tax.ohio.gov/channels/research/property_tax_statistics.stm.

Worksites. Worksites are identified using zip code data on employment and location from the Census web site. As indicated in the text of the paper, "The commuting variable is average employment-weighted distance to the Cleveland area's five major worksites. A worksite was defined as a set of zip codes with at least 5,000 jobs that are within 6 miles of a central point, that contain at least one zip code with 25,000 jobs, and that contain at least 45,000 total jobs. Because many jobs are clustered near a beltway around Cleveland, one worksite was defined as a ring 7 to 11 miles from the center of the downtown site. The five resulting centers are downtown Cleveland, the beltway, Mentor/Painesville, Bedford/Solon, and Lorain/Elyria. Except for the beltway, the worksite center was set near concentrations of business buildings (identified using satellite images on the U.S. Geological Survey web site) near the job-weighted centroid of the zip-code cluster. The radius of the beltway, 8.59 miles, was selected to minimize the squared employment-weighted distance to the associated zip codes. These five worksites accounted for 75.8 percent of the jobs in the Cleveland area in 2000, with job shares of 35.3, 40.7, 9.3, 7.7, and 7.0 percent, respectively. Each CBD was assigned to the closest worksite consistent with the requirement that the share of workers assigned to each site (based on CBG data) equals the share of employment located at that site (based on zip code data).

A map of these worksites is provide in Figure A-3. The employment shares are the weights used to obtain weighted commuting distance. Note that the squared term in the regressions is designed to approximate an envelope for bid-rent functions. This envelope is the analog to equations (16) and (17) in the text, and an individual bid-rent function is the analog to equation (12). I am currently working on another envelope paper that derives and estimates formal envelope expressions for commuting distance. This (unpublished) paper shows that a quadratic specification approximates the case with an income elasticity of demand for housing equal to unity and a value of σ_3 (which has the same meaning as in equation (14) in the text) equal to ½.

Descriptive Statistics. Table A-1 provides names for all these variables that link to the names given in the text. Table A-2 provides descriptive statistics for all these variables along with all the others listed in Tables 3 through 5 in the text. For some variables, this table complements Table 2 in the text, which facilitates interpretation by presenting these variables in raw form instead of the form in which they enter the regressions.

Figure A-1: Corner Solutions for Distance to Lake Erie

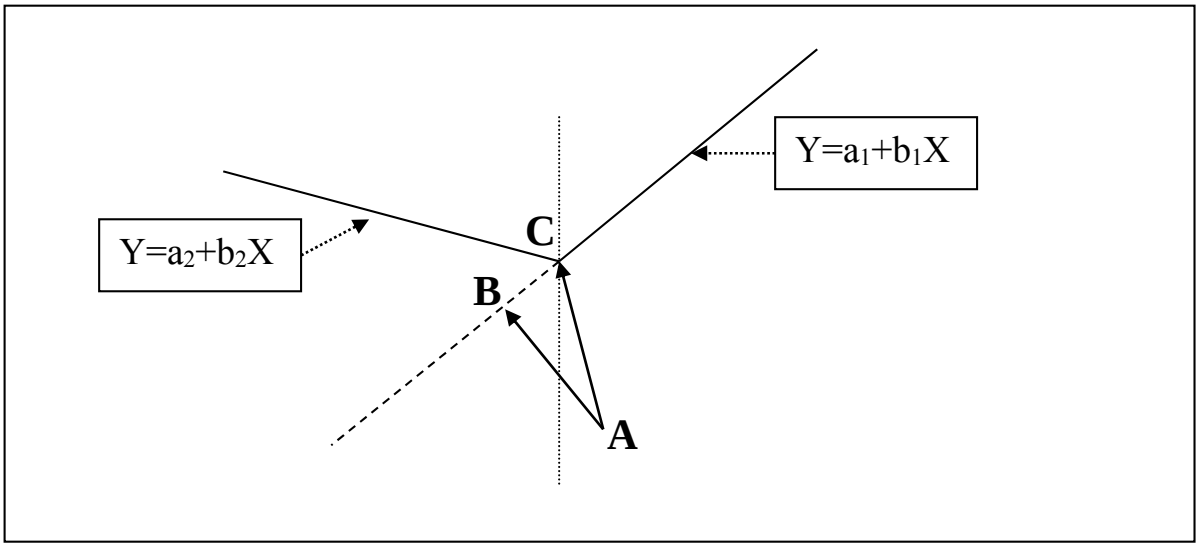


Figure A-2. Crime Hot Spots and Ghetto Tracts, 1999-2000

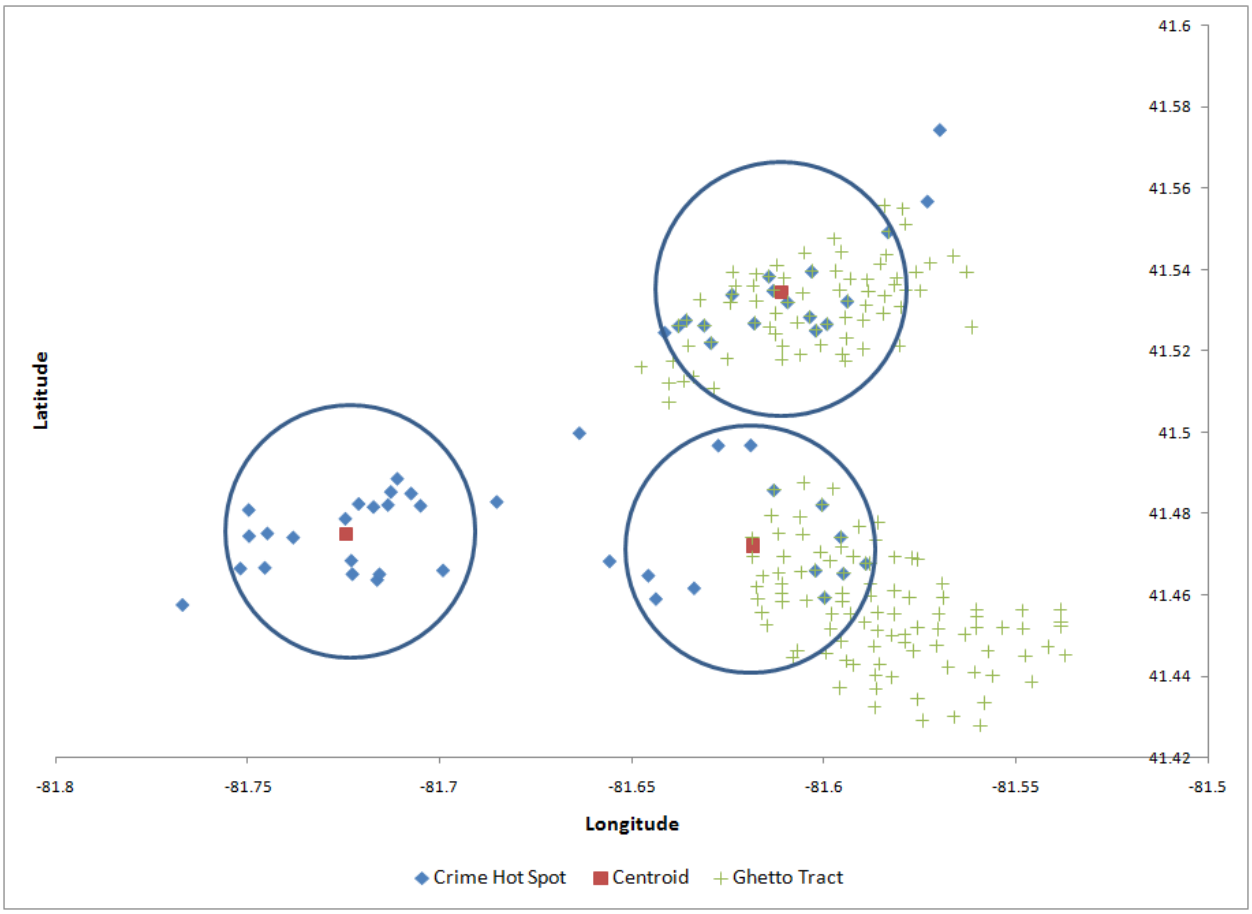


Table A-3. Location of Worksites in Cleveland MSA, 2000

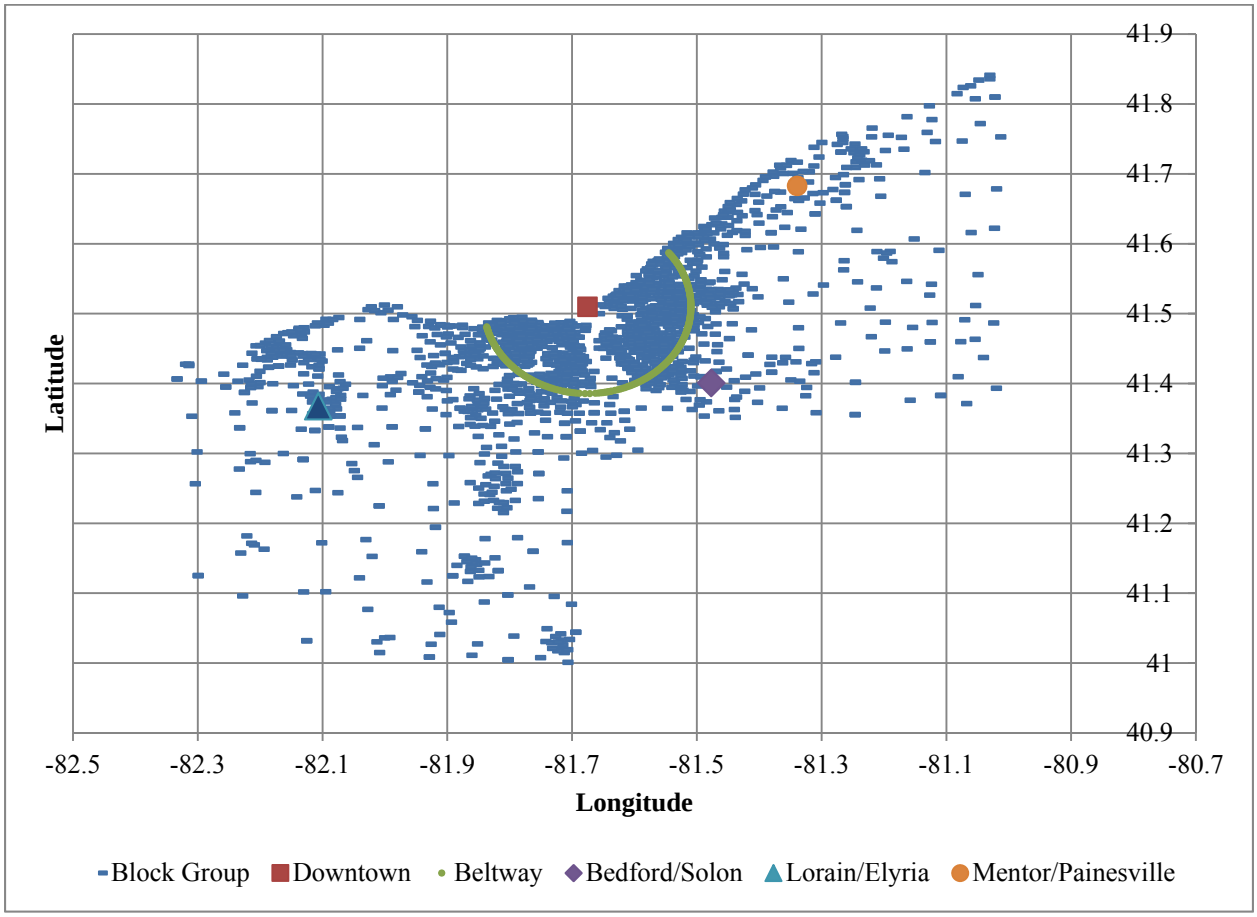


Table A-1: Variable Definitions

Variable	Definition
lbidprice	Natural log of CBG Price per unit of Housing
ser1	Relative Elementary Score Outside Cleveland & E. Cle.
ser2	High School Passing Rate
ser3	Elementary Value Added
ser4	Share Minority Teachers
ser6	Share Non-Black in CBG
ser7	Share Non-Hispanic in CBG
waydist	Weighted Commuting Distance
ytrate	Income Tax Rate
tax1b	School Tax Rate
tax2b	City Tax Rate
tax3b	Tax Break Rate
noas	No A-to-S
nocity	Not a City
lakefront4	Lakefront
dfront	Distance to Lake
snow1	Snowbelt 1
snow2	Snowbelt 2
ghet	Ghetto
nearghet5	Near Ghetto
airport10	Near Airport
dap10	Airport Distance
near2sch	Near Public
d2sch	Distance to Public
near5pri	Near Private
d5pri	Distance3 to Private
nearhaz	Near Hazard
dhaz	Distance to Hazard
pollzone	Smog
dsco	Smog Distance
neighamen	Local Amenities
freeway	Freeway
shopping	Shopping
hospital	Hospital
airport	Small Airport
regpark	Big Park
railroad	Railroad
historic	Historic District
polpop1	City Population
village	Village
township	Township
copol	County Police
vneare	Near Elderly PH
vnearsf	Near Small Fam. PH
vnearlf	Near Big Fam. PH
lowhigh	Crime Lowhigh
highlow	Crime Highlow
highhigh	Crime Highhigh

Table A-1. Variable Definitions, Continued

Variable	Definition
cc1	Crime Hotspot1
cc2	Crime Hotspot2
cc3	Crime Hotspot3
cc4	Crime Hotspot4
co2	Geauga County
co3	Lake County
co4	Lorain County
co5	Medina County
dw2	Worksite 2
dw3	worksite 3
dw4	Worksite 4
dw5	Worksite 5
clesd	Dummy for Cleveland and East Cleveland SDs
cser1	Rel. Elem. Score for Cleveland and East Cleveland

Note: These definitions refer to Tables 2 through 4 in the text.

Table A-2. Descriptive Statistics

Variable	Obs	Mean	Std. Dev.	Min	Max
lbidprice	1665	11.31319	.2658345	10.38021	12.75177
ser1	1222	.308482	.0756794	.001	.554
ser2	1665	.319702	.2040226	.0491107	.7675417
ser3	1665	24.0021	9.416387	1	49.6
ser4	1665	.1329091	.1547839	.001	.6146364
ser6	1665	.8021929	.3226002	.001	1
ser7	1665	.9602518	.0810411	.365347	.998
waydist	1665	13.20461	7.456672	7.266043	39.52358
ytrate	1665	.0001471	.0011567	0	.01
tax1b	1665	.0309153	.0082999	.0172086	.0642917
tax2b	1665	.0497413	.0238843	0	.1032802
tax3b	1665	.0283646	.0160103	0	.0790975
noas	1665	.1339339	.3406836	0	1
nocity	1665	.1393393	.3464043	0	1
lakefront4	1665	.1843844	.3879139	0	1
dfront	1665	.1802176	.4656408	0	1.997163
snow1	1665	.8279953	3.065022	0	19.58192
snow2	1665	10.07429	46.86643	0	383.4517
ghet	1665	.1045045	.3060058	0	1
nearghet5	1665	.1753754	.3804021	0	1
airport10	1665	.3207207	.4668938	0	1
dap10	1665	1.829285	2.902368	0	9.989283
near2sch	1665	.9207207	.2702553	0	1
d2sch	1665	.5876341	.4396461	0	1.996817
near5pri	1665	.8312312	.3746601	0	1
d5pri	1665	1.625187	1.256893	0	4.990643
nearhaz	1665	.6288288	.4832633	0	1
dhaz	1665	.3249864	.3133328	0	.999694
pollzone	1665	.1981982	.3987621	0	1
dsco	1665	2.549414	5.766226	0	19.99535
neighamen	1665	.2900901	.4859112	0	2
freeway	1665	.1357357	.3426106	0	1
shopping	1665	.2834835	.4508244	0	1
hospital	1665	.2504505	.4334026	0	1
airport	1665	.024024	.1531697	0	1
regpark	1665	.1345345	.3413282	0	1
railroad	1665	.2078078	.4058604	0	1
historic	1665	.1561562	.3631124	0	1
polpop1	1665	140936.7	204937.2	0	497903
village	1665	.0252252	.1568556	0	1
township	1665	.0048048	.0691707	0	1
copol	1665	.2114114	.4084321	0	1
vneare	1665	.0474474	.2126578	0	1
vnearsf	1665	.0228228	.1493832	0	1
vnearlf	1665	.0162162	.1263441	0	1
lowhigh	1665	.0252252	.1568556	0	1
highlow	1665	.1291291	.3354436	0	1
highhigh	1665	.1933934	.3950774	0	1

Table A-2. Descriptive Statistics, Continued

Variable	Obs	Mean	Std. Dev.	Min	Max
cc1	1665	.0126126	.1116289	0	1
cc2	1665	.0354354	.1849332	0	1
cc3	1665	.0846847	.2784955	0	1
cc4	1665	.2666667	.4423495	0	1
co2	1665	.0402402	.1965812	0	1
co3	1665	.0858859	.2802796	0	1
co4	1665	.1303303	.3367677	0	1
co5	1665	.0678679	.2515946	0	1
dw2	1665	.4384384	.4963448	0	1
dw3	1665	.0708709	.2566861	0	1
dw4	1665	.0702703	.2556787	0	1
dw5	1665	.0708709	.2566861	0	1
clesd	1665	.2660661	.442032	0	1
cser1	443	.3322935	.1177632	.15	.6465

Appendix B: Estimation Procedures

The regressions reported in this paper were all estimated using Stata (see www.stata.com). As indicated in the text, the first step of my two-step method is estimated in two stages. The first stage includes fixed effects for CBGs and is estimated with the Stata "areg" command.

The coefficients of the fixed effects (plus the constant term) become the dependent variable for the second stage. In the notation in the text, these coefficients can be interpreted as $\ln\{P\}$. The simple linear and quadratic versions of the envelope, which appear in the first two columns of Table 5 in the paper, can be estimated with OLS, that is, with the Stata "reg" command.

The regressions in the other columns use nonlinear least squares (NLLS), which is much more difficult to estimate for two reasons. First, the regressions are complicated enough so that a separate program must be written for each form of the estimating equation. (In Stata terminology, these programs are called "function evaluator programs.") The key regression programs are presented in Appendix C. Second, in the case of a complex estimation such as mine, NLLS does not converge or else finds (often nonsensical) local minima unless it is given initial values that are close to the final values. As a result, the main programming challenge is to develop procedures for determining starting values. The nonlinearity comes first of all from the price elasticities (the μs), so my approach is based on iterating over reasonable values of the μs.

Consider first the regression in the third column of Table 5, which assumes that $\sigma_3 = 1$ for all amenities. In this case, my procedure is:

1. Select a value for μ_i for each amenity S_i;
2. Calculate the implied values of $S_{i1} = [(S_i)^{\lambda_i}-1]/\lambda_i$ and $S_{i2} = [(S_i)^{\lambda_i+1}-1]/(\lambda_i+1)$, where $\lambda_i = (1+\mu_i)/\mu_i$;
3. Estimate an OLS regression using S_{i1} and S_{i2} as variables;

4. Use equation (17) in the text to calculate the values of σ_1 and σ_2 from the coefficients of S_{i1} and S_{i2} ;
5. Iterate over reasonable values of the values for μ_is and select the regression with the lowest sum of squared errors (SSE).
6. Set the starting values for the NLLS regression equal to the values of the μs and standard regression coefficients from this regression.

For the first pass of this procedure, I used a fairly coarse grid of values for the μ_is covering a fairly wide range to find the ranges in which the SSE-minimizing values are likely to fall. (Programs for these coarse-grid loops are not included below.) If one of the SSE-minimizing values falls at the edge of the range I select, I repeat the procedure with a wider range. For the second pass, I use a finer grid in the range established during the first pass. I set the μ_is at the SSE-minimizing values from the first pass and then iterate, one amenity at a time, over a finer grid around the starting value. I retain the SSE-minimizing value of μ_i for each amenity and then move on to the next amenity. I repeat this loop through all the amenities to make sure that the SSE-minimizing value of μ_i for one amenity is not altered by a change in the selected value of μ_i for any of the other amenities. In my experience with this method, the order in which I selected the amenities did not seem to matter, and doing more than two loops through all the amenities also did not change the answer.

The final step is to estimate the NLLS regression using the "nl" command in Stata and a function evaluator program. The estimates in the text all assume that the price elasticity of demand for housing, v, equals -1, which means that the NLLS regression (and all the regressions to determine starting values) use $\ln\{P\}$ as the dependent variable. The standard errors are estimated

using the "vce(hc3)" option in Stata, which is recommended by Davidson and Mackinnon (2004) for models with heteroskedasticity. In most cases, this NLLS command converges rapidly.

Table 5 in the paper has two columns that make use of this estimating procedure. The results in the third column treat Share Non-Black and Share Non-Hispanic as the neighborhood ethnicity variables. As discussed in the text of the paper, this is not quite right conceptually, because some people regard an increase in one of these variables as a decrease in the amenity. The model is set up so that the amenity should always be positive. To address this issue, the model in column four makes three changes. First, it re-defines the amenity to be $(S - S^M)^2$. This amenity has a minimum at S^M, and is always positive. Second, the values of S^M for both Share Non-Black and Share Non-Hispanic are estimated as part of the non-linear procedure. Third, a constraint is imposed at values of S near S^M. This constraint, which is illustrated in Figure B-1, ensures that the envelope does not have a (nonsensical!) large increase at values near S^M. This figure applies to Share Non-Black, but the comparable figure for Share Non-Hispanic is similar. This constraint creates an integrated zone in which the envelope does not change with neighborhood ethnicity. Because the constraint only affects a few observations, 63 for Share Non-Black and 2 for Share Non-Hispanic, it has virtually no impact on any of the estimated coefficients, but, not surprisingly, it lowers the explanatory power of the regression.

I estimate the constrained model in three steps. First, I set S^M equal to 0.25 and 0.5 for Share Non-Black and Share Non-Hispanic, respectively, and obtain starting values using the iterative procedure described above. Second, I estimating a model without the constraint but with the two values of S^M treated as parameters using the starting values from the first step. Third, I use use the estimated coefficients from the second step as starting values for the estimation of the constrained model.

Table 5 also presents specification tests that determine whether each new model adds explanatory power to the simpler model in the previous column. The test in the second column is a standard F-test for the addition of a set of new variables, in this case the squared terms for the amenity variables. The other tests all take the form discussed by Davidson and MacKinnon (2004). The predicted values from the new model are added as a variable to the simpler model and the t-test for this variable provides the desired specification test (using a robust standard error as calculated by the vce(hc3) option in Stata). In the case of the last column, this test is done both ways, and neither this column nor the previous one can be rejected in favor of the other.

Finally, Table 5 also gives the result for tests of the hypothesis that $v = -1$. In the first and second column, this test re-estimates the OLS model as a left-side only Box-Cox model using the Stata "boxcox" command. This command estimates the exponent of the left side Box-Cox variable, which, as shown by equation (17), is $1+v$. Hence the appropriate test is whether the exponent equals zero. This test appears in the Stata output. In the third and fourth column, an equivalent test is implemented by re-doing the nonlinear estimation treating v as a parameter and using the specification for v in equation (16). The starting value for v is set near -1 instead of being found through iteration. The appropriate test is whether the NLLS estimate of v is significantly different from -1 (using standard errors estimated with the vce(hc3) option in Stata).

The logic of these estimations fits neatly into the "partial identification" framework reviewed in Tamer (2010, p. 168).

> The partial identification approach to econometrics views economic models as sets of assumptions, some of which are plausible—e.g. based on economic principles that respect constraints and optimizing behavior—and some of which are esoteric and are needed only to complete a model. These latter assumptions are usually termed functional forms or distributional assumptions. Partial identification calls for analyzing the sensitivity of our inferences on the parameter of interest to these esoteric assumptions….

> For example, it is accepted that (unobserved) heterogeneity plays a key role in empirical microeconomic models. Economic theory is largely silent regarding the choice of the distribution of unobserved heterogeneity, and in many cases, the choice of this distribution is based on folklore, familiarity, and computational grounds. This is especially important in nonlinear models in which mean independence assumptions are not sufficient. In these models, it is important to examine the role played by assumptions made on the heterogeneity distribution.

Tamer goes on to discuss various approaches to this type of situation. One approach that is particularly appropriate here is (p. 169):

> a top-down approach in which a fully parametric model that point identifies the parameter of interest using a set of assumptions is first considered. Each of the unsettled assumptions yields to a different model that identifies a value for the parameter of interest, so this sensitivity analysis approach collects in a set different values of the parameter of interest that correspond to the different models.

Tamer also shows that in many cases, the available information makes it possible to devise formal bounds on parameter estimates that correspond to theoretical or empirical bounds on the assumptions about the "esoteric" parameters. No such bounds are available in my application, however, so the "partial identification" literature only prodes a framework, not a formal estimating procedure.

The key "esoteric" parameter in my case, of course, is σ_3. The estimates in Table 5 are all based on the assumption that $\sigma_3 = 1$. Table 6 describes estimates with alternative assumptions about σ_3. These estimates all use the nonlinear estimating techniques described above. The first panel of Table 5 is based on alternative assumptions for σ_3 for each of the four main amenity variables. The second panel makes various assumptions about the price elasticities of demand, the μs, and estimates σ_3. To keep the programming manageable, the estimations in the second panel make three simplifying assumptions. First, the Elementary variable is entered simply as a quadratic, which is equivalent to setting its value of $\mu = -\infty$. As discussed in the text, other values

for this parameter can be rejected in favor of -∞, and nothing is gained by estimating models in which this parameter is allowed to vary. Second, the value of the S^M parameter is set at 0.25 or 0.5 for Share Non-Black and Share Non-Hispanic, respectively. This simplifies the estimation considerably, but appears to have little impact on the results because these values are very close to the precisely estimated values for this parameter in other estimations. Third, the models are estimated without the "integration" constraint for Share Non-Black and Share Non-Hispanic. As noted earlier, this constraint only affects a few observations and does not appear to have a significant impact on any of the parameter estimates.

Table 7 in the paper presents results for the analog to Rosen's second step. The dependent variable is the relative slope of a household type's bid function, ψ, which is defined by equation (11), and the explanatory variables are determinants of that relative slope, such as income. For the purposes of these regressions, a household "type" is defined as the households in a CBG. The indirect tests regress ψ on median homeowner income in the CBG.

I do not actually observe homeowner income at the CBG level, but I do observe overall median income in the CBG and median homeowner income at the census tract level. With tract data, the ratio of median owner income to overall median income can be closely predicted with a regression on the share of households who are homeowners, the log of overall median income, and an interaction between these two variables. The results are then used to predict median owner income at the CBG level. The regression results, which are based on the 667 tracts in the Cleveland area in 2000 with at least one owner household, are

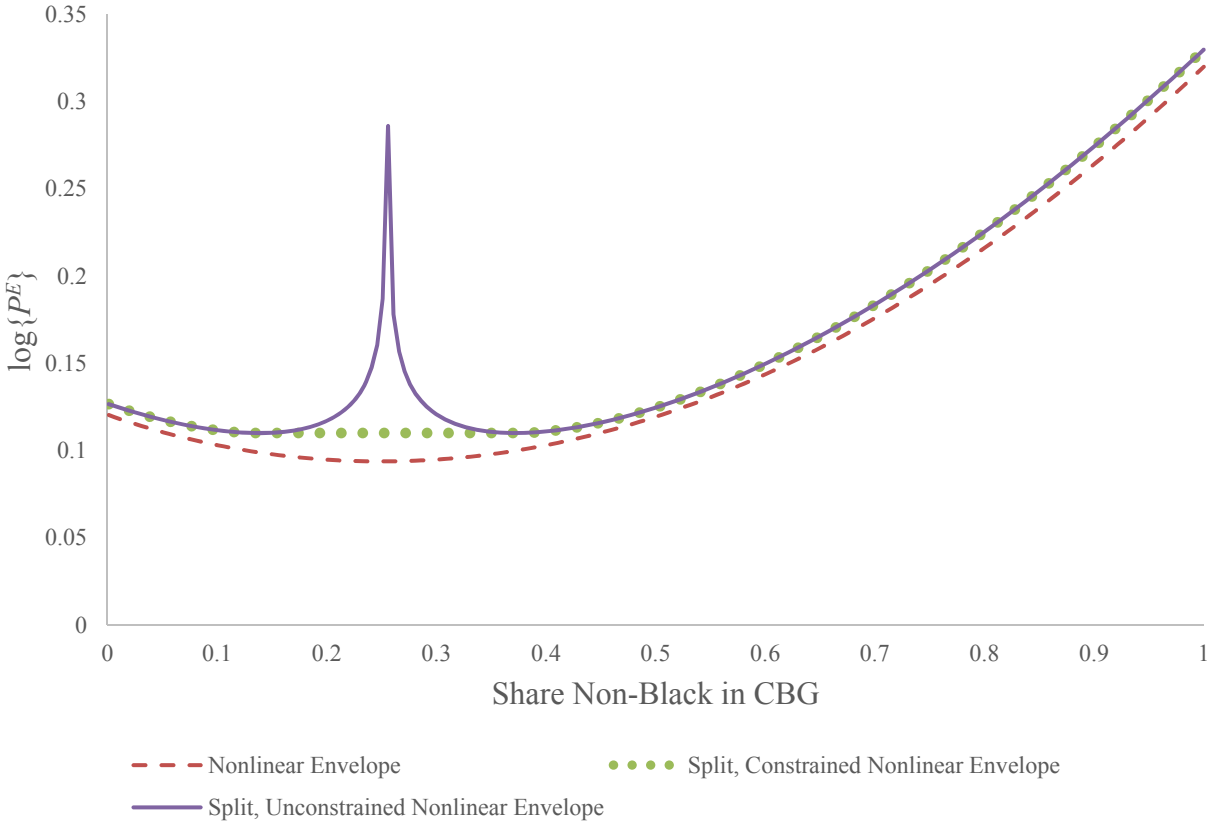
	Coefficient	*Standard Error*	*t Stat*
Intercept	9.5987	0.7919	12.12
Share Owner	-11.6604	1.1599	-10.05
Log(Overall Median Income)	-0.7441	0.0790	-9.42
Interaction	1.0229	0.1100	9.30

The R^2 for this regression os 0.428. These explanatory variables are all observed at the CBG level, so median owner income at the CBG level can be predicted using the CBG values for these variables and the estimated coefficients from the tract-level regression.

The results in Table 7 are based only on observations with a positively sloped envelope or, in the case of the amenity variables, with a value of S greater than S^M. Although the demand concepts apply to observations in which the amenity variables involve S less than S^M, there are not enough observations for the second-step regressions in this case. For Elementary, the regressions are based on the observations in Cleveland with a positively sloped envelope.

Note that the indirect tests incorporate all variables correlated with income, whether they are observed or unobserved, and therefore yield unbiased coefficients. The direct tests are unbiased if unobserved determinants of ψ are not correlated with income. The regressions have an extensive set of ψ determinants, but this type of bias cannot be ruled out.

Figure B-1. Alternative Envelopes for Share Non-Black



Appendix C: Selected Programs

Introduction

This appendix presents selected Stata programs used to implement the procedures in Appendix B. The key variable names are given in Appendix Table A-1. The key amenity variables are ser1 (Elementary), ser2 (High School), ser3 (Value Added), ser4 (Minority Teachers), ser6 (Share Non-Black), and ser7 (Share Non-Hispanic). Other variable names are defined by the programs.

Two key programs presented are:

Program for Column 4 of Table 5 and Table 7

Program for Panel B of Table 6

Program for Column 4 of Table 5 and for Table 7

```
*****************************************
* This program is intended to do the baseline regessions
* with estimates of the mus
* and with sigmas set at start
*
* The ethnicity variables are split and constrained
*
*********************************
*Define variables for envelope
*
generate double sigma31=1
generate double sigma32=1
generate double sigma36=1
generate double sigma37=1
*
generate double mufin1=-1.01
generate double mufin2=-1.01
generate double mufin6=-1.01
generate double mufin7=-1.01
*
generate double ser11=0
generate double ser12=0
generate double cer11=0
generate double cer12=0
generate double der11=0
generate double der12=0
generate double ser21=0
generate double ser22=0
generate double ser61=0
generate double ser62=0
generate double ser71=0
generate double ser72=0
*
generate double ter11=0
generate double ter12=0
generate double ter21=0
generate double ter22=0
generate double ter61=0
generate double ter62=0
generate double ter71=0
generate double ter72=0
*
scalar sstar6=.25
```

```
scalar sstar7=.5
replace ser11=(1-clesd)*ser1
replace ser12=(1-clesd)*ser1^2
replace ser21=log(ser2)
replace ser22=ser2-1
replace ser61=log(ser6)
replace ser62=ser6-1
replace ser71=log(ser7)
replace ser72=ser7-1
replace cer11=clesd*ser1
replace cer12=clesd*ser1^2
*
**********
*Matrices and variables for program
*
global matrix init
global matrix coco
global matrix fico
global scalar ec11
global scalar ec12
global scalar cc11
global scalar cc12
global scalar ec21
global scalar ec22
global scalar ec61
global scalar ec62
global scalar ec71
global scalar ec72
global scalar mu1
global scalar mu2
global scalar mu6
global scalar mu7
global scalar star6
global scalar star7
*
**********
*Nonlinear regression programs
*
*set trace on
*
program nlfullpriceA
syntax varlist(min=64 max=64) [if] , at(name)
local lbidprice : word 1 of `varlist'
local ser1 : word 2 of `varlist'
local ser2 : word 3 of `varlist'
```

```
local ser3 : word 4 of `varlist'
local ser4 : word 5 of `varlist'
local pser6 : word 6 of `varlist'
local pser7 : word 7 of `varlist'
local waydist : word 8 of `varlist'
local ytrate : word 9 of `varlist'
local tax1b : word 10 of `varlist'
local tax2b : word 11 of `varlist'
local tax3b : word 12 of `varlist'
local noas : word 13 of `varlist'
local nocity : word 14 of `varlist'
local lakefront4 : word 15 of `varlist'
local dfront : word 16 of `varlist'
local snow1 : word 17 of `varlist'
local snow2 : word 18 of `varlist'
local ghet : word 19 of `varlist'
local nearghet5 : word 20 of `varlist'
local airport10 : word 21 of `varlist'
local dap10 : word 22 of `varlist'
local near2sch : word 23 of `varlist'
local d2sch : word 24 of `varlist'
local near5pri : word 25 of `varlist'
local d5pri : word 26 of `varlist'
local nearhaz : word 27 of `varlist'
local dhaz : word 28 of `varlist'
local pollzone : word 29 of `varlist'
local dsco : word 30 of `varlist'
local neighamen : word 31 of `varlist'
local freeway : word 32 of `varlist'
local shopping : word 33 of `varlist'
local hospital : word 34 of `varlist'
local airport : word 35 of `varlist'
local regpark : word 36 of `varlist'
local railroad : word 37 of `varlist'
local historic : word 38 of `varlist'
local polpop1 : word 39 of `varlist'
local polpop2 : word 40 of `varlist'
local polpop3 : word 41 of `varlist'
local polpop4 : word 42 of `varlist'
local village : word 43 of `varlist'
local township : word 44 of `varlist'
local copol : word 45 of `varlist'
local vneare : word 46 of `varlist'
local vnearsf : word 47 of `varlist'
local vnearlf : word 48 of `varlist'
```

```
local lowhigh : word 49 of `varlist'
local highlow : word 50 of `varlist'
local highhigh : word 51 of `varlist'
local cc1 : word 52 of `varlist'
local cc2 : word 53 of `varlist'
local cc3 : word 54 of `varlist'
local cc4 : word 55 of `varlist'
local co2 : word 56 of `varlist'
local co3 : word 57 of `varlist'
local co4 : word 58 of `varlist'
local co5 : word 59 of `varlist'
local dw2 : word 60 of `varlist'
local dw3 : word 61 of `varlist'
local dw4 : word 62 of `varlist'
local dw5 : word 63 of `varlist'
local clesd : word 64 of `varlist'
#delimit ;
tempname sn11 sn12 sn21 sn22 sn61 sn62 sn71 sn72 sn31 sn32 sn41 sn42 sc1 sc2 alpha beta1
beta2 beta3  ;
tempname bx1 bx2 bx3 bx4 bx5 bx6 bx7 bx8 bx9 bx10 bx11 bx12 bx13 bx14 bx15 bx16 bx17
bx18 bx19 bx20 bx21 bx22 bx23 bx24 bx25 bx26 bx27 bx28 bx29 bx30 bx31 bx32 bx33 bx34
bx35 bx36 bx37 bx38 bx39 bx40 bx41 bx42 bx43 bx44 bx45 bx46 bx47 bx48 bx49 bx50 bx51 c0
star6 star7 ;
tempname mu1 mu2 mu6 mu7 cle10 cle11 cle12 ;
#delimit cr
scalar `sn11' = `at'[1, 1]
scalar `sn12' = `at'[1, 2]
scalar `cle11' = `at'[1, 3]
scalar `cle12' = `at'[1, 4]
scalar `sn21' = `at'[1, 5]
scalar `sn22' = `at'[1, 6]
scalar `sn61' = `at'[1, 7]
scalar `sn62' = `at'[1, 8]
scalar `sn71' = `at'[1, 9]
scalar `sn72' = `at'[1, 10]
scalar `sn31' = `at'[1, 11]
scalar `sn32' = `at'[1, 12]
scalar `sn41' = `at'[1, 13]
scalar `sn42' = `at'[1, 14]
scalar `sc1' = `at'[1, 15]
scalar `sc2' = `at'[1, 16]
scalar `alpha' = `at'[1, 17]
scalar `beta1' = `at'[1, 18]
scalar `beta2' = `at'[1, 19]
scalar `beta3' = `at'[1, 20]
```

```
scalar `bx1' = `at'[1, 21]
scalar `bx2' = `at'[1, 22]
scalar `bx3' = `at'[1, 23]
scalar `bx4' = `at'[1, 24]
scalar `bx5' = `at'[1, 25]
scalar `bx6' = `at'[1, 26]
scalar `bx7' = `at'[1, 27]
scalar `bx8' = `at'[1, 28]
scalar `bx9' = `at'[1, 29]
scalar `bx10' = `at'[1, 30]
scalar `bx11' = `at'[1, 31]
scalar `bx12' = `at'[1, 32]
scalar `bx13' = `at'[1, 33]
scalar `bx14' = `at'[1, 34]
scalar `bx15' = `at'[1, 35]
scalar `bx16' = `at'[1, 36]
scalar `bx17' = `at'[1, 37]
scalar `bx18' = `at'[1, 38]
scalar `bx19' = `at'[1, 39]
scalar `bx20' = `at'[1, 40]
scalar `bx21' = `at'[1, 41]
scalar `bx22' = `at'[1, 42]
scalar `bx23' = `at'[1, 43]
scalar `bx24' = `at'[1, 44]
scalar `bx25' = `at'[1, 45]
scalar `bx26' = `at'[1, 46]
scalar `bx27' = `at'[1, 47]
scalar `bx28' = `at'[1, 48]
scalar `bx29' = `at'[1, 49]
scalar `bx30' = `at'[1, 50]
scalar `bx31' = `at'[1, 51]
scalar `bx32' = `at'[1, 52]
scalar `bx33' = `at'[1, 53]
scalar `bx34' = `at'[1, 54]
scalar `bx35' = `at'[1, 55]
scalar `bx36' = `at'[1, 56]
scalar `bx37' = `at'[1, 57]
scalar `bx38' = `at'[1, 58]
scalar `bx39' = `at'[1, 59]
scalar `bx40' = `at'[1, 60]
scalar `bx41' = `at'[1, 61]
scalar `bx42' = `at'[1, 62]
scalar `bx43' = `at'[1, 63]
scalar `bx44' = `at'[1, 64]
scalar `bx45' = `at'[1, 65]
```

```
scalar `bx46' = `at'[1, 66]
scalar `bx47' = `at'[1, 67]
scalar `bx48' = `at'[1, 68]
scalar `bx49' = `at'[1, 69]
scalar `bx50' = `at'[1, 70]
scalar `bx51' = `at'[1, 71]
scalar `cle10' = `at'[1, 72]
scalar `c0' = `at'[1, 73]
scalar `mu2' = `at'[1, 74]
scalar `mu6' = `at'[1, 75]
scalar `mu7' = `at'[1, 76]
scalar `star6' = `at'[1, 77]
scalar `star7' = `at'[1, 78]
*tempvar lcaprate
tempvar aser6 aser7 integ6 integ7
tempvar bids1 bids2 bids6 bids7
tempvar bids bida bidu bidx phat
tempname sig31 sig32 sig36 sig37
*here
scalar `sig31'=1
scalar `sig32'=1
scalar `sig36'=1
scalar `sig37'=1
tempvar ex11 ex12 ex21 ex22 ex61 ex62 ex71 ex72
generate double `aser6'=(`pser6'-`star6')^2 `if'
generate double `aser7'=(`pser7'-`star7')^2 `if'
generate double `integ6'=`aser6'^(1/`sig36')<=`sn61' `if'
generate double `integ7'=`aser7'^(1/`sig37')<=`sn71' `if'
generate double `ex11'=1 `if'
*generate double `ex11'=(1+`mu1')/`mu1' `if'
generate double `ex12'=`ex11'+1/`sig31' `if'
generate double `ex21'=(1+`mu2')/`mu2' `if'
generate double `ex22'=`ex21'+1/`sig32' `if'
generate double `ex61'=(1+`mu6')/`mu6' `if'
generate double `ex62'=`ex61'+1/`sig36' `if'
generate double `ex71'=(1+`mu7')/`mu7' `if'
generate double `ex72'=`ex71'+1/`sig37' `if'
#delimit ;
*generate double `lcaprate'=log(1+`beta1'*`tax1b'+`beta2'*`tax2b'
+`beta3'*`tax3b') `if' ;
generate double `bids1'=(1-`clesd')*((-1)*(`sn11'/`sn12')*(`ser1'^`ex11'-
1)/`ex11'+(1/`sn12')*(`ser1'^`ex12'-1)/`ex12')
+`clesd'*(`cle10'-((`cle11')/(`cle12'))*(`ser1'^`ex11'-1)/`ex11'+(1/(`cle12'))*(`ser1'^`ex12'-
1)/`ex12') `if' ;
*generate double `bids1'=-(`sn11'/`sn12')*(`ser1'^`ex11'-1)/`ex11'
```

```
+(1/`sn12')*(`ser1'^`ex12'-1)/`ex12'
+`clesd'*(`cle10'-(`cle11'/`cle12')*(`ser1'^`ex11'-1)/`ex11'
+(1/`cle12')*(`ser1'^`ex12'-1)/`ex12') `if' ;
generate double `bids2'=-(`sn21'/`sn22')*(`ser2'^`ex21'-1)/`ex21'
+(1/`sn22')*(`ser2'^`ex22'-1)/`ex22' `if' ;
generate double `bids6'=-(`sn61'/`sn62')*(`aser6'^`ex61'-1)/`ex61'
+(1/`sn62')*(`aser6'^`ex62'-1)/`ex62' `if' ;
*replace `bids6'=`bids6'*(1-`integ6')
+(-(`sn61'/`sn62')*((`sn61'^`sig36')^`ex61'-1)/`ex61'
+(1/`sn62')*((`sn61'^`sig36')^`ex62'-1)/`ex62')*`integ6' `if' ;
generate double `bids7'=-(`sn71'/`sn72')*(`aser7'^`ex71'-1)/`ex71'
+(1/`sn72')*(`aser7'^`ex72'-1)/`ex72' `if' ;
*replace `bids7'=`bids7'*(1-`integ7')
+(-(`sn71'/`sn72')*((`sn71'^`sig37')^`ex71'-1)/`ex71'
+(1/`sn72')*((`sn71'^`sig37')^`ex72'-1)/`ex72')*`integ7' `if' ;
generate double `bids'=`bids1'+`bids2'
+`bids6'+`bids7' `if' ;
generate double `bida'=`sn31'*`ser3'+`sn32'*(`ser3'^2)+ `sn41'*`ser4'+`sn42'*(`ser4'^2)  `if' ;
generate double `bidu' = `sc1'*`waydist'
+`sc2'*((`waydist')^2)+`bx48'*`dw2'+`bx49'*`dw3'
+`bx50'*`dw4'+`bx51'*`dw5' `if' ;
generate double `bidx' = `alpha'*`ytrate'
+`beta1'*`tax1b'+`beta2'*`tax2b'+`beta3'*`tax3b'
+`bx1'*`noas'+`bx2'*`nocity'+`bx3'*`lakefront4'+`bx4'*`dfront'
+`bx5'*`snow1'+`bx6'*`snow2'+`bx7'*`ghet'+`bx8'*`nearghet5'
+`bx9'*`airport10'+`bx10'*`dap10'+`bx11'*`near2sch'
+`bx12'*`d2sch'+`bx13'*`near5pri'+`bx14'*`d5pri'
+`bx15'*`nearhaz'+`bx16'*`dhaz'
+`bx17'*`pollzone'+`bx18'*`dsco'
+`bx19'*`neighamen'+`bx20'*`freeway'
+`bx21'*`shopping'+`bx22'*`hospital'+`bx23'*`airport'
+`bx24'*`regpark'+`bx25'*`railroad'+`bx26'*`historic'
+`bx27'*`polpop1'+`bx28'*`polpop2'+`bx29'*`polpop3'+`bx30'*`polpop4'
+`bx31'*`village'+`bx32'*`township'+`bx33'*`copol'+`bx34'*`vneare'
+`bx35'*`vnearsf'+`bx36'*`vnearlf'+`bx37'*`lowhigh'+`bx38'*`highlow'
+`bx39'*`highhigh'+`bx40'*`cc1'+`bx41'*`cc2'+`bx42'*`cc3'
+`bx43'*`cc4'+`bx44'*`co2'+`bx45'*`co3'+`bx46'*`co4'+`bx47'*`co5' `if' ;
generate double `phat' =`c0'+`bids'+`bida'+`bidu'+`bidx' `if' ;
*replace `lbidprice' = `phat' - `lcaprate' `if' ;
replace `lbidprice' = `phat' `if' ;
#delimit cr
end
*******************************************
*
*
```

```
program nlfullprice
syntax varlist(min=64 max=64) [if] , at(name)
local lbidprice : word 1 of `varlist'
local ser1 : word 2 of `varlist'
local ser2 : word 3 of `varlist'
local ser3 : word 4 of `varlist'
local ser4 : word 5 of `varlist'
local pser6 : word 6 of `varlist'
local pser7 : word 7 of `varlist'
local waydist : word 8 of `varlist'
local ytrate : word 9 of `varlist'
local tax1b : word 10 of `varlist'
local tax2b : word 11 of `varlist'
local tax3b : word 12 of `varlist'
local noas : word 13 of `varlist'
local nocity : word 14 of `varlist'
local lakefront4 : word 15 of `varlist'
local dfront : word 16 of `varlist'
local snow1 : word 17 of `varlist'
local snow2 : word 18 of `varlist'
local ghet : word 19 of `varlist'
local nearghet5 : word 20 of `varlist'
local airport10 : word 21 of `varlist'
local dap10 : word 22 of `varlist'
local near2sch : word 23 of `varlist'
local d2sch : word 24 of `varlist'
local near5pri : word 25 of `varlist'
local d5pri : word 26 of `varlist'
local nearhaz : word 27 of `varlist'
local dhaz : word 28 of `varlist'
local pollzone : word 29 of `varlist'
local dsco : word 30 of `varlist'
local neighamen : word 31 of `varlist'
local freeway : word 32 of `varlist'
local shopping : word 33 of `varlist'
local hospital : word 34 of `varlist'
local airport : word 35 of `varlist'
local regpark : word 36 of `varlist'
local railroad : word 37 of `varlist'
local historic : word 38 of `varlist'
local polpop1 : word 39 of `varlist'
local polpop2 : word 40 of `varlist'
local polpop3 : word 41 of `varlist'
local polpop4 : word 42 of `varlist'
local village : word 43 of `varlist'
```

```
local township : word 44 of `varlist'
local copol : word 45 of `varlist'
local vneare : word 46 of `varlist'
local vnearsf : word 47 of `varlist'
local vnearlf : word 48 of `varlist'
local lowhigh : word 49 of `varlist'
local highlow : word 50 of `varlist'
local highhigh : word 51 of `varlist'
local cc1 : word 52 of `varlist'
local cc2 : word 53 of `varlist'
local cc3 : word 54 of `varlist'
local cc4 : word 55 of `varlist'
local co2 : word 56 of `varlist'
local co3 : word 57 of `varlist'
local co4 : word 58 of `varlist'
local co5 : word 59 of `varlist'
local dw2 : word 60 of `varlist'
local dw3 : word 61 of `varlist'
local dw4 : word 62 of `varlist'
local dw5 : word 63 of `varlist'
local clesd : word 64 of `varlist'
#delimit ;
tempname sn11 sn12 sn21 sn22 sn61 sn62 sn71 sn72 sn31 sn32 sn41 sn42 sc1 sc2 alpha beta1
beta2 beta3 cle11 cle12 ;
tempname bx1 bx2 bx3 bx4 bx5 bx6 bx7 bx8 bx9 bx10 bx11 bx12 bx13 bx14 bx15 bx16 bx17
bx18 bx19 bx20 bx21 bx22 bx23 bx24 bx25 bx26 bx27 bx28 bx29 bx30 bx31 bx32 bx33 bx34
bx35 bx36 bx37 bx38 bx39 bx40 bx41 bx42 bx43 bx44 bx45 bx46 bx47 bx48 bx49 bx50 bx51 c0
star6 star7 ;
tempname mu1 mu2 mu6 mu7 cle10 cle11 cle12 ecle10 sigma3A sigma3B ;
#delimit cr
scalar `sn11' = `at'[1, 1]
scalar `sn12' = `at'[1, 2]
scalar `cle11' = `at'[1, 3]
scalar `cle12' = `at'[1, 4]
scalar `sn21' = `at'[1, 5]
scalar `sn22' = `at'[1, 6]
scalar `sn61' = `at'[1, 7]
scalar `sn62' = `at'[1, 8]
scalar `sn71' = `at'[1, 9]
scalar `sn72' = `at'[1, 10]
scalar `sn31' = `at'[1, 11]
scalar `sn32' = `at'[1, 12]
scalar `sn41' = `at'[1, 13]
scalar `sn42' = `at'[1, 14]
scalar `sc1' = `at'[1, 15]
```

```
scalar `sc2' = `at'[1, 16]
scalar `alpha' = `at'[1, 17]
scalar `beta1' = `at'[1, 18]
scalar `beta2' = `at'[1, 19]
scalar `beta3' = `at'[1, 20]
scalar `bx1' = `at'[1, 21]
scalar `bx2' = `at'[1, 22]
scalar `bx3' = `at'[1, 23]
scalar `bx4' = `at'[1, 24]
scalar `bx5' = `at'[1, 25]
scalar `bx6' = `at'[1, 26]
scalar `bx7' = `at'[1, 27]
scalar `bx8' = `at'[1, 28]
scalar `bx9' = `at'[1, 29]
scalar `bx10' = `at'[1, 30]
scalar `bx11' = `at'[1, 31]
scalar `bx12' = `at'[1, 32]
scalar `bx13' = `at'[1, 33]
scalar `bx14' = `at'[1, 34]
scalar `bx15' = `at'[1, 35]
scalar `bx16' = `at'[1, 36]
scalar `bx17' = `at'[1, 37]
scalar `bx18' = `at'[1, 38]
scalar `bx19' = `at'[1, 39]
scalar `bx20' = `at'[1, 40]
scalar `bx21' = `at'[1, 41]
scalar `bx22' = `at'[1, 42]
scalar `bx23' = `at'[1, 43]
scalar `bx24' = `at'[1, 44]
scalar `bx25' = `at'[1, 45]
scalar `bx26' = `at'[1, 46]
scalar `bx27' = `at'[1, 47]
scalar `bx28' = `at'[1, 48]
scalar `bx29' = `at'[1, 49]
scalar `bx30' = `at'[1, 50]
scalar `bx31' = `at'[1, 51]
scalar `bx32' = `at'[1, 52]
scalar `bx33' = `at'[1, 53]
scalar `bx34' = `at'[1, 54]
scalar `bx35' = `at'[1, 55]
scalar `bx36' = `at'[1, 56]
scalar `bx37' = `at'[1, 57]
scalar `bx38' = `at'[1, 58]
scalar `bx39' = `at'[1, 59]
scalar `bx40' = `at'[1, 60]
```

```
scalar `bx41' = `at'[1, 61]
scalar `bx42' = `at'[1, 62]
scalar `bx43' = `at'[1, 63]
scalar `bx44' = `at'[1, 64]
scalar `bx45' = `at'[1, 65]
scalar `bx46' = `at'[1, 66]
scalar `bx47' = `at'[1, 67]
scalar `bx48' = `at'[1, 68]
scalar `bx49' = `at'[1, 69]
scalar `bx50' = `at'[1, 70]
scalar `bx51' = `at'[1, 71]
scalar `cle10' = `at'[1, 72]
scalar `c0' = `at'[1, 73]
scalar `mu2' = `at'[1, 74]
scalar `mu6' = `at'[1, 75]
scalar `mu7' = `at'[1, 76]
scalar `star6' = `at'[1, 77]
scalar `star7' = `at'[1, 78]
*scalar `sigma3A' = `at'[1, 80]
*scalar `sigma3B' = `at'[1, 81]
*tempvar lcaprate
tempvar aser6 aser7 integ6 integ7
tempvar bids1 bids2 bids6 bids7
tempvar bids bida bidu bidx phat
tempname sig31 sig32 sig36 sig37
*here
scalar `sig31'=1
scalar `sig32'=1
scalar `sig36'=1
scalar `sig37'=1
tempvar ex11 ex12 ex21 ex22 ex61 ex62 ex71 ex72
generate double `aser6'=(`pser6'-`star6')^2 `if'
generate double `aser7'=(`pser7'-`star7')^2 `if'
generate double `integ6'=`aser6'^(1/`sig36')<=`sn61' `if'
generate double `integ7'=`aser7'^(1/`sig37')<=`sn71' `if'
*generate double `ex11'=(1+`mu1')/`mu1' `if'
generate double `ex11'=1 `if'
generate double `ex12'=`ex11'+1/`sig31' `if'
generate double `ex21'=(1+`mu2')/`mu2' `if'
generate double `ex22'=`ex21'+1/`sig32' `if'
generate double `ex61'=(1+`mu6')/`mu6' `if'
generate double `ex62'=`ex61'+1/`sig36' `if'
generate double `ex71'=(1+`mu7')/`mu7' `if'
generate double `ex72'=`ex71'+1/`sig37' `if'
#delimit ;
```

```
*generate double `lcaprate'=log(1+`beta1'*`tax1b'+`beta2'*`tax2b'
+`beta3'*`tax3b') `if' ;
generate double `bids1'=(1-`clesd')*((-1)*(`sn11'/`sn12')*(`ser1'^`ex11'-
1)/`ex11'+(1/`sn12')*(`ser1'^`ex12'-1)/`ex12')
+`clesd'*(`cle10'-((`cle11')/(`cle12'))*(`ser1'^`ex11'-1)/`ex11'+(1/(`cle12'))*(`ser1'^`ex12'-
1)/`ex12') `if' ;
*generate double `bids1'=-(`sn11'/`sn12')*(`ser1'^`ex11'-1)/`ex11'
+(1/`sn12')*(`ser1'^`ex12'-1)/`ex12'
+`clesd'*(`cle10'-(`cle11'/`cle12')*(`ser1'^`ex11'-1)/`ex11'
+(1/`cle12')*(`ser1'^`ex12'-1)/`ex12') `if' ;
generate double `bids2'=-(`sn21'/`sn22')*(`ser2'^`ex21'-1)/`ex21'
+(1/`sn22')*(`ser2'^`ex22'-1)/`ex22' `if' ;
generate double `bids6'=-(`sn61'/`sn62')*(`aser6'^`ex61'-1)/`ex61'
+(1/`sn62')*(`aser6'^`ex62'-1)/`ex62' `if' ;
replace `bids6'=`bids6'*(1-`integ6')
+(-(`sn61'/`sn62')*((`sn61'^`sig36')^`ex61'-1)/`ex61'
+(1/`sn62')*((`sn61'^`sig36')^`ex62'-1)/`ex62')*`integ6' `if' ;
generate double `bids7'=-(`sn71'/`sn72')*(`aser7'^`ex71'-1)/`ex71'
+(1/`sn72')*(`aser7'^`ex72'-1)/`ex72' `if' ;
replace `bids7'=`bids7'*(1-`integ7')
+(-(`sn71'/`sn72')*((`sn71'^`sig37')^`ex71'-1)/`ex71'
+(1/`sn72')*((`sn71'^`sig37')^`ex72'-1)/`ex72')*`integ7' `if' ;
generate double `bids'=`bids1'+`bids2'
+`bids6'+`bids7' `if' ;
generate double `bida'=`sn31'*`ser3'+`sn32'*(`ser3'^2)+ `sn41'*`ser4'+`sn42'*(`ser4'^2)  `if' ;
generate double `bidu' = `sc1'*`waydist'
+`sc2'*((`waydist')^2)+`bx48'*`dw2'+`bx49'*`dw3'
+`bx50'*`dw4'+`bx51'*`dw5' `if' ;
generate double `bidx' = `alpha'*`ytrate'
+`beta1'*`tax1b'+`beta2'*`tax2b'+`beta3'*`tax3b'
+`bx1'*`noas'+`bx2'*`nocity'+`bx3'*`lakefront4'+`bx4'*`dfront'
+`bx5'*`snow1'+`bx6'*`snow2'+`bx7'*`ghet'+`bx8'*`nearghet5'
+`bx9'*`airport10'+`bx10'*`dap10'+`bx11'*`near2sch'
+`bx12'*`d2sch'+`bx13'*`near5pri'+`bx14'*`d5pri'
+`bx15'*`nearhaz'+`bx16'*`dhaz'
+`bx17'*`pollzone'+`bx18'*`dsco'
+`bx19'*`neighamen'+`bx20'*`freeway'
+`bx21'*`shopping'+`bx22'*`hospital'+`bx23'*`airport'
+`bx24'*`regpark'+`bx25'*`railroad'+`bx26'*`historic'
+`bx27'*`polpop1'+`bx28'*`polpop2'+`bx29'*`polpop3'+`bx30'*`polpop4'
+`bx31'*`village'+`bx32'*`township'+`bx33'*`copol'+`bx34'*`vneare'
+`bx35'*`vnearsf'+`bx36'*`vnearlf'+`bx37'*`lowhigh'+`bx38'*`highlow'
+`bx39'*`highhigh'+`bx40'*`cc1'+`bx41'*`cc2'+`bx42'*`cc3'
+`bx43'*`cc4'+`bx44'*`co2'+`bx45'*`co3'+`bx46'*`co4'+`bx47'*`co5' `if' ;
generate double `phat' =`c0'+`bids'+`bida'+`bidu'+`bidx' `if' ;
```

```
*replace `lbidprice' = `phat' - `lcaprate' `if' ;
replace `lbidprice' = `phat' `if' ;
#delimit cr
end
*****************************************
*
*Run baseline regression
*
generate double resid0=0
generate double resid1=0
generate double mu=0
generate double mua=0
generate double mub=0
#delimit ;
reg lbidprice ser11 ser12 ser21 ser22 ser61 ser62 ser71 ser72  ser3 sersq3 ser4 sersq4 waydist
waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet nearghet5
airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen freeway
shopping hospital airport regpark railroad historic polpop1-polpop4 village township copol vneare
vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11 cer12 , vce(hc3)
;
#delimit cr
replace resid0=e(rss)
*
***********************************
*Now loop through special cases for each service
*
generate madem=6
generate sigsig=1
generate finsig=1
generate restart=resid0
*
*
*Outer loop
*
forvalues i=1/2 {
*
**************
*I. Service 1
*
* Interations Dropped
*
**************
*II. Service 2
*
*II.X. Interations
```

```
*
quietly replace mu=-1.305
*
forvalues i=1/700 {
quietly replace mua=(1+mu)/mu
quietly replace mub=mua+(1/sigma32)
quietly replace ter21=(ser2^mua-1)/mua
quietly replace ter22=(ser2^mub-1)/mub
#delimit ;
quietly reg lbidprice ser11 ser12 ter21 ter22 ser61 ser62 ser71 ser72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11
cer12;
#delimit cr
quietly replace resid1=e(rss)
quietly replace mufin2=mu if((resid1<resid0)&(e(N)==1665))
quietly replace ser21=ter21 if((resid1<resid0)&(e(N)==1665))
quietly replace ser22=ter22 if((resid1<resid0)&(e(N)==1665))
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
quietly replace mu=mu+.001
}
*
*II.B. Mu = -1
*
quietly replace ter21=log(ser2)
quietly replace ter22=(ser2^(1/sigma32)-1)/(1/sigma32)
#delimit ;
quietly reg lbidprice ser11 ser12 ter21 ter22 ser61 ser62 ser71 ser72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11
cer12;
#delimit cr
quietly replace resid1=e(rss)
quietly replace mufin2=-1.01 if((resid1<resid0)&(e(N)==1665))
quietly replace ser21=ter21 if((resid1<resid0)&(e(N)==1665))
quietly replace ser22=ter22 if((resid1<resid0)&(e(N)==1665))
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
*
*II.C. Mu = -2
*
quietly replace ter21=2*(ser2^(1/2)-1)
```

```
#delimit ;
quietly replace ter22=(ser2^(1/2+1/sigma32)-1)/(1/2+1/sigma32) ;
quietly reg lbidprice ser11 ser12 ter21 ter22 ser61 ser62 ser71 ser72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11
cer12;
#delimit cr
quietly replace resid1=e(rss)
quietly replace mufin2=-2.0 if((resid1<resid0)&(e(N)==1665))
quietly replace ser21=ter21 if((resid1<resid0)&(e(N)==1665))
quietly replace ser22=ter22 if((resid1<resid0)&(e(N)==1665))
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
*
***************
*III. Service 6
*
*
*III.X. Interations
*
quietly replace mu=-1.105
*
forvalues i=1/800 {
quietly replace mua=(1+mu)/mu
quietly replace mub=mua+(1/sigma36)
quietly replace ter61=(ser6^mua-1)/mua
quietly replace ter62=(ser6^mub-1)/mub
#delimit ;
quietly reg lbidprice ser11 ser12 ser21 ser22 ter61 ter62 ser71 ser72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11
cer12;
#delimit cr
quietly replace resid1=e(rss)
quietly replace mufin6=mu if((resid1<resid0)&(e(N)==1665))
quietly replace ser61=ter61 if((resid1<resid0)&(e(N)==1665))
quietly replace ser62=ter62 if((resid1<resid0)&(e(N)==1665))
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
quietly replace mu=mu+.001
}
*
*III.A. Mu = -0.5
```

```
*
quietly replace ter61=ser6^(-1)
#delimit ;
quietly replace ter62=
(ser6^((1-sigma36)/sigma36)-1)/((1-sigma36)/sigma36) if(sigma36!=1) ;
#delimit cr
quietly replace ter62=log(ser6) if(sigma36==1)
*
#delimit ;
quietly reg lbidprice ser11 ser12 ser21 ser22 ter61 ter62 ser71 ser72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11
cer12;
#delimit cr
quietly replace resid1=e(rss)
quietly replace mufin6=-.501 if((resid1<resid0)&(e(N)==1665))
quietly replace ser61=ter61 if((resid1<resid0)&(e(N)==1665))
quietly replace ser62=ter62 if((resid1<resid0)&(e(N)==1665))
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
*
*III.B. Mu = -1
*
quietly replace ter61=log(ser6)
quietly replace ter62=(ser6^(1/sigma36)-1)/(1/sigma36)
#delimit ;
quietly reg lbidprice ser11 ser12 ser21 ser22 ter61 ter62 ser71 ser72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11
cer12;
#delimit cr
quietly replace resid1=e(rss)
quietly replace mufin6=-1.01 if((resid1<resid0)&(e(N)==1665))
quietly replace ser61=ter61 if((resid1<resid0)&(e(N)==1665))
quietly replace ser62=ter62 if((resid1<resid0)&(e(N)==1665))
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
*
**********
*IV. Service 7
*
*IV.X. Interations
*
```

```
quietly replace mu=-1.105
*
forvalues i=1/800 {
quietly replace mua=(1+mu)/mu
quietly replace mub=mua+(1/sigma37)
quietly replace ter71=(ser7^mua-1)/mua
quietly replace ter72=(ser7^mub-1)/mub
#delimit ;
quietly reg lbidprice ser11 ser12 ser21 ser22 ser61 ser62 ter71 ter72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11
cer12;
#delimit cr
quietly replace resid1=e(rss)
quietly replace mufin7=mu if((resid1<resid0)&(e(N)==1665))
quietly replace ser71=ter71 if((resid1<resid0)&(e(N)==1665))
quietly replace ser72=ter72 if((resid1<resid0)&(e(N)==1665))
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
quietly replace mu=mu+.001
}
*
*IV.A. Mu = -0.5
*
quietly replace ter71=ser7^(-1)
#delimit ;
quietly replace ter72=
(ser7^((1-sigma37)/sigma37)-1)/((1-sigma37)/sigma37) if(sigma37!=1) ;
#delimit cr
quietly replace ter72=log(ser7) if(sigma37==1)
*
#delimit ;
quietly reg lbidprice ser11 ser12 ser21 ser22 ser61 ser62 ter71 ter72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11
cer12;
#delimit cr
quietly replace resid1=e(rss)
quietly replace mufin7=-.501 if((resid1<resid0)&(e(N)==1665))
quietly replace ser71=ter71 if((resid1<resid0)&(e(N)==1665))
quietly replace ser72=ter72 if((resid1<resid0)&(e(N)==1665))
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
```

```
*
*IV.B. Mu = -1
*
quietly replace ter71=log(ser7)
quietly replace ter72=(ser7^(1/sigma37)-1)/(1/sigma37)
#delimit ;
quietly reg lbidprice ser11 ser12 ser21 ser22 ser61 ser62 ter71 ter72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd cer11
cer12;
#delimit cr
quietly replace resid1=e(rss)
quietly replace mufin7=-1.01 if((resid1<resid0)&(e(N)==1665))
quietly replace ser71=ter71 if((resid1<resid0)&(e(N)==1665))
quietly replace ser72=ter72 if((resid1<resid0)&(e(N)==1665))
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
*
}
*V. Final Regression
*
#delimit ;
reg lbidprice ser11 ser12 cer11 cer12 ser21 ser22 ser61 ser62 ser71 ser72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd;
reg lbidprice ser11 ser12 cer11 cer12 ser21 ser22 ser61 ser62 ser71 ser72 ser3 sersq3 ser4 sersq4
waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd , vce(hc3)
;
#delimit cr
*
*
**********
*Save coefficients for nonlinear run
*
matrix coco=e(b)
matrix ficoef=coco[1,11...]
*summarize mufin1  , meanonly
*scalar mu1=r(mean)
summarize mufin2  , meanonly
```

```
scalar mu2=r(mean)
summarize mufin6  , meanonly
scalar mu6=r(mean)
summarize mufin7  , meanonly
scalar mu7=r(mean)
scalar star6=sstar6
scalar star7=sstar7
scalar ec11=-_b[ser11]/_b[ser12]
scalar ec12=1/_b[ser12]
scalar cc11=-_b[cer11]/_b[cer12]
scalar cc12=1/_b[cer12]
scalar ec21=-_b[ser21]/_b[ser22]
scalar ec22=1/_b[ser22]
scalar ec61=max(.001,-_b[ser61]/_b[ser62])
scalar ec62=1/_b[ser62]
scalar ec71=max(.001,-_b[ser71]/_b[ser72])
scalar ec72=1/_b[ser72]
#delimit ;
matrix init=(ec11, ec12, cc11, cc12, ec21, ec22, ec61, ec62, ec71, ec72, ficoef, mu2, mu6, mu7,
star6, star7) ;
#delimit cr
matrix list init
*
**********************
*Run final nonlinear version
*
*set trace on
*
* The regressions--------------------------
#delimit ;
nl fullpriceA @ lbidprice ser1 ser2 ser3 ser4 pser6 pser7 waydist ytrate tax1b tax2b tax3b noas
nocity lakefront4 dfront snow1 snow2 ghet nearghet5 airport10 dap10 near2sch d2sch near5pri
d5pri nearhaz dhaz pollzone dsco neighamen freeway shopping hospital airport regpark railroad
historic polpop1-polpop4 village township copol vneare vnearsf vnearlf lowhigh highlow highhigh
cc1-cc4 co2-co5 dw2-dw5 clesd , parameters(sn11 sn12 cle11 cle12 sn21 sn22 sn61 sn62 sn71
sn72 sn31 sn32 sn41 sn42 sc1 sc2 alpha beta1 beta2 beta3 bx1 bx2 bx3 bx4 bx5 bx6 bx7 bx8 bx9
bx10 bx11 bx12 bx13 bx14 bx15 bx16 bx17 bx18 bx19 bx20 bx21 bx22 bx23 bx24 bx25 bx26
bx27 bx28 bx29 bx30 bx31 bx32 bx33 bx34 bx35 bx36 bx37 bx38 bx39 bx40 bx41 bx42 bx43
bx44 bx45 bx46 bx47 bx48 bx49 bx50 bx51 cle10 c0 mu2 mu6 mu7 star6 star7) initial(init)
variables(lbidprice ser1 ser2 ser3 ser4 pser6 pser7 waydist ytrate tax1b tax2b tax3b noas nocity
lakefront4 dfront snow1 snow2 ghet nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri
nearhaz dhaz pollzone dsco neighamen freeway shopping hospital airport regpark railroad historic
polpop1-polpop4 village township copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4
co2-co5 dw2-dw5 clesd ) iterate(5000) hasconstant(c0) vce(hc3) ;
#delimit cr
```

```
*
matrix stepco=e(b)
*
#delimit ;
nl fullprice @ lbidprice ser1 ser2 ser3 ser4 pser6 pser7 waydist ytrate tax1b tax2b tax3b noas
nocity lakefront4 dfront snow1 snow2 ghet nearghet5 airport10 dap10 near2sch d2sch near5pri
d5pri nearhaz dhaz pollzone dsco neighamen freeway shopping hospital airport regpark railroad
historic polpop1-polpop4 village township copol vneare vnearsf vnearlf lowhigh highlow highhigh
cc1-cc4 co2-co5 dw2-dw5 clesd , parameters(sn11 sn12 cle11 cle12 sn21 sn22 sn61 sn62 sn71
sn72 sn31 sn32 sn41 sn42 sc1 sc2 alpha beta1 beta2 beta3 bx1 bx2 bx3 bx4 bx5 bx6 bx7 bx8 bx9
bx10 bx11 bx12 bx13 bx14 bx15 bx16 bx17 bx18 bx19 bx20 bx21 bx22 bx23 bx24 bx25 bx26
bx27 bx28 bx29 bx30 bx31 bx32 bx33 bx34 bx35 bx36 bx37 bx38 bx39 bx40 bx41 bx42 bx43
bx44 bx45 bx46 bx47 bx48 bx49 bx50 bx51 cle10 c0 mu2 mu6 mu7 star6 star7) initial(stepco)
variables(lbidprice ser1 ser2 ser3 ser4 pser6 pser7 waydist ytrate tax1b tax2b tax3b noas nocity
lakefront4 dfront snow1 snow2 ghet nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri
nearhaz dhaz pollzone dsco neighamen freeway shopping hospital airport regpark railroad historic
polpop1-polpop4 village township copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4
co2-co5 dw2-dw5 clesd ) iterate(5000) hasconstant(c0) vce(hc3) ;
#delimit cr
*
generate b2sum=(1/_b[/sn12])+(1/_b[/cle12])
generate sig2cle=1/b2sum
generate b1b2sum=-(_b[/sn11]/_b[/sn12]+_b[/cle11]/_b[/cle12])
generate sig1cle=-b1b2sum*sig2cle
matrix nlco=e(b)
*****************
*Now on to the demand stuff
*Recover psi
*Regress psi on demand characteristics
*
*Estimate owner income
* Median owner income within a CBG is
* (Overall median income)*(9.5987-11.6604*(Share owner)
* -0.7441*log(Overall median income)+1.02293*Interaction)
*
generate double lmedy=log(Mediany_cbg)
#delimit ;
generate double owninc=Mediany_cbg*(9.5987-11.6604*Ownerocc_cbg/100
-0.7741*lmedy+1.02293*Ownerocc_cbg*lmedy/100) ;
#delimit cr
generate double lowninc=log(owninc)
*
*slope for ser1 – Cleveland only
*
generate double slopes1=0
```

```
#delimit ;
replace slopes1=(ser1^(1/sigma31)-sig1cle)/sig2cle if(clesd==1) ;
#delimit cr
*
*slope for ser2 – without Cleveland
*
generate double slopes2=0
replace slopes2=(ser2^(1/sigma32)-nlco[1,5])/nlco[1,6] if(clesd==0)
*
*slopes for ser6 and ser7 – above split only
*
generate double slopes6=0
generate double slopes7=0
#delimit ;
replace slopes6=(ser6^(1/sigma36)-nlco[1,7])/nlco[1,8] if(pser6>(_b[/star6]+nlco[1,7])) ;
replace slopes7=(ser7^(1/sigma37)-nlco[1,9])/nlco[1,10] if(pser7>(_b[/star7]+nlco[1,9])) ;
#delimit cr
sum slopes1 if(clesd==1)
sum slopes2 if(clesd==0)
sum slopes6 if(pser6>(_b[/star6]+nlco[1,7]))
sum slopes7 if(pser7>(_b[/star7]+nlco[1,9]))
*
generate double lslopes1=log(slopes1) if(slopes1>0)
generate double lslopes2=log(slopes2) if(slopes2>0)
generate double lslopes6=log(slopes6) if(slopes6>0)
generate double lslopes7=log(slopes7) if(slopes7>0)
*
regress lslopes1 lowninc , vce(hc3)
regress lslopes2 lowninc , vce(hc3)
regress lslopes6 lowninc , vce(hc3)
regress lslopes7 lowninc , vce(hc3)
*
#delimit ;
regress lslopes1 lowninc Pct65pls_cbg Pctkids_cbg Pctmar_cbg Pctenglish_cbg pctapi_cbg
pctlt1_ct pcths_cbg pctlesba_cbg pctba_cbg Pctgraddeg_cbg , vce(hc3) ;
regress lslopes2 lowninc Pct65pls_cbg Pctkids_cbg Pctmar_cbg Pctenglish_cbg pctapi_cbg
pctlt1_ct pcths_cbg pctlesba_cbg pctba_cbg Pctgraddeg_cbg , vce(hc3) ;
regress lslopes2 lowninc Pct65pls_cbg Pctkids_cbg Pctmar_cbg Pctforeign_cbg Pctenglish_cbg
pctapi_cbg pctlt1_ct pcths_cbg pctlesba_cbg pctba_cbg Pctgraddeg_cbg  , vce(hc3) ;
regress lslopes6 lowninc Pct65pls_cbg Pctkids_cbg Pctmar_cbg Pctenglish_cbg pctapi_cbg
pctlt1_ct pcths_cbg pctlesba_cbg pctba_cbg Pctgraddeg_cbg , vce(hc3) ;
regress lslopes7 lowninc Pct65pls_cbg Pctkids_cbg Pctmar_cbg Pctenglish_cbg pctapi_cbg
pctlt1_ct pcths_cbg pctlesba_cbg pctba_cbg Pctgraddeg_cbg , vce(hc3) ;
#delimit cr
log
```

Program for Panel B of Table 7

```
*********************************
*Define variables for envelope
*
generate double sigma32=1
generate double sigma36=1
generate double sigma37=1
*
generate double ser21=0
generate double ser22=0
generate double ser61=0
generate double ser62=0
generate double ser71=0
generate double ser72=0
*
*
************************
*Nonlinear regression program
*
*set trace on
*
program nlfullprice
syntax varlist(min=67 max=67) [if] , at(name) mycoefs(numlist)
local lbidprice : word 1 of `varlist'
local pser1 : word 2 of `varlist'
local ser21 : word 3 of `varlist'
local ser2 : word 4 of `varlist'
local ser3 : word 5 of `varlist'
local ser4 : word 6 of `varlist'
local ser61 : word 7 of `varlist'
local ser6 : word 8 of `varlist'
local ser71 : word 9 of `varlist'
local ser7 : word 10 of `varlist'
local waydist : word 11 of `varlist'
local ytrate : word 12 of `varlist'
local tax1b : word 13 of `varlist'
local tax2b : word 14 of `varlist'
local tax3b : word 15 of `varlist'
local noas : word 16 of `varlist'
local nocity : word 17 of `varlist'
local lakefront4 : word 18 of `varlist'
local dfront : word 19 of `varlist'
local snow1 : word 20 of `varlist'
local snow2 : word 21 of `varlist'
```

```
local ghet : word 22 of `varlist'
local nearghet5 : word 23 of `varlist'
local airport10 : word 24 of `varlist'
local dap10 : word 25 of `varlist'
local near2sch : word 26 of `varlist'
local d2sch : word 27 of `varlist'
local near5pri : word 28 of `varlist'
local d5pri : word 29 of `varlist'
local nearhaz : word 30 of `varlist'
local dhaz : word 31 of `varlist'
local pollzone : word 32 of `varlist'
local dsco : word 33 of `varlist'
local neighamen : word 34 of `varlist'
local freeway : word 35 of `varlist'
local shopping : word 36 of `varlist'
local hospital : word 37 of `varlist'
local airport : word 38 of `varlist'
local regpark : word 39 of `varlist'
local railroad : word 40 of `varlist'
local historic : word 41 of `varlist'
local polpop1 : word 42 of `varlist'
local polpop2 : word 43 of `varlist'
local polpop3 : word 44 of `varlist'
local polpop4 : word 45 of `varlist'
local village : word 46 of `varlist'
local township : word 47 of `varlist'
local copol : word 48 of `varlist'
local vneare : word 49 of `varlist'
local vnearsf : word 50 of `varlist'
local vnearlf : word 51 of `varlist'
local lowhigh : word 52 of `varlist'
local highlow : word 53 of `varlist'
local highhigh : word 54 of `varlist'
local cc1 : word 55 of `varlist'
local cc2 : word 56 of `varlist'
local cc3 : word 57 of `varlist'
local cc4 : word 58 of `varlist'
local co2 : word 59 of `varlist'
local co3 : word 60 of `varlist'
local co4 : word 61 of `varlist'
local co5 : word 62 of `varlist'
local dw2 : word 63 of `varlist'
local dw3 : word 64 of `varlist'
local dw4 : word 65 of `varlist'
local dw5 : word 66 of `varlist'
```

```
local clesd : word 67 of `varlist'
#delimit ;
tempname sn11 sn12 sn21 sn22 sn61 sn62 sn71 sn72 sn31 sn32 sn41 sn42 sc1 sc2 alpha beta1
beta2 beta3  ;
tempname bx1 bx2 bx3 bx4 bx5 bx6 bx7 bx8 bx9 bx10 bx11 bx12 bx13 bx14 bx15 bx16 bx17
bx18 bx19 bx20 bx21 bx22 bx23 bx24 bx25 bx26 bx27 bx28 bx29 bx30 bx31 bx32 bx33 bx34
bx35 bx36 bx37 bx38 bx39 bx40 bx41 bx42 bx43 bx44 bx45 bx46 bx47 bx48 bx49 bx50 bx51 c0 ;
tempname sigma32 sigma36 sigma37 cle10 cle11 cle12 ;
#delimit cr
scalar `sn11' = `at'[1, 1]
scalar `sn12' = `at'[1, 2]
scalar `cle11' = `at'[1, 3]
scalar `cle12' = `at'[1, 4]
scalar `sn21' = `at'[1, 5]
scalar `sn22' = `at'[1, 6]
scalar `sn61' = `at'[1, 7]
scalar `sn62' = `at'[1, 8]
scalar `sn71' = `at'[1, 9]
scalar `sn72' = `at'[1, 10]
scalar `sn31' = `at'[1, 11]
scalar `sn32' = `at'[1, 12]
scalar `sn41' = `at'[1, 13]
scalar `sn42' = `at'[1, 14]
scalar `sc1' = `at'[1, 15]
scalar `sc2' = `at'[1, 16]
scalar `alpha' = `at'[1, 17]
scalar `beta1' = `at'[1, 18]
scalar `beta2' = `at'[1, 19]
scalar `beta3' = `at'[1, 20]
scalar `bx1' = `at'[1, 21]
scalar `bx2' = `at'[1, 22]
scalar `bx3' = `at'[1, 23]
scalar `bx4' = `at'[1, 24]
scalar `bx5' = `at'[1, 25]
scalar `bx6' = `at'[1, 26]
scalar `bx7' = `at'[1, 27]
scalar `bx8' = `at'[1, 28]
scalar `bx9' = `at'[1, 29]
scalar `bx10' = `at'[1, 30]
scalar `bx11' = `at'[1, 31]
scalar `bx12' = `at'[1, 32]
scalar `bx13' = `at'[1, 33]
scalar `bx14' = `at'[1, 34]
scalar `bx15' = `at'[1, 35]
scalar `bx16' = `at'[1, 36]
```

```
scalar `bx17' = `at'[1, 37]
scalar `bx18' = `at'[1, 38]
scalar `bx19' = `at'[1, 39]
scalar `bx20' = `at'[1, 40]
scalar `bx21' = `at'[1, 41]
scalar `bx22' = `at'[1, 42]
scalar `bx23' = `at'[1, 43]
scalar `bx24' = `at'[1, 44]
scalar `bx25' = `at'[1, 45]
scalar `bx26' = `at'[1, 46]
scalar `bx27' = `at'[1, 47]
scalar `bx28' = `at'[1, 48]
scalar `bx29' = `at'[1, 49]
scalar `bx30' = `at'[1, 50]
scalar `bx31' = `at'[1, 51]
scalar `bx32' = `at'[1, 52]
scalar `bx33' = `at'[1, 53]
scalar `bx34' = `at'[1, 54]
scalar `bx35' = `at'[1, 55]
scalar `bx36' = `at'[1, 56]
scalar `bx37' = `at'[1, 57]
scalar `bx38' = `at'[1, 58]
scalar `bx39' = `at'[1, 59]
scalar `bx40' = `at'[1, 60]
scalar `bx41' = `at'[1, 61]
scalar `bx42' = `at'[1, 62]
scalar `bx43' = `at'[1, 63]
scalar `bx44' = `at'[1, 64]
scalar `bx45' = `at'[1, 65]
scalar `bx46' = `at'[1, 66]
scalar `bx47' = `at'[1, 67]
scalar `bx48' = `at'[1, 68]
scalar `bx49' = `at'[1, 69]
scalar `bx50' = `at'[1, 70]
scalar `bx51' = `at'[1, 71]
scalar `cle10' = `at'[1, 72]
scalar `c0' = `at'[1, 73]
scalar `sigma32' = `at'[1, 74]
scalar `sigma36' = `at'[1, 75]
scalar `sigma37' = `at'[1, 76]
local ex21: word 1 of `mycoefs'
local ex61: word 2 of `mycoefs'
local ex71: word 3 of `mycoefs'
tempvar bids1 bids2 bids6 bids7
tempvar bids bida bidu bidx phat
```

```
tempvar ex22 ex62 ex72
tempvar ser11 ser12 cser11 cser12
tempvar ser22 ser62 ser72
*
generate double `ex22'=`ex21'+1/`sigma32' `if'
generate double `ex62'=`ex61'+1/`sigma36' `if'
generate double `ex72'=`ex71'+1/`sigma37' `if'
generate double `ser11'=(1-`clesd')*`pser1' `if'
generate double `ser12'=`ser11'^2 `if'
generate double `cser11'=`clesd'*`pser1' `if'
generate double `cser12'=`cser11'^2 `if'
generate double `ser22'=(`ser2'^`ex22'-1)/`ex22' `if'
generate double `ser62'=(`ser6'^`ex62'-1)/`ex62' `if'
generate double `ser72'=(`ser7'^`ex72'-1)/`ex72' `if'
*
#delimit ;
generate double `bids1'=`sn11'*`ser11'+`sn12'*`ser12'+
`cle11'*`cser11'+`cle12'*`cser12'+`cle10'*`clesd'  `if' ;
generate double `bids2'=-(`sn21'/`sn22')*`ser21'
+(1/`sn22')*`ser22' `if' ;
generate double `bids6'=-(`sn61'/`sn62')*`ser61'
+(1/`sn62')*`ser62' `if' ;
generate double `bids7'=-(`sn71'/`sn72')*`ser71'
+(1/`sn72')*`ser72' `if' ;
generate double `bids'=`bids1'+`bids2'
+`bids6'+`bids7' `if' ;
generate double `bida'=`sn31'*`ser3'+`sn32'*(`ser3'^2)+ `sn41'*`ser4'+`sn42'*(`ser4'^2)  `if' ;
generate double `bidu' = `sc1'*`waydist'
+`sc2'*((`waydist')^2)+`bx48'*`dw2'+`bx49'*`dw3'
+`bx50'*`dw4'+`bx51'*`dw5' `if' ;
generate double `bidx' = `alpha'*`ytrate'
+`beta1'*`tax1b'+`beta2'*`tax2b'+`beta3'*`tax3b'
+`bx1'*`noas'+`bx2'*`nocity'+`bx3'*`lakefront4'+`bx4'*`dfront'
+`bx5'*`snow1'+`bx6'*`snow2'+`bx7'*`ghet'+`bx8'*`nearghet5'
+`bx9'*`airport10'+`bx10'*`dap10'+`bx11'*`near2sch'
+`bx12'*`d2sch'+`bx13'*`near5pri'+`bx14'*`d5pri'
+`bx15'*`nearhaz'+`bx16'*`dhaz'
+`bx17'*`pollzone'+`bx18'*`dsco'
+`bx19'*`neighamen'+`bx20'*`freeway'
+`bx21'*`shopping'+`bx22'*`hospital'+`bx23'*`airport'
+`bx24'*`regpark'+`bx25'*`railroad'+`bx26'*`historic'
+`bx27'*`polpop1'+`bx28'*`polpop2'+`bx29'*`polpop3'+`bx30'*`polpop4'
+`bx31'*`village'+`bx32'*`township'+`bx33'*`copol'+`bx34'*`vneare'
+`bx35'*`vnearsf'+`bx36'*`vnearlf'+`bx37'*`lowhigh'+`bx38'*`highlow'
+`bx39'*`highhigh'+`bx40'*`cc1'+`bx41'*`cc2'+`bx42'*`cc3'
```

```
+`bx43'*`cc4'+`bx44'*`co2'+`bx45'*`co3'+`bx46'*`co4'+`bx47'*`co5' `if' ;
generate double `phat' =`c0'+`bids'+`bida'+`bidu'+`bidx' `if' ;
replace `lbidprice' = `phat' `if' ;
#delimit cr
end
*********************************************
*
*Prepare Loops
*
generate maxs2=1
generate maxs6=1
generate maxs7=1
generate mins2=1
generate mins6=1
generate mins7=1
generate resid0=999
generate resid1=0
*
*Loops
*
forvalues i=1/4 {
scalar ex2=0+(-.99)*(`i'==1)+(1/2)*(`i'==3)+(1)*(`i'==4)
quietly replace ser21=1/ser2 if(`i'==1)
quietly replace ser22=log(ser2) if(`i'==1)
quietly replace ser21=log(ser2) if(`i'==2)
quietly replace ser22=ser2-1 if(`i'==2)
quietly replace ser21=ser2^(1/2)/(1/2) if(`i'==3)
quietly replace ser22=(ser2^(3/2)-1)/(3/2) if(`i'==3)
quietly replace ser21=ser2 if(`i'==4)
quietly replace ser22=(ser2^2-1)/2 if(`i'==4)
*
forvalues j=1/4 {
scalar ex6=0+(-.99)*(`j'==1)+(1/2)*(`j'==3)+(1)*(`j'==4)
quietly replace ser61=1/ser6 if(`j'==1)
quietly replace ser62=log(ser6) if(`j'==1)
quietly replace ser62=ser6-1 if(`j'==2)
quietly replace ser61=ser6^(1/2)/(1/2) if(`j'==3)
quietly replace ser62=(ser6^(3/2)-1)/(3/2) if(`j'==3)
quietly replace ser61=ser6 if(`j'==4)
quietly replace ser62=(ser6^2-1)/2 if(`j'==4)
*
forvalues k=1/4 {
scalar ex7=0+(-.99)*(`k'==1)+(1/2)*(`k'==3)+(1)*(`k'==4)
quietly replace ser71=1/ser7 if(`k'==1)
quietly replace ser72=log(ser7) if(`k'==1)
```

```
quietly replace ser71=log(ser7) if(`k'==2)
quietly replace ser72=ser7-1 if(`k'==2)
quietly replace ser71=ser7^(1/2)/(1/2) if(`k'==3)
quietly replace ser72=(ser7^(3/2)-1)/(3/2) if(`k'==3)
quietly replace ser71=ser7 if(`k'==4)
quietly replace ser72=(ser7^2-1)/2 if(`k'==4)
*
**************
*V. Final Regression
*
#delimit ;
quietly reg lbidprice ser1 sersq1 cser1 csersq1 ser21 ser22 ser61 ser62 ser71 ser72 ser3 sersq3 ser4
sersq4 waydist waydist2 ytrate tax1b tax2b tax3b noas nocity lakefront4 dfront snow1 snow2 ghet
nearghet5 airport10 dap10 near2sch d2sch near5pri d5pri nearhaz dhaz pollzone dsco neighamen
freeway shopping hospital airport regpark railroad historic polpop1-polpop4 village township
copol vneare vnearsf vnearlf lowhigh highlow highhigh cc1-cc4 co2-co5 dw2-dw5 clesd ;
#delimit cr
*
*********************************
*Save coefficients for nonlinear run
*
matrix coco=e(b)
matrix coefone=coco[1,1..4]
matrix coeftwo=coco[1,11...]
scalar ec21=-_b[ser21]/_b[ser22]
scalar ec22=1/_b[ser22]
scalar ec61=-_b[ser61]/_b[ser62]
scalar ec62=1/_b[ser62]
scalar ec71=-_b[ser71]/_b[ser72]
scalar ec72=1/_b[ser72]
scalar sig32=1
scalar sig36=1
scalar sig37=1
*
matrix exx=(ex2, ex6, ex7)
local ex21=exx[1,1]
local ex61=exx[1,2]
local ex71=exx[1,3]
*
*
#delimit ;
matrix init=(coefone, ec21, ec22, ec61, ec62, ec71, ec72, coeftwo, sig32, sig36, sig37) ;
#delimit cr
*
**********************
```

```
*Run final nonlinear version
*
*set trace on
*
* The regression---------------------------
display "**********"
display "**********" `i' `j' `k'
#delimit ;
nl fullprice @ lbidprice pser1 ser21 ser2 ser3 ser4 ser61 ser6 ser71 ser7 waydist ytrate tax1b tax2b
tax3b noas nocity lakefront4 dfront snow1 snow2 ghet nearghet5 airport10 dap10 near2sch d2sch
near5pri d5pri nearhaz dhaz pollzone dsco neighamen freeway shopping hospital airport regpark
railroad historic polpop1-polpop4 village township copol vneare vnearsf vnearlf lowhigh highlow
highhigh cc1-cc4 co2-co5 dw2-dw5 clesd , parameters(sn11 sn12 cle11 cle12 sn21 sn22 sn61 sn62
sn71 sn72 sn31 sn32 sn41 sn42 sc1 sc2 alpha beta1 beta2 beta3 bx1 bx2 bx3 bx4 bx5 bx6 bx7 bx8
bx9 bx10 bx11 bx12 bx13 bx14 bx15 bx16 bx17 bx18 bx19 bx20 bx21 bx22 bx23 bx24 bx25 bx26
bx27 bx28 bx29 bx30 bx31 bx32 bx33 bx34 bx35 bx36 bx37 bx38 bx39 bx40 bx41 bx42 bx43
bx44 bx45 bx46 bx47 bx48 bx49 bx50 bx51 cle10 c0 sigma32 sigma36 sigma37) initial(init)
variables(lbidprice pser1 ser21 ser2 ser3 ser4 ser61 ser6 ser71 ser7 waydist ytrate tax1b tax2b
tax3b noas nocity lakefront4 dfront snow1 snow2 ghet nearghet5 airport10 dap10 near2sch d2sch
near5pri d5pri nearhaz dhaz pollzone dsco neighamen freeway shopping hospital airport regpark
railroad historic polpop1-polpop4 village township copol vneare vnearsf vnearlf lowhigh highlow
highhigh cc1-cc4 co2-co5 dw2-dw5 clesd ) iterate(5000) hasconstant(c0) vce(hc3) mycoefs(`ex21'
`ex61' `ex71') ;
#delimit cr
*
matrix stepco=e(b)
quietly replace resid1=e(rss)
quietly replace resid0=min(resid1,resid0) if(e(N)==1665)
quietly replace maxs2=max(maxs2, _b[/sigma32]) if(e(N)==1665)
quietly replace mins2=min(mins2, _b[/sigma32]) if(e(N)==1665)
quietly replace maxs6=max(maxs6, _b[/sigma36]) if(e(N)==1665)
quietly replace mins6=min(mins6, _b[/sigma36]) if(e(N)==1665)
quietly replace maxs7=max(maxs7, _b[/sigma37]) if(e(N)==1665)
quietly replace mins7=min(mins7, _b[/sigma37]) if(e(N)==1665)
*
}
}
}
sum resid0
sum maxs2 maxs6 maxs7
sum mins2 mins6 mins7
log
```

Appendix D: Detailed Results

This appendix presents the full regression results for Tables 3-6 in the text. The tables are:

Results for Column 1 of Table 5

Results for Column 2 of Table 5

Results for Column 3 of Table 5 (and for Tables 3 and 4)

Results for Column 4 of Table 5

Results for Table 7

Results for Column 1 of Table 5

```
Linear regression                                       Number of obs =    1665
                                                        F( 64,  1599) =       .
                                                        Prob > F      =       .
                                                        R-squared     =  0.7046
                                                        Root MSE      =  .14739

------------------------------------------------------------------------------
             |               Robust HC3
   lbidprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       nser1 |   .1268396   .0580838     2.18   0.029     .0129112     .240768
        ser2 |   .4825989   .0599763     8.05   0.000     .3649584    .6002394
        ser3 |   .1757928   .0711233     2.47   0.014     .0362881    .3152975
        ser4 |   .0317136    .098924     0.32   0.749    -.1623208    .2257481
        ser6 |   .2633311   .0342152     7.70   0.000     .1962197    .3304425
        ser7 |   .4849493   .0689321     7.04   0.000     .3497425    .6201562
     waydist |  -.0259707   .0072195    -3.60   0.000    -.0401315     -.01181
    waydist2 |   .0003902   .0001552     2.51   0.012     .0000859    .0006946
      ytrate |   .5883636   5.085243     0.12   0.908     -9.38608    10.56281
       tax1b |   2.907146   1.198331     2.43   0.015     .5566817    5.257611
       tax2b |  -1.510871   1.333538    -1.13   0.257    -4.126536    1.104795
       tax3b |   3.656837    2.04802     1.79   0.074    -.3602506    7.673924
        noas |  -.0471537   .0332656    -1.42   0.157    -.1124024    .0180949
      nocity |   .0609611   .0380593     1.60   0.109    -.0136902    .1356124
  lakefront4 |   .0811194   .0240238     3.38   0.001     .0339979    .1282409
      dfront |  -.0367966   .0192893    -1.91   0.057    -.0746316    .0010384
       snow1 |   .0348531   .0065044     5.36   0.000     .0220951     .047611
       snow2 |  -.0016168   .0003925    -4.12   0.000    -.0023867   -.0008469
        ghet |   .0622065   .0411238     1.51   0.131    -.0184557    .1428687
   nearghet5 |   .0022464   .0248332     0.09   0.928    -.0464626    .0509554
   airport10 |   .0135611   .0306629     0.44   0.658    -.0465825    .0737047
       dap10 |  -.0007691   .0035323    -0.22   0.828    -.0076976    .0061594
    near2sch |   .0002469   .0214712     0.01   0.991    -.0418678    .0423616
       d2sch |   -.024944   .0104123    -2.40   0.017    -.0453671   -.0045208
    near5pri |   .0110514   .0226473     0.49   0.626    -.0333701    .0554729
       d5pri |  -.0015937   .0051012    -0.31   0.755    -.0115995    .0084121
     nearhaz |  -.0614919   .0166628    -3.69   0.000    -.0941751   -.0288087
        dhaz |   .0736739    .022446     3.28   0.001     .0296472    .1177007
    pollzone |  -.2444783   .0677511    -3.61   0.000    -.3773686   -.1115881
        dsco |   .0076019   .0037745     2.01   0.044     .0001984    .0150055
   neighamen |   .0227113   .0082467     2.75   0.006     .0065358    .0388869
     freeway |   .0130856   .0117812     1.11   0.267    -.0100226    .0361937
    shopping |  -.0164728   .0094229    -1.75   0.081    -.0349553    .0020096
    hospital |   .0135813   .0100208     1.36   0.176    -.0060739    .0332365
     airport |   .0124053   .0215974     0.57   0.566    -.0299568    .0547675
     regpark |   .0133628   .0109355     1.22   0.222    -.0080865    .0348122
    railroad |  -.0304192   .0103868    -2.93   0.003    -.0507924   -.0100459
    historic |    .001386   .0182839     0.08   0.940    -.0344768    .0372489
     polpop1 |   -.000021   5.06e-06    -4.15   0.000    -.0000309   -.0000111
     polpop2 |   5.60e-06   1.35e-06     4.14   0.000     2.94e-06    8.26e-06
     polpop3 |  -4.73e-07   1.14e-07    -4.14   0.000    -6.98e-07   -2.49e-07
     polpop4 |   7.42e-09   1.79e-09     4.14   0.000     3.90e-09    1.09e-08
     village |  -.0956368   .0491547    -1.95   0.052    -.1920513    .0007776
```

```
    township |   -.099593   .0369366    -2.70   0.007    -.1720422   -.0271438
       copol |  -.1858435    .047844    -3.88   0.000    -.2796871   -.0919999
      vneare |  -.0088544   .0243118    -0.36   0.716    -.0565407    .0388319
     vnearsf |  -.0208257   .0282523    -0.74   0.461    -.0762412    .0345897
     vnearlf |  -.0927772   .0414661    -2.24   0.025    -.1741108   -.0114435
     lowhigh |  -.1103189   .0347876    -3.17   0.002     -.178553   -.0420848
     highlow |  -.0316323   .0150703    -2.10   0.036    -.0611919   -.0020726
    highhigh |  -.0842861   .0208707    -4.04   0.000    -.1252229   -.0433493
         cc1 |  -.2260143   .0492004    -4.59   0.000    -.3225183   -.1295103
         cc2 |  -.0907338    .039287    -2.31   0.021    -.1677931   -.0136744
         cc3 |  -.1200086   .0331765    -3.62   0.000    -.1850826   -.0549346
         cc4 |  -.0519578   .0158556    -3.28   0.001    -.0830577   -.0208579
         co2 |  -.0120743   .0480877    -0.25   0.802    -.1063959    .0822472
         co3 |   .2441672   .0321505     7.59   0.000     .1811056    .3072287
         co4 |   .1516757   .0321571     4.72   0.000     .0886013    .2147502
         co5 |   .0259361   .0452631     0.57   0.567    -.0628452    .1147174
         dw2 |   .0546066   .0181279     3.01   0.003     .0190497    .0901635
         dw3 |   .0929252   .0344785     2.70   0.007     .0252974     .160553
         dw4 |    .068312    .033178     2.06   0.040      .003235    .1333891
         dw5 |  -.0044389   .0242229    -0.18   0.855     -.051951    .0430731
       clesd |   .1390735   .0662898     2.10   0.036     .0090494    .2690976
       cser1 |  -.0605892   .0820061    -0.74   0.460    -.2214399    .1002615
       _cons |   10.69716   .1164916    91.83   0.000     10.46867    10.92565
------------------------------------------------------------------------------
```

Results for Column 2 of Table 5

nser1=ser1 outside the Clevelands

Linear regression

Number of obs	1665
F(71, 1592)	.
Prob > F	.
R-squared	0.7156
Root MSE	.14493

lbidprice	Coef.	Robust HC3 Std. Err.	t	P>|t|	[95% Conf.	Interval]
nser1	.2447737	.2480284	0.99	0.324	-.2417228	.7312703
nsersq1	-.2086156	.3583879	-0.58	0.561	-.9115775	.4943462
ser2	-.0862194	.2631166	-0.33	0.743	-.6023108	.4298721
sersq2	.604884	.2848972	2.12	0.034	.0460709	1.163697
ser3	1.146119	.387741	2.96	0.003	.3855824	1.906655
sersq3	-1.756795	.6942836	-2.53	0.011	-3.118601	-.3949891
ser4	.4262766	.2119523	2.01	0.044	.0105418	.8420115
sersq4	-.8556545	.3944914	-2.17	0.030	-1.629432	-.0818772
ser6	-.2119642	.1247291	-1.70	0.089	-.4566148	.0326864
sersq6	.4114305	.1031232	3.99	0.000	.2091589	.6137021
ser7	-.2440062	.4305656	-0.57	0.571	-1.088541	.6005289
sersq7	.4831619	.2871038	1.68	0.093	-.0799794	1.046303
waydist	-.0249038	.0071511	-3.48	0.001	-.0389305	-.0108772
waydist2	.0003668	.000153	2.40	0.017	.0000667	.000667
ytrate	1.340511	5.179442	0.26	0.796	-8.818733	11.49975
tax1b	3.671586	1.278768	2.87	0.004	1.163339	6.179833
tax2b	-1.449712	1.343924	-1.08	0.281	-4.085758	1.186334
tax3b	3.391758	2.042074	1.66	0.097	-.6136795	7.397195
noas	-.0358389	.0348668	-1.03	0.304	-.1042286	.0325508
nocity	.0663456	.0385985	1.72	0.086	-.0093636	.1420548
lakefront4	.0708946	.023895	2.97	0.003	.0240257	.1177636
dfront	-.0295842	.0191263	-1.55	0.122	-.0670996	.0079312
snow1	.0291383	.0066272	4.40	0.000	.0161394	.0421373
snow2	-.0013425	.0003959	-3.39	0.001	-.002119	-.000566
ghet	-.0276454	.0416216	-0.66	0.507	-.1092843	.0539935
nearghet5	.0004653	.0250272	0.02	0.985	-.0486244	.049555
airport10	.0316659	.0324746	0.98	0.330	-.0320315	.0953633
dap10	-.0015164	.0036247	-0.42	0.676	-.0086261	.0055933
near2sch	-.0137144	.022095	-0.62	0.535	-.0570527	.0296238
d2sch	-.0169097	.0108202	-1.56	0.118	-.038133	.0043136
near5pri	.0213518	.0227986	0.94	0.349	-.0233666	.0660702
d5pri	-.0025683	.0050149	-0.51	0.609	-.0124047	.0072682
nearhaz	-.0604038	.016767	-3.60	0.000	-.0932915	-.027516
dhaz	.0729477	.0227546	3.21	0.001	.0283156	.1175798
pollzone	-.2301017	.0694335	-3.31	0.001	-.3662924	-.093911
dsco	.0079135	.0039017	2.03	0.043	.0002604	.0155666
neighamen	.0195654	.0080565	2.43	0.015	.0037629	.0353679
freeway	.0131863	.0116772	1.13	0.259	-.0097181	.0360906
shopping	-.0161171	.0093835	-1.72	0.086	-.0345225	.0022882
hospital	.0119001	.0097606	1.22	0.223	-.0072449	.0310452
airport	.0296603	.0217098	1.37	0.172	-.0129225	.0722431

regpark	.0026876	.0107996	0.25	0.804	-.0184953	.0238704
railroad	-.0303969	.0103389	-2.94	0.003	-.0506762	-.0101176
historic	.0059354	.0181072	0.33	0.743	-.0295809	.0414518
polpop1	-.000021	4.96e-06	-4.23	0.000	-.0000307	-.0000113
polpop2	5.65e-06	1.34e-06	4.22	0.000	3.03e-06	8.27e-06
polpop3	-4.83e-07	1.14e-07	-4.25	0.000	-7.05e-07	-2.60e-07
polpop4	7.58e-09	1.78e-09	4.25	0.000	4.08e-09	1.11e-08
village	-.1037408	.0491806	-2.11	0.035	-.2002063	-.0072753
township	-.1080501	.0429801	-2.51	0.012	-.1923537	-.0237466
copol	-.1786297	.0461647	-3.87	0.000	-.2691797	-.0880797
vneare	-.0069109	.023617	-0.29	0.770	-.0532346	.0394128
vnearsf	-.0228816	.0263448	-0.87	0.385	-.0745557	.0287925
vnearlf	-.0857655	.0419138	-2.05	0.041	-.1679775	-.0035536
lowhigh	-.0976074	.0333175	-2.93	0.003	-.1629583	-.0322565
highlow	-.0335687	.0149426	-2.25	0.025	-.0628779	-.0042595
highhigh	-.0742902	.0209459	-3.55	0.000	-.1153747	-.0332057
cc1	-.2114088	.0497348	-4.25	0.000	-.3089615	-.1138561
cc2	-.0746088	.0389553	-1.92	0.056	-.1510178	.0018002
cc3	-.0955874	.0336122	-2.84	0.005	-.1615162	-.0296586
cc4	-.0406254	.0156239	-2.60	0.009	-.071271	-.0099798
co2	-.0113323	.0497601	-0.23	0.820	-.1089344	.0862699
co3	.2461417	.0351948	6.99	0.000	.1771087	.3151747
co4	.1495149	.0326531	4.58	0.000	.0854672	.2135625
co5	.0131668	.0468776	0.28	0.779	-.0787816	.1051152
dw2	.0451286	.018113	2.49	0.013	.0096007	.0806566
dw3	.0876259	.0352213	2.49	0.013	.0185409	.1567109
dw4	.0541736	.034745	1.56	0.119	-.0139771	.1223243
dw5	-.0150678	.0244736	-0.62	0.538	-.0630716	.032936
clesd	.3625578	.1020769	3.55	0.000	.1623386	.562777
cser1	-1.297563	.3910954	-3.32	0.001	-2.064679	-.530447
csersq1	1.673976	.4875996	3.43	0.001	.7175708	2.63038
_cons	10.96158	.2040944	53.71	0.000	10.56126	11.3619

Results for Column 3 of Table 5

```
Iteration 26:  residual SS =  33.31685

Nonlinear regression                                  Number of obs =      1665
                                                      R-squared     =    0.7167
                                                      Adj R-squared =    0.7035
                                                      Root MSE      =  .1447549
                                                      Res. dev.     = -1787.611

------------------------------------------------------------------------------
             |               Robust HC3
   lbidprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       /sn11 |   .5675861   .3912368     1.45   0.147     -.199808     1.33498
       /sn12 |  -2.228119   3.551583    -0.63   0.531    -9.194397    4.738158
      /cle11 |   .3903411   .0230219    16.96   0.000     .3451845    .4354976
      /cle12 |   .3003638   .0884515     3.40   0.001       .12687    .4738577
       /sn21 |   .2177236   .0338432     6.43   0.000     .1513417    .2841055
       /sn22 |   1.313947   .5328454     2.47   0.014     .2687931      2.3591
       /sn61 |   .2493629   .0940076     2.65   0.008     .0649711    .4337547
       /sn62 |   1.257058   .3475831     3.62   0.000     .5752883    1.938827
       /sn71 |   .5000756   .0211418    23.65   0.000     .4586068    .5415444
       /sn72 |   .8543962   .2590351     3.30   0.001       .34631    1.362482
       /sn31 |   1.194496   .3916125     3.05   0.002      .426365    1.962627
       /sn32 |   -1.82908   .6986289    -2.62   0.009     -3.19941   -.4587493
       /sn41 |   .3273999   .2171556     1.51   0.132    -.0985413    .7533412
       /sn42 |  -.7200687    .396136    -1.82   0.069    -1.497073    .0569351
        /sc1 |  -.0264092   .0071541    -3.69   0.000    -.0404417   -.0123767
        /sc2 |    .000393    .000153     2.57   0.010     .0000929    .0006931
      /alpha |   1.909203   5.159343     0.37   0.711    -8.210626    12.02903
      /beta1 |    3.84045   1.295102     2.97   0.003     1.300162    6.380738
      /beta2 |  -1.973518   1.389422    -1.42   0.156    -4.698809    .7517729
      /beta3 |   4.096999   2.114305     1.94   0.053    -.0501187    8.244117
        /bx1 |  -.0417474   .0352144    -1.19   0.236    -.1108191    .0273242
        /bx2 |   .0668916   .0384302     1.74   0.082    -.0084875    .1422708
        /bx3 |   .0697976   .0238433     2.93   0.003     .0230301    .1165651
        /bx4 |   -.029237   .0191037    -1.53   0.126     -.066708     .008234
        /bx5 |   .0290501   .0065778     4.42   0.000     .0161481    .0419521
        /bx6 |  -.0013255   .0003923    -3.38   0.001     -.002095   -.0005561
        /bx7 |  -.0330325   .0445969    -0.74   0.459    -.1205074    .0544425
        /bx8 |  -.0034975   .0245506    -0.14   0.887    -.0516523    .0446574
        /bx9 |   .0304152   .0326833     0.93   0.352    -.0336918    .0945222
       /bx10 |  -.0017637   .0037256    -0.47   0.636    -.0090713    .0055439
       /bx11 |  -.0132642   .0220083    -0.60   0.547    -.0564325    .0299042
       /bx12 |  -.0165756   .0107795    -1.54   0.124    -.0377191    .0045679
       /bx13 |   .0210441   .0228778     0.92   0.358    -.0238298    .0659181
       /bx14 |  -.0032108   .0049803    -0.64   0.519    -.0129795    .0065579
       /bx15 |   -.061092   .0168335    -3.63   0.000    -.0941102   -.0280738
       /bx16 |   .0752312   .0227965     3.30   0.001     .0305167    .1199456
       /bx17 |  -.2337177   .0698929    -3.34   0.001    -.3708096   -.0966257
       /bx18 |   .0081046   .0039228     2.07   0.039     .0004103     .015799
       /bx19 |   .0188628    .008097     2.33   0.020     .0029809    .0347447
       /bx20 |   .0123076   .0115844     1.06   0.288    -.0104147    .0350298
       /bx21 |  -.0158876   .0094061    -1.69   0.091    -.0343373    .0025621
```

```
        /bx22 |   .0118424   .0097779     1.21   0.226    -.0073365    .0310213
        /bx23 |   .0291222    .021719     1.34   0.180    -.0134786     .071723
        /bx24 |    .003101   .0107685     0.29   0.773    -.0180209    .0242229
        /bx25 |  -.0305058   .0103343    -2.95   0.003    -.0507762   -.0102355
        /bx26 |   .0068943    .018202     0.38   0.705    -.0288082    .0425968
        /bx27 |  -.0000209   4.87e-06    -4.30   0.000    -.0000305   -.0000114
        /bx28 |   5.69e-06   1.33e-06     4.27   0.000     3.08e-06    8.30e-06
        /bx29 |  -4.89e-07   1.14e-07    -4.30   0.000    -7.12e-07   -2.66e-07
        /bx30 |   7.69e-09   1.79e-09     4.30   0.000     4.18e-09    1.12e-08
        /bx31 |  -.1052797   .0492837    -2.14   0.033    -.2019477   -.0086118
        /bx32 |  -.1120045   .0432011    -2.59   0.010    -.1967416   -.0272674
        /bx33 |  -.1757333   .0449499    -3.91   0.000    -.2639005    -.087566
        /bx34 |  -.0044308   .0235403    -0.19   0.851    -.0506041    .0417425
        /bx35 |   -.025656   .0265974    -0.96   0.335    -.0778256    .0265136
        /bx36 |  -.0905431    .041794    -2.17   0.030    -.1725202   -.0085659
        /bx37 |  -.0877903   .0336831    -2.61   0.009    -.1538582   -.0217224
        /bx38 |  -.0342967   .0149775    -2.29   0.022    -.0636745   -.0049189
        /bx39 |  -.0775394    .021409    -3.62   0.000    -.1195322   -.0355465
        /bx40 |  -.2041916   .0497217    -4.11   0.000    -.3017186   -.1066646
        /bx41 |  -.0688421   .0392082    -1.76   0.079    -.1457472     .008063
        /bx42 |  -.0907906   .0339963    -2.67   0.008    -.1574729   -.0241083
        /bx43 |  -.0406236   .0157342    -2.58   0.010    -.0714856   -.0097615
        /bx44 |  -.0208151   .0495645    -0.42   0.675    -.1180337    .0764034
        /bx45 |   .2448088   .0350566     6.98   0.000     .1760467    .3135708
        /bx46 |   .1490487   .0324907     4.59   0.000     .0853196    .2127778
        /bx47 |   .0114182   .0467025     0.24   0.807    -.0801868    .1030232
        /bx48 |   .0434947   .0183388     2.37   0.018     .0075239    .0794655
        /bx49 |   .0893845   .0352678     2.53   0.011     .0202084    .1585607
        /bx50 |   .0630281   .0354356     1.78   0.075    -.0064774    .1325336
        /bx51 |  -.0098313   .0246677    -0.40   0.690    -.0582159    .0385533
       /cle10 |   .4846053   .3269597     1.48   0.138    -.1567121    1.125923
          /c0 |   11.84317    .213595    55.45   0.000     11.42421    12.26213
         /mu2 |  -.7555186   .2751771    -2.75   0.006    -1.295267   -.2157705
         /mu6 |  -34.66291   605.0691    -0.06   0.954     -1221.48    1152.154
         /mu7 |  -.3772773    .140633    -2.68   0.007     -.653123   -.1014316
------------------------------------------------------------------------------
  Parameter c0 taken as constant term in model
```

Results for Column 4 of Table 5 (and for Tables 3 and 4)

```
Iteration 36:  residual SS =  33.29318

Nonlinear regression                                  Number of obs =      1665
                                                      R-squared     =    0.7169
                                                      Adj R-squared =    0.7033
                                                      Root MSE      =  .1447946
                                                      Res. dev.     = -1788.794

------------------------------------------------------------------------------
             |               Robust HC3
   lbidprice |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       /sn11 |   .6031816    .505708     1.19   0.233    -.3887438    1.595107
       /sn12 |  -2.510102   4.546923    -0.55   0.581     -11.4287    6.408501
      /cle11 |   .3907792   .0229345    17.04   0.000      .345794    .4357643
      /cle12 |    .297943   .0875316     3.40   0.001     .1262534    .4696325
       /sn21 |    .216555   .0341955     6.33   0.000      .149482    .2836281
       /sn22 |   1.325458   .5366366     2.47   0.014     .2728673    2.378049
       /sn61 |   .0136188   .0543641     0.25   0.802    -.0930141    .1202517
       /sn62 |   2.655585   1.419265     1.87   0.062    -.1282452    5.439416
       /sn71 |   .0013613   .0075753     0.18   0.857    -.0134972    .0162199
       /sn72 |   3.992441   3.452935     1.16   0.248    -2.780349    10.76523
       /sn31 |   1.191284   .3923636     3.04   0.002     .4216784    1.960889
       /sn32 |  -1.828277   .6997512    -2.61   0.009     -3.20081   -.4557437
       /sn41 |   .3260223   .2169209     1.50   0.133    -.0994591    .7515037
       /sn42 |  -.7161022   .3921232    -1.83   0.068    -1.485236    .0530313
        /sc1 |  -.0265209   .0071656    -3.70   0.000    -.0405759    -.012466
        /sc2 |   .0003952   .0001532     2.58   0.010     .0000946    .0006957
      /alpha |   1.881785   5.160718     0.36   0.715    -8.240751    12.00432
      /beta1 |   3.833446   1.296835     2.96   0.003     1.289757    6.377135
      /beta2 |  -1.964674   1.389582    -1.41   0.158    -4.690281    .7609334
      /beta3 |   4.087357   2.114608     1.93   0.053    -.0603596    8.235073
        /bx1 |  -.0421904   .0352832    -1.20   0.232     -.111397    .0270162
        /bx2 |   .0672725   .0385251     1.75   0.081    -.0082928    .1428378
        /bx3 |   .0699544   .0238718     2.93   0.003      .023131    .1167779
        /bx4 |  -.0296854   .0191631    -1.55   0.122     -.067273    .0079022
        /bx5 |   .0290689   .0065776     4.42   0.000     .0161672    .0419707
        /bx6 |  -.0013266   .0003923    -3.38   0.001    -.0020961   -.0005571
        /bx7 |  -.0309149   .0439278    -0.70   0.482    -.1170774    .0552477
        /bx8 |  -.0032753   .0246144    -0.13   0.894    -.0515555     .045005
        /bx9 |   .0291627   .0328998     0.89   0.376    -.0353688    .0936943
       /bx10 |   -.001672   .0037704    -0.44   0.658    -.0090674    .0057235
       /bx11 |  -.0132538   .0220059    -0.60   0.547    -.0564174    .0299098
       /bx12 |  -.0167093    .010785    -1.55   0.122    -.0378637     .004445
       /bx13 |   .0208803   .0229443     0.91   0.363     -.024124    .0658846
       /bx14 |   -.003125   .0050034    -0.62   0.532    -.0129388    .0066889
       /bx15 |  -.0606635   .0169242    -3.58   0.000    -.0938596   -.0274673
       /bx16 |   .0749098   .0229193     3.27   0.001     .0299546     .119865
       /bx17 |  -.2330509   .0701275    -3.32   0.001    -.3706031   -.0954987
       /bx18 |   .0080382    .003931     2.04   0.041     .0003276    .0157487
       /bx19 |   .0189241   .0081145     2.33   0.020      .003008    .0348403
       /bx20 |   .0122783   .0116159     1.06   0.291    -.0105057    .0350624
```

/bx21	-.0160812	.0094383	-1.70	0.089	-.0345941	.0024318
/bx22	.011744	.009781	1.20	0.230	-.0074409	.030929
/bx23	.0295355	.0218962	1.35	0.178	-.0134129	.072484
/bx24	.0036634	.0107876	0.34	0.734	-.017496	.0248227
/bx25	-.0305338	.0103662	-2.95	0.003	-.0508666	-.0102009
/bx26	.0071861	.0182925	0.39	0.694	-.0286939	.043066
/bx27	-.0000209	4.88e-06	-4.28	0.000	-.0000305	-.0000113
/bx28	5.69e-06	1.33e-06	4.26	0.000	3.07e-06	8.30e-06
/bx29	-4.89e-07	1.14e-07	-4.28	0.000	-7.12e-07	-2.65e-07
/bx30	7.69e-09	1.79e-09	4.29	0.000	4.17e-09	1.12e-08
/bx31	-.1055739	.0492504	-2.14	0.032	-.2021765	-.0089713
/bx32	-.1118011	.0430834	-2.59	0.010	-.1963075	-.0272947
/bx33	-.175793	.045165	-3.89	0.000	-.2643822	-.0872038
/bx34	-.0039829	.0236882	-0.17	0.866	-.0504464	.0424806
/bx35	-.0256398	.0266869	-0.96	0.337	-.077985	.0267054
/bx36	-.0909367	.0420018	-2.17	0.031	-.1733215	-.0085519
/bx37	-.0868714	.0333464	-2.61	0.009	-.1522789	-.0214639
/bx38	-.034056	.0149365	-2.28	0.023	-.0633532	-.0047587
/bx39	-.0773373	.0214325	-3.61	0.000	-.1193763	-.0352982
/bx40	-.2030268	.050017	-4.06	0.000	-.3011331	-.1049204
/bx41	-.0689017	.0391674	-1.76	0.079	-.145727	.0079237
/bx42	-.0908243	.0341056	-2.66	0.008	-.1577211	-.0239274
/bx43	-.0409746	.0157622	-2.60	0.009	-.0718915	-.0100577
/bx44	-.0199302	.0497115	-0.40	0.689	-.1174373	.0775769
/bx45	.2448414	.0353108	6.93	0.000	.1755806	.3141022
/bx46	.1492612	.0324552	4.60	0.000	.0856017	.2129208
/bx47	.0121942	.0467372	0.26	0.794	-.079479	.1038674
/bx48	.043722	.0183819	2.38	0.017	.0076666	.0797774
/bx49	.088863	.0353591	2.51	0.012	.0195076	.1582185
/bx50	.0630554	.0354225	1.78	0.075	-.0064244	.1325352
/bx51	-.0095357	.0246419	-0.39	0.699	-.0578697	.0387983
/cle10	.4755413	.3269452	1.45	0.146	-.1657482	1.116831
/c0	12.30751	.2839808	43.34	0.000	11.7505	12.86453
/mu2	-.7511464	.2704338	-2.78	0.006	-1.281591	-.2207015
/mu6	-.8914684	.529977	-1.68	0.093	-1.930997	.1480598
/mu7	-.627294	.1557798	-4.03	0.000	-.9328498	-.3217382
/star6	.2554039	.1193368	2.14	0.032	.0213297	.4894781
/star7	.5277194	.0330362	15.97	0.000	.4629202	.5925187

Parameter c0 taken as constant term in model

Results for Table 7

Note: lslopesi = log(psii) where the last letter indicates the amenity

Indirect Tests

```
Linear regression                                       Number of obs =     142
                                                        F(  1,   140) =    9.33
                                                        Prob > F      =  0.0027
                                                        R-squared     =  0.0849
                                                        Root MSE      =  1.0832

------------------------------------------------------------------------------
             |             Robust HC3
    lslopes1 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lowninc |   1.769869   .5792956     3.06   0.003     .6245705    2.915167
       _cons |   -19.7342    5.97656    -3.30   0.001    -31.55018   -7.918219
------------------------------------------------------------------------------


Linear regression                                       Number of obs =    1113
                                                        F(  1,  1111) =  326.39
                                                        Prob > F      =  0.0000
                                                        R-squared     =  0.2028
                                                        Root MSE      =  .62614

------------------------------------------------------------------------------
             |             Robust HC3
    lslopes2 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lowninc |   1.018914   .0563985    18.07   0.000     .9082546    1.129574
       _cons |  -12.77014   .6020148   -21.21   0.000    -13.95135   -11.58892
------------------------------------------------------------------------------


Linear regression                                       Number of obs =    1417
                                                        F(  1,  1415) =   53.12
                                                        Prob > F      =  0.0000
                                                        R-squared     =  0.0280
                                                        Root MSE      =  .69881

------------------------------------------------------------------------------
             |             Robust HC3
    lslopes6 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lowninc |   .3795374   .0520723     7.29   0.000     .2773902    .4816846
       _cons |  -5.873054   .5589047   -10.51   0.000    -6.969425   -4.776684
------------------------------------------------------------------------------
```

```
Linear regression                                       Number of obs =    1649
                                                        F(  1,  1647) =   63.18
                                                        Prob > F      =  0.0000
                                                        R-squared     =  0.0362
                                                        Root MSE      =  .41553

------------------------------------------------------------------------------
             |               Robust HC3
    lslopes7 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lowninc |   .2514914   .0316388     7.95   0.000     .1894349     .313548
       _cons |  -5.584744   .3398238   -16.43   0.000    -6.251276   -4.918212
------------------------------------------------------------------------------
```

Direct Tests

```
Linear regression                                       Number of obs =     142
                                                        F( 11,   130) =    4.23
                                                        Prob > F      =  0.0000
                                                        R-squared     =  0.2236
                                                        Root MSE      =  1.0354

------------------------------------------------------------------------------
             |               Robust HC3
    lslopes1 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
     lowninc |   1.354031   .8058643     1.68   0.095    -.2402749    2.948337
  Pct65pls_cbg |   .0413486   .0177043     2.34   0.021     .0063228    .0763744
   Pctkids_cbg |   .0119116   .0131655     0.90   0.367    -.0141347    .0379579
    Pctmar_cbg |   .0077418   .0106165     0.73   0.467    -.0132617    .0287452
Pctenglish_cbg |  -.0152121    .007618    -2.00   0.048    -.0302834   -.0001408
    pctapi_cbg |   .0120307   .0407895     0.29   0.769    -.0686664    .0927278
     pctlt1_ct |   .0126461   .0159353     0.79   0.429    -.0188799    .0441721
     pcths_cbg |  -.0240751   .0137234    -1.75   0.082    -.0512253     .003075
  pctlesba_cbg |   .0067049   .0164709     0.41   0.685    -.0258808    .0392906
     pctba_cbg |   .0241524   .0197228     1.22   0.223    -.0148668    .0631716
Pctgraddeg_cbg |  -.0020355   .0405548    -0.05   0.960    -.0822684    .0781974
         _cons |  -15.07579   8.384565    -1.80   0.074    -31.66365    1.512071
------------------------------------------------------------------------------
```

```
Linear regression                               Number of obs =    1113
                                                F( 11,  1101) =   58.84
                                                Prob > F      =  0.0000
                                                R-squared     =  0.3026
                                                Root MSE      =  .58831

------------------------------------------------------------------------------
             |               Robust HC3
     lslopes2 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
--------------+---------------------------------------------------------------
      lowninc |    .642606   .0905614     7.10   0.000     .4649136    .8202984
 Pct65pls_cbg |   .0163823    .003375     4.85   0.000     .0097602    .0230044
  Pctkids_cbg |   .0077452   .0028468     2.72   0.007     .0021595     .013331
   Pctmar_cbg |   .0047941   .0020076     2.39   0.017     .0008549    .0087333
Pctenglish_cbg |   -.013009   .0030241    -4.30   0.000    -.0189427   -.0070753
   pctapi_cbg |   .0184615   .0084072     2.20   0.028     .0019657    .0349574
    pctlt1_ct |  -.0061625   .0034384    -1.79   0.073     -.012909     .000584
    pcths_cbg |   .0118403    .004352     2.72   0.007     .0033011    .0203795
 pctlesba_cbg |   .0153781   .0046916     3.28   0.001     .0061727    .0245835
    pctba_cbg |   .0271826   .0034974     7.77   0.000     .0203203    .0340449
Pctgraddeg_cbg |   .0047813   .0042711     1.12   0.263    -.0035991    .0131617
         _cons |  -9.634362   .8904272   -10.82   0.000    -11.38149   -7.887236
------------------------------------------------------------------------------

Linear regression                               Number of obs =    1417
                                                F( 11,  1405) =   19.28
                                                Prob > F      =  0.0000
                                                R-squared     =  0.2171
                                                Root MSE      =  .62941

------------------------------------------------------------------------------
             |               Robust HC3
     lslopes6 |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
--------------+---------------------------------------------------------------
      lowninc |   .4575399   .1052404     4.35   0.000     .2510946    .6639852
 Pct65pls_cbg |  -.0052623   .0031863    -1.65   0.099    -.0115126    .0009881
  Pctkids_cbg |  -.0166856   .0030911    -5.40   0.000    -.0227492   -.0106219
   Pctmar_cbg |   .0232675   .0029383     7.92   0.000     .0175035    .0290314
Pctenglish_cbg |  -.0156733   .0030081    -5.21   0.000    -.0215741   -.0097726
   pctapi_cbg |  -.0054632   .0063529    -0.86   0.390    -.0179254    .0069989
    pctlt1_ct |   .0021304   .0033924     0.63   0.530    -.0045243    .0087851
    pcths_cbg |   .0159899   .0041367     3.87   0.000     .0078752    .0241046
 pctlesba_cbg |  -.0042782   .0046387    -0.92   0.357    -.0133777    .0048213
    pctba_cbg |   .0090331   .0036028     2.51   0.012     .0019656    .0161006
Pctgraddeg_cbg |  -.0098069   .0040455    -2.42   0.015    -.0177428    -.001871
         _cons |  -6.398852    1.05783    -6.05   0.000    -8.473949   -4.323755
------------------------------------------------------------------------------
```

Linear regression

Number of obs	1649
F(11, 1637)	23.45
Prob > F	0.0000
R-squared	0.4335
Root MSE	.31953

lslopes7	Coef.	Robust HC3 Std. Err.	t	P>|t|	[95% Conf.	Interval]
lowninc	.0392968	.0432796	0.91	0.364	-.0455924	.124186
Pct65pls_cbg	.0070968	.0013891	5.11	0.000	.0043722	.0098214
Pctkids_cbg	-.0010322	.0015815	-0.65	0.514	-.0041342	.0020698
Pctmar_cbg	-.0018698	.0009592	-1.95	0.051	-.0037512	.0000117
Pctenglish_cbg	.0273133	.0042532	6.42	0.000	.018971	.0356556
pctapi_cbg	.0253509	.0063438	4.00	0.000	.0129081	.0377937
pctlt1_ct	-.0071491	.001719	-4.16	0.000	-.0105207	-.0037774
pcths_cbg	.0053497	.0026092	2.05	0.040	.0002321	.0104673
pctlesba_cbg	.0058534	.0020891	2.80	0.005	.0017559	.0099509
pctba_cbg	.0089773	.0015475	5.80	0.000	.0059421	.0120125
Pctgraddeg_cbg	.0044425	.0020589	2.16	0.031	.0004042	.0084807
_cons	-6.229191	.7681105	-8.11	0.000	-7.735773	-4.722608