Housing and Transportation Affordability Initiative

Understanding the combined cost of housing and transportation on affordability.

Modeling Code

Location Affordability Index

Version 2.0

Contents

Introduction

This paper provides a summary analysis of the primary issues encountered with the development of the models for the Location Affordability Index Version 2 (LAI V2). Also included are appendices providing data definitions, Structured Query Language (SQL), PHP and R programs developed for this effort.

Data Sources

Data comes from several sources. The criteria for using data are that it be ubiquitously available for the entire United States, and be generated by a federal source.

American Community Survey

Most of the data for the LAI comes from the 2008-2012 American Community Survey (ACS) 5-year estimates. Table 2 in Appendix A: ACS Data Inputs shows the ACS variables used, and how they are defined for the LAI models. In order to put these variables into the LAI database the entire 2008-2012 ACS 5-year estimates were downloaded[1] and states were merged into a set of 119 tables containing data for all geographies (block groups, tracts, places, counties, etc.). These variables are stored in a database table which is accessed for the regression analysis, as well as for the final website. Program 1 in Appendix A: ACS Data Inputs is the PHP script used to create this input table.

Longitudinal Employer-Household Dynamics

There are five independent variables that depend on employment information:

1. median distance for journey to work;
2. employment intensity (or gravity);
3. retail employment intensity (or gravity);
4. local employment density; and
5. local retail employment density.

The main source for these employment data is the Longitudinal Employer-Household Dynamic (LEHD) data set. For each block group, data on number of employees, type of employment and where employees came from were developed. The raw Longitudinal Employer-Household Dynamic Origin Destination (LODES) data was downloaded from the “On the Map” LEHD website.[2] The precise version used was version 6 which included data for states except Massachusetts, and used the 2010 tiger census geographies (this download interface is no longer available).

Building the employment intensity and employment diversity indices requires aggregating employees to Block Groups. This was done with a simple SQL query that summed the employees by each employment category from the worker table (wac_jt00_2010) using all jobs. This forms table “blkgrps_2010_led.” See

[1] Source: http://www2.census.gov/acs2010_5yr/summaryfile/2008-2012_ACSSF_By_State_All_Tables/

[2] Source: http://lehd.did.census.gov/data/#lodes. Note: The user interface used for the raw data download is no longer available.

Program 2 in Appendix B: Employment Data for the complete script in how the employment gravity measures are created.

The distance workers travel to get to work is also measured in the LODES data. Using the origin destination tables, the median and the mean of the distance traveled by block group are determined and stored in a table for use in the regression analysis.

Massachusetts Employment Data

Unfortunately, the state of Massachusetts has no data in the LODES data base. As noted on the LEHD website[3]:

> "All 50 states, the District of Columbia, Puerto Rico, and the U.S. Virgin Islands have joined the LED Partnership, although the LEHD program is not yet producing public-use statistics for Massachusetts, Puerto Rico, or the U.S. Virgin Islands."

Consequently, it was necessary to develope methods to fill in the employment data needed for the regression analysis.

The 2000 Census Transportation Planning Products (CTPP) data was used to determine the number of employees and their industry type. Employment by county was obtained for 2010 using the Massachusetts ES202 database query tool.[4] Using a constant share method from the 2000 CTPP employment data at the block group level, an estimate of 2010 employment was made for every block group in Massachusetts. See Program 4 and Program 5 in Appendix B: Employment Data for the code utilized.

Illinois Odometer Readings

The Illinois Department of Natural Resources collects data on odometer reading for autos as they complete their Vehicle Emissions Testing Program. These data were obtained for years surrounding 2010. The data shared contains the following:

- "VIN": The unique Vehicle Identification Number, available for every vehicle in the world
- "ZIPPLUS4": 5 digit zip code plus 4 (Zip+4) additional digits that further define location
- "ODOMETER": Odometer reading at time of test rounded to the 1,000th place
- "TEST_DATE_AND_TIME": Date on which the test was performed
- "MAKE": Make of the vehicle
- "MODEL": Model of vehicle
- "MY": Model Year of Vehicle
- "PURCHASE_DATE": Purchase data of vehicle

[3] See http://lehd.ces.census.gov/ for more information.

[4] Source: (http://lmi2.detma.org/lmi/lmi_es_a.asp.

By linking these two data sets and requiring that the VIN and the 5-digit zip codes are the same, two unique readings of odometer for every vehicle is determined. These data are then aggregated to the block groups using a Zip+4 to Census block group lookup table.

Making Data to Fit

A view in PostgreSQL, with all the information required to fit every model, is made to make one overall data file. See Program 6 in Appendix C: Creating Data for Fitting.

A set of tools that automatically creates SQL and R code for fitting was developed which allows the user to examine the residual distributions against all the independent variables, and manage output such as the coefficients and residuals.

Creating R Readable Data

Data is extracted from the larger data set used by the R statistics package for the simultaneous (or structural) equation model (SEM) fit and the vehicle mile traveled (VMT) fit. The fitting program creates SQL that makes a data table in the schemes for each fit. See Program 7 in Appendix C: Creating Data for Fitting for the two SQL queries that create the data tables for the SEM and the VMT fit.

Linear Transformations

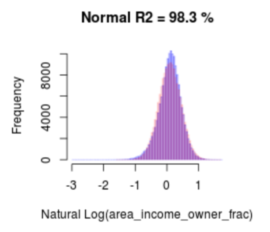
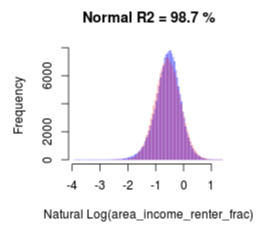
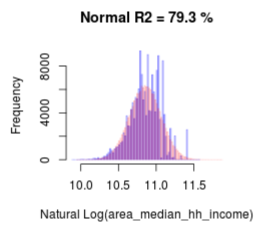
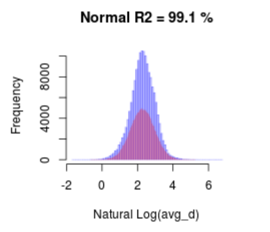
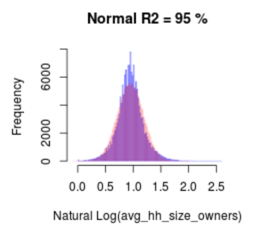
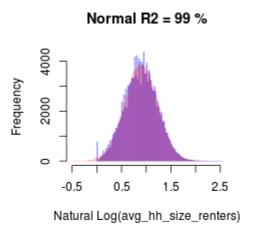
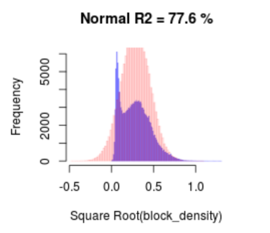
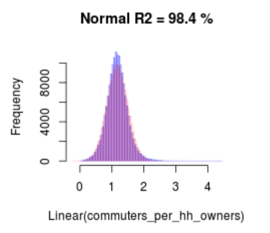
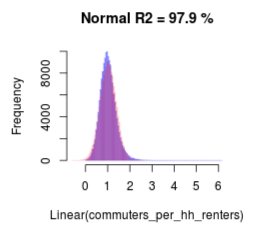
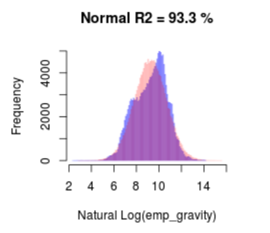
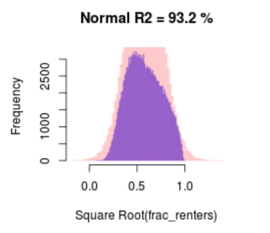
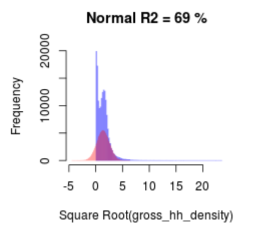
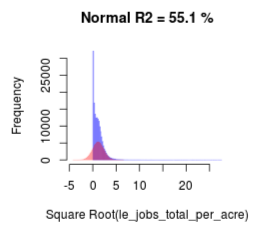
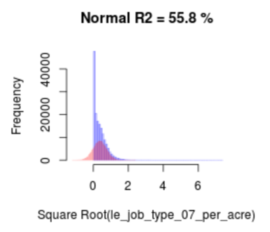
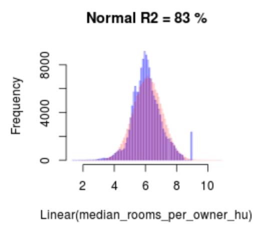
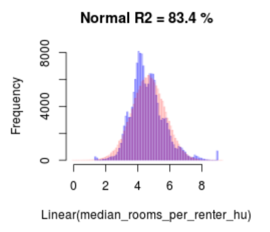
The variables, once read in by R, need to be transformed into their linearized form. See Program 8 in Appendix C: Creating Data for Fitting for the code that transforms and “un-transforms” these variables. Recall that the three possible transformations are

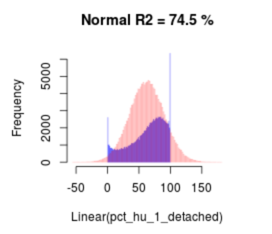
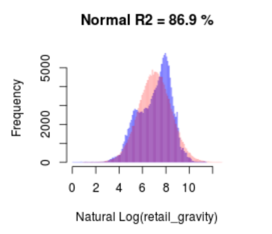
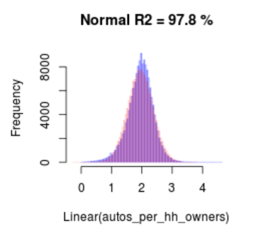
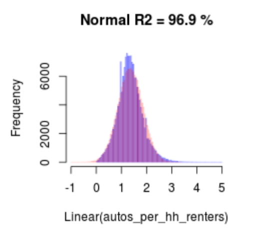
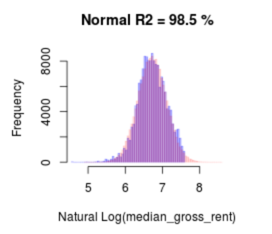
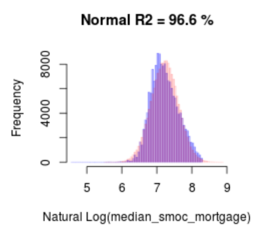
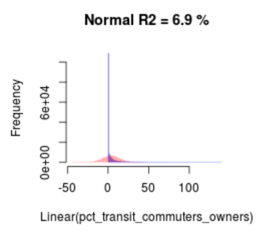
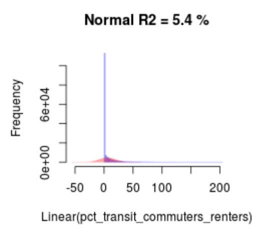
- Linear
- Square Root
- Natural Log

Choosing Transformation

Every independent and dependent variable was examined to decide what transformation was most appropriate. The following table shows the results for all of the variables used in the SEM fit. For the VMT the same transformations were used; the VMT variable itself used no transformation (linear).

Table 1: Distributions of SEM Variables over All Census Block Groups

Area Income Fraction Owners	Area Income Fraction Renters	Area Median Income	Median J2W Miles
			
HH Size Owner	HH Size renter	Block Density	Commuters/HH Owners
			
Commuters/HH Renters	Employment Access	Fraction Rental Units	Gross HH Density
			
Local Job Density	Local Retail Jobs per acre	Median Rooms/Owner HU	Median Rooms/Renter HU
			

Fraction Single Detached HU	Retail Gravity	Autos/HH Owners	Autos/HH Renters
			
Gross Rent	SMOC	Transit %J2W Owners	Transit %J2W renters
			

Fitting

The regression was completed using Ordinary Least Squares (OLS), with a functional form known as the Second Order Flexible Form for fitting the VMT model and SEM for fitting the rest.

Choosing Significant Variables for VMT Model

Because there is an inherent spatial autocorrelation for the dependent variables, a robust variance calculation was employed to estimate the statistical significance of the regression coefficients. The method for estimating the error on the coefficients used geographical clustering. Three natural geographical clustering definitions were tested: state, county and CBSA. The testing showed that the error estimate increased (as expected) when using this robust approach, and that the state clustering increased the error estimate the least, with the county and CBSA clustering having a similar estimates; therefore CBSA clustering was employed. This method results in a better estimate of the probability that a coefficient is insignificant (often referred to as the p-value). The criterion used to include a coefficient in the final model is that the p-value is less that 0.05 or 5 percent; this is a very commonly used criterion in regression analysis.

There is a high probability that the independent variables are multi-collinear. To eliminate as much of this as possible, the variance inflation factor (VIF)[5] was examined. After eliminating the coefficients with high p-value, the VIF is required to be less than 20. This is somewhat higher than commonly used; often the criteria are set at 10. However, for this analysis the values tended to be greater than 10,000 to begin with, and drop perceptibly as the highly multi-collinear coefficients are excluded.

[5] For a definition of VIF see http://en.wikipedia.org/wiki/Variance_inflation_factor.

Program 10 in Appendix D: Choosing Variables and Interaction Terms show how this was done. The method is to first fit all the terms, square terms and interaction terms, then find the one with the largest error, and see if it passes the established criteria. If it does not, that term is not used, and the terms are fit again. This sequence is repeated until there are no terms with too large of an error (which is also too large of P(|t|) < 0.05). The VIF of each fit coefficient is then examined; the largest to fit within the criteria (VIF<20) is required. This is completed without using the clustering algorithm that will estimate the errors. The clustering algorithm (see Program 12, Appendix E: Final Fits) is then employed to estimate the errors better and perform the same analysis. This step is done last because adding the clustering adds quite a bit of CPU time to the calculations. Variables are then stored in the database and final fits run.

Final Fits

The final fits are run once the significant terms are determined for the VMT model and the interactions determined for the SEM fit. Each model has a different code generated to complete this final fit. The R code contains many lines that create diagnostic plots and graphs and are all included in

Appendix E: Final Fits. Finally Appendix F: Running Models and Calculation Costs contain the code used to run the final model and calculate the costs and the LAI.

Appendix A: ACS Data Inputs

Table 2 below shows the variables that are imported into the working database table (acs_bg_2012_data). The columns represent:

- var_name working name for the variable used in the LAI
- acs_field_ name from the ACS for that variable, note that this can also be a sum
- description description of the variable
- tab_order order in which the date in the table – of no real consequence

There are a few variables in Table 2 that were not used in the final models (example - Median Year Structure Build Owner). Although these variables were tested, they were not used in the final models, or they were used in other maps or tables during research.

Table 2: Raw ACS Variable Definition

var_name	acs_field_name	description	tab_order
Population	b01003_001	Total Population	1
Households	b11012_001	Total Number of Households	2
hh_population	b11002_001	Population in Households	3
population_occ_hu	b25008_001	Population in Occupied Housing Units	4
population_owner_occ_hu	b25008_002	Population in Owner Occupied Housing Units	5
population_renter_occ_hu	b25008_003	Population in Renter Occupied Housing Units	6
avg_hh_size	b25010_001	Average HH Size from ACS	8
avg_hh_size_all	b25010_001	Average HH Size both Renters and Owners	9
avg_hh_size_owners	b25010_002	Average HH Size Owners	10
avg_hh_size_renters	b25010_003	Average HH Size Renters	11
median_hh_income	b19049_001	Median Household Income	12
median_hh_income_owners	b25119_002	Median Household Income for Owners	13
median_hh_income_renters	b25119_003	Median Household Income for Renters	14
agg_hh_income	b19050_001	Aggregate Household Income	16

var_name	acs_field_name	description	tab_order
agg_hh_vehicles	b25046_001	Aggregate HH Vehicles	20
agg_hh_vehicles_owners	b25046_002	Aggregate owners vehicles	21
agg_hh_vehicles_renters	b25046_003	Aggregate Autos Available to Renters	22
hh_workers	b99085_001	Total Workers in Households	30
total_workers	b08301_001	All Workers	31
workers_car	b08301_002	Workers Using Autos	32
workers_car_one	b08301_003	Workers Using Auto Drive Alone	33
workers_carpooled	b08301_004	Workers Using Carpools	34
workers_transit	b08301_010	Workers Using Transit	40
workers_bus	b08301_011	Workers Using Bus	41
workers_streetcar	b08301_012	Workers Using Streetcar	42
workers_subway	b08301_013	Workers Using Subway or Elevated	43
workers_rr	b08301_014	Workers Using Railroad	44
workers_ferry	b08301_015	Workers Using Ferry	45
workers_taxi	b08301_016	Workers Using Taxi	46
workers_motorcycle	b08301_017	Workers Using Motorcycle	47
workers_bike	b08301_018	Workers Using Bicycle	48
workers_walked	b08301_019	Workers Using Walk	49
workers_other	b08301_020	Workers Using Other	50
workers_home	b08301_021	Workers Stay at Home	51
median_age	b01002_001	Median Age of Population	60
pop_under_5	b01001_003+b01001_027	Population under 5 years old	61
pop_5_9	b01001_004+b01001_028	Population from 5 to 9	62
pop_10_14	b01001_005+b01001_029	Population from 10 to 14	63
pop_15_17	b01001_006+b01001_030	Population from 15 to 17	64
pop_18_19	b01001_007+b01001_031	18 and 19 years	65
pop_20	b01001_008+b01001_032	20 years	66

var_name	acs_field_name	description	tab_order
pop_21	b01001_009+b01001_033	21 years	67
pop_22_24	b01001_010+b01001_034	22 to 24 years	68
pop_25_29	b01001_011+b01001_035	25 to 29 years	69
pop_30_34	b01001_012+b01001_036	30 to 34 years	70
pop_35_39	b01001_013+b01001_037	35 to 39 years	71
pop_40_44	b01001_014+b01001_038	40 to 44 years	72
pop_45_49	b01001_015+b01001_039	45 to 49 years	73
pop_50_54	b01001_016+b01001_040	50 to 54 years	74
pop_55_59	b01001_017+b01001_041	55 to 59 years	75
pop_60_61	b01001_018+b01001_042	60 and 61 years	76
pop_62_64	b01001_019+b01001_043	62 to 64 years	77
pop_65_66	b01001_020+b01001_044	65 and 66 years	78
pop_67_69	b01001_021+b01001_045	67 to 69 years	79
pop_70_74	b01001_022+b01001_046	70 to 74 years	80
pop_75_79	b01001_023+b01001_047	75 to 79 years	81
pop_80_84	b01001_024+b01001_048	80 to 84 years	82
pop_80_over	b01001_025+b01001_049	85 years and over	83
housing_units	b25002_001	Total Housing Units	110
occupied_hu	b25002_002	Occupied HU	111
vacant_hu	b25002_003	Vacant HU	112
owner_occupied_hu	b25003_002	Owner Occupied HU	113
renter_occupied_hu	b25003_003	Renter Occupied HU	114
owner_hu	b25087_001	Owner Housing Units from Mortgage Status	117
owner_hu_with_mortgage	b25087_002	Owner Housing Units with Mortgage from Mortgage Status	118
owner_hu_without_mortgage	b25087_018	Owner Housing Units without Mortgage from Mortgage Status	119
median_smoc	b25088_001	Median SMOC	120

var_name	acs_field_name	description	tab_order
median_smoc_mortgage	b25088_002	Medina SMOC with Mortgage	121
median_smoc_no_mortgage	b25088_003	Median SMOC without Mortgage	122
median_gross_rent	b25064_001	Median Gross Rent	123
median_yr_structure_build	b25035_001	Median Year Structure Build	130
renter_hu	b25063_001	Renter HU from Gross Rent Total	140
renter_cash_hu	b25063_002	Renter HU from Gross Rent Paying With Cash	141
renter_nocash_hu	b25063_024	Renter HU from Gross Rent Not Paying With Cash	142
hh_population_renters	b25008_003	Population of Renters	203
renter_workers_total	b08137_003	Total Workers by Tenure - Renters	231
agg_time_j2w_commuters	b08135_001	Aggregate Time J2W for workers not working at home	300
commuters_total	b08303_001	Total Workers not working at home	301
commuters_j2wt_lt_5min	b08303_002	Workers not working at home who commute less than 5 minutes	302
commuters_j2wt_5_to_9min	b08303_003	Workers not working at home who commute 5 to 9 minutes	303
commuters_j2wt_10_to_14min	b08303_004	Workers not working at home who commute 10 to 14 minutes	304
commuters_j2wt_15_to_19min	b08303_005	Workers not working at home who commute 15 to 19 minutes	305
commuters_j2wt_20_to_24min	b08303_006	Workers not working at home who commute 20 to 24 minutes	306
commuters_j2wt_25_to_29min	b08303_007	Workers not working at home who commute 25 to 29 minutes	307
commuters_j2wt_30_to_34min	b08303_008	Workers not working at home who commute 30 to 34 minutes	308
commuters_j2wt_35_to_39min	b08303_009	Workers not working at home who commute 35 to 39 minutes	309
commuters_j2wt_40_to_44min	b08303_010	Workers not working at home who commute 40 to 44 minutes	310

var_name	acs_field_name	description	tab_order
commuters_j2wt_45_to_59min	b08303_011	Workers not working at home who commute 45 to 59 minutes	311
commuters_j2wt_60_to_89min	b08303_012	Workers not working at home who commute 60 to 89 minutes	312
commuters_j2wt_90min_plus	b08303_013	Workers not working at home who commute 90 minutes plus	313
tenure_workers_total	b08137_001	Total Workers by Tenure	500
owner_workers_total	b08137_002	Total Workers by Tenure - Owners	501
tenure_workers_sov	b08137_004	SOV Workers by Tenure	503
owner_workers_sov	b08137_005	SOV Workers by Tenure - Owners	504
renter_workers_sov	b08137_006	SOV Workers by Tenure - Renters	505
tenure_workers_carpool	b08137_007	Carpool Workers by Tenure	506
owner_workers_carpool	b08137_008	Carpool Workers by Tenure - Owners	507
renter_workers_carpool	b08137_009	Carpool Workers by Tenure - Renters	508
tenure_workers_transit	b08137_010	Transit Workers by Tenure	509
owner_workers_transit	b08137_011	Transit Workers by Tenure - Owners	510
renter_workers_transit	b08137_012	Transit Workers by Tenure - Renters	511
tenure_workers_walk	b08137_013	Walking Workers by Tenure	512
owner_workers_walk	b08137_014	Walking Workers by Tenure - Owners	513
renter_workers_walk	b08137_015	Walking Workers by Tenure - Renters	514
tenure_workers_tmbo	b08137_016	Taxi, Motorcycle, Bicycle or Other Workers by Tenure	515
owner_workers_tmbo	b08137_017	Taxi, Motorcycle, Bicycle or Other Workers by Tenure - Owners	516
renter_workers_tmbo	b08137_018	Taxi, Motorcycle, Bicycle or Other Workers by Tenure - Renters	517
tenure_workers_home	b08137_019	Home Workers by Tenure	518
owner_workers_home	b08137_020	Home Workers by Tenure - Owners	519
renter_workers_home	b08137_021	Home Workers by Tenure - Renters	520

var_name	acs_field_name	description	tab_order
gr_universe	b25063_001	Gross Rent Universe	601
gr_cash_rent	b25063_002	Gross Rent Paying Cash Universe	602
gr_000_100	b25063_003	Gross Rent Less than 100	603
gr_100_149	b25063_004	Gross Rent 100 to 149	604
gr_150_199	b25063_005	Gross Rent 150 to 199	605
gr_200_249	b25063_006	Gross Rent 200 to 249	606
gr_250_299	b25063_007	Gross Rent 250 to 249	607
gr_300_349	b25063_008	Gross Rent 300 to 349	608
gr_350_399	b25063_009	Gross Rent 350 to 399	609
gr_400_449	b25063_010	Gross Rent 400 to 449	610
gr_450_499	b25063_011	Gross Rent 450 to 499	611
gr_500_549	b25063_012	Gross Rent 500 to 549	612
gr_550_599	b25063_013	Gross Rent 550 to 599	613
gr_600_649	b25063_014	Gross Rent 600 to 649	614
gr_650_699	b25063_015	Gross Rent 650 to 699	615
gr_700_749	b25063_016	Gross Rent 700 to 749	616
gr_750_799	b25063_017	Gross Rent 750 to 799	617
gr_800_899	b25063_018	Gross Rent 800 to 899	618
gr_900_999	b25063_019	Gross Rent 900 to 999	619
gr_1000_1249	b25063_020	Gross Rent 1000 to 1249	620
gr_1250_1499	b25063_021	Gross Rent 1250 to 1499	621
gr_1500_1999	b25063_022	Gross Rent 1500 to 1999	622
gr_2000_plus	b25063_023	Gross Rent 2000 Plus	623
gr_nocash	b25063_024	Gross Rent No Cash	624
smoc_universe	b25087_001	SMOC Universe	625
smoc_hu_mort	b25087_002	SMOC With Mortgage Universe	626
smoc_mort_000_200	b25087_003	SMOC With Mortgage less than 200	627

var_name	acs_field_name	description	tab_order
smoc_mort_200_299	b25087_004	SMOC With Mortgage 200 to 299	628
smoc_mort_300_399	b25087_005	SMOC With Mortgage 300 to 399	629
smoc_mort_400_499	b25087_006	SMOC With Mortgage 400 to 499	630
smoc_mort_500_599	b25087_007	SMOC With Mortgage 500 to 599	631
smoc_mort_600_699	b25087_008	SMOC With Mortgage 600 to 699	632
smoc_mort_700_799	b25087_009	SMOC With Mortgage 700 to 799	633
smoc_mort_800_899	b25087_010	SMOC With Mortgage 800 to 899	634
smoc_mort_900_999	b25087_011	SMOC With Mortgage 900 to 999	635
smoc_mort_1000_1249	b25087_012	SMOC With Mortgage 1000 to 1249	636
smoc_mort_1250_1499	b25087_013	SMOC With Mortgage 1250 to 1499	637
smoc_mort_1500_1999	b25087_014	SMOC With Mortgage 1500 to 1999	638
smoc_mort_2000_2499	b25087_015	SMOC With Mortgage 2000 to 2499	639
smoc_mort_2500_2999	b25087_016	SMOC With Mortgage 2500 to 2999	640
smoc_mort_3000_plus	b25087_017	SMOC With Mortgage 3000 plus	641
smoc_nomort_hu	b25087_018	SMOC No Mortgage Housing Units	642
smoc_nomort_000_100	b25087_019	SMOC No Mortgage less than 100	643
smoc_nomort_100_149	b25087_020	SMOC No Mortgage 100 to 149	644
smoc_nomort_150_199	b25087_021	SMOC No Mortgage 150 to 199	645
smoc_nomort_200_249	b25087_022	SMOC No Mortgage 200 to 249	646
smoc_nomort_250_299	b25087_023	SMOC No Mortgage 250 to 299	647
smoc_nomort_300_349	b25087_024	SMOC No Mortgage 300 to 349	648
smoc_nomort_350_399	b25087_025	SMOC No Mortgage 350 to 399	649
smoc_nomort_400_499	b25087_026	SMOC No Mortgage 400 to 499	650
smoc_nomort_500_599	b25087_027	SMOC No Mortgage 500 to 599	651
smoc_nomort_600_699	b25087_028	SMOC No Mortgage 600 to 699	652
smoc_nomort_700_plus	b25087_029	SMOC No Mortgage 700 plus	653
income_hh	b19001_001	Total Households in Income	901

var_name	acs_field_name	description	tab_order
income_hh_lt_10k	b19001_002	Households with Income Less than $10,000	902
income_hh_10k_15k	b19001_003	Households with Income $10,000 to $14,999	903
income_hh_15k_20k	b19001_004	Households with Income $15,000 to $19,999	904
income_hh_20k_25k	b19001_005	Households with Income $20,000 to $24,999	905
income_hh_25k_30k	b19001_006	Households with Income $25,000 to $29,999	906
income_hh_20k_35k	b19001_007	Households with Income $30,000 to $34,999	907
income_hh_35k_40k	b19001_008	Households with Income $35,000 to $39,999	908
income_hh_40k_45k	b19001_009	Households with Income $40,000 to $44,999	909
income_hh_45k_50k	b19001_010	Households with Income $45,000 to $49,999	910
income_hh_50k_60k	b19001_011	Households with Income $50,000 to $59,999	911
income_hh_60k_75k	b19001_012	Households with Income $60,000 to $74,999	912
income_hh_75k_100k	b19001_013	Households with Income $75,000 to $99,999	913
income_hh_100k_125k	b19001_014	Households with Income $100,000 to $124,99	914
income_hh_125k_150k	b19001_015	Households with Income $125,000 to $149,999	915
income_hh_150k_200k	b19001_016	Households with Income $150,000 to $199,999	916
income_hh_100k_plus	b19001_017	Households with Income $200,000 or more	917
total_families	b17026_001	Total Families	921
families_in_poverty_50pct_less	b17026_002	Families in Poverty Less Than 50%	922
families_in_poverty_50_75pct	b17026_003	Families in Poverty 50% to 75%	923
families_in_poverty_75_100pct	b17026_004	Families in Poverty 75% to 100%	924
families_in_poverty_100_125pct	b17026_005	Families in Poverty 100% to 125%	925
families_in_poverty_125_150pct	b17026_006	Families in Poverty 125% to 150%	926
families_in_poverty_150_175pct	b17026_007	Families in Poverty 150% to 175%	927
families_in_poverty_175_185pct	b17026_008	Families in Poverty 175% to 185%	928
families_in_poverty_185_200pct	b17026_009	Families in Poverty 185% to 200%	929
families_in_poverty_200_300pct	b17026_010	Families in Poverty 200% to 300%	930
families_in_poverty_300_400pct	b17026_011	Families in Poverty 300% to 400%	931

var_name	acs_field_name	description	tab_order
families_in_poverty_400_500pct	b17026_012	Families in Poverty 400% to 500%	932
families_in_poverty_500pct_plus	b17026_013	Families in Poverty 500% or More	933
workers_by_tenure_total	b08137_001	Total	5001
workers_by_tenure_total_owner	b08137_002	Householder lived in owner-occupied housing units	5002
workers_by_tenure_total_renters	b08137_003	Householder lived in renter-occupied housing units	5003
workers_by_tenure_home	b08137_019	Worked at home	5019
workers_by_tenure_home_owner	b08137_020	Householder lived in owner-occupied housing units - Worked at home	5020
workers_by_tenure_home_renters	b08137_021	Householder lived in renter-occupied housing units - Worked at home	5021
median_house_value	b25077_001	Median Value of home	5501
lower_quartile_house_value	b25076_001	Median Value of Home for lowest quartile	5502
higher_quartile_house_value	b25078_001	Median Value of Home in Highest Quartile	5503
median_year_moved_into_hu	b25039_001	Median Year Householder into unit	5511
median_year_moved_into_hu_owner	b25039_002	Median Year Householder into unit Owners	5512
median_year_moved_into_hu_renter	b25039_003	Median Year Householder into unit Renters	5513
hu_total	b25024_001	Universe: Housing units	6001
hu_1_detached	b25024_002	Housing Units 1, Detached	6002
hu_1_attached	b25024_003	Housing Units 1, Attached	6003
hu_2	b25024_004	Housing Units: 2	6004
hu_3_4	b25024_005	Housing Units: 3 or 4	6005
hu_5_9	b25024_006	Housing Units: 5 to 9	6006
hu_10_19	b25024_007	Housing Units: 10 to 19	6007
hu_20_49	b25024_008	Housing Units: 20 to 49	6008
hu_50_plus	b25024_009	Housing Units: 50 or more	6009
hu_mobile	b25024_010	Housing Units: Mobile Home	6010
hu_brv	b25024_011	Housing Units: Boat, RV, van, etc.	6011

var_name	acs_field_name	description	tab_order
rooms_total	b25017_001	Housing Units by Rooms Total	6021
rooms_1	b25017_002	Housing Units 1 Room	6022
rooms_2	b25017_003	Housing Units 2 Rooms	6023
rooms_3	b25017_004	Housing Units 3 Rooms	6024
rooms_4	b25017_005	Housing Units 4 Rooms	6025
rooms_5	b25017_006	Housing Units 5 Rooms	6026
rooms_6	b25017_007	Housing Units 6 Rooms	6027
rooms_7	b25017_008	Housing Units 7 Rooms	6028
rooms_8	b25017_009	Housing Units 8 Rooms	6029
rooms_9plus	b25017_010	Housing Units 9 Plus Rooms	6030
median_number_rooms	b25018_001	Median Number of Rooms in HUs	6031
aggrigate_number_rooms	b25019_001	Aggregate Number of Rooms in HUs	6032
median_rooms_per_hu	b25021_001	Median Rooms Per HU	6040
median_rooms_per_owner_hu	b25021_002	Median Rooms Per Owner Occupied HUs	6041
median_rooms_per_renter_hu	b25021_003	Median Rooms Per Renter Occupied HUs	6042
aggrigate_rooms_per_hu	b25022_001	Aggregate Rooms per HUs	6050
aggrigate_rooms_per_owner_hu	b25022_002	Aggregate Rooms per Owner HUs	6051
aggrigate_rooms_per_renter_hu	b25022_003	Aggregate Rooms per Renter HUs	6052

Program 1: PHP Code to get Data from ACS into LAI Data Structure

```
<?php
$db_acs="acs";
$acs_scheme="acs2012";
$db_hti="hud_lai_2014";
$data_tbl = "acs_bg_2012_data";
include "../../rpgmodel/dbFunctions.inc.php";
ini_set("memory_limit","1280M");
```

```
$sumlev='150';
#
# make a new file if needed by setting the flag $make_new_file to be 'yes'
#
$make_new_file='no';
if($make_new_file == 'yes'){
  $acs_stfid = get_array_from_db($db_acs,"select state||county||tract||blkgrp as
stfid,state,county,tract,state||county as state_county from acs2012.g_2012_5 where sumlevel =
'$sumlev' order by state||county||tract||blkgrp");
  exec_sql($db_hti,"drop table $data_tbl;
                    CREATE TABLE $data_tbl
                    (
                       stfid character varying,
                       state character varying,
                       county character varying,
                       tract character varying,
                       state_county character varying,
                       cbsa character varying
                     )
                     WITH(OIDS = FALSE);");
  foreach($acs_stfid as $bg){
    $stfid=$bg['stfid'];
    $state=$bg['state'];
    $county=$bg['county'];
    $tract=$bg['tract'];
    $state_county=$bg['state_county'];
    exec_sql($db_hti,"Insert into $data_tbl (stfid,state,county,tract,state_county)
                                  values ('$stfid','$state','$county','$tract','$state_county');") ;
  }
  exec_sql($db_hti,"ALTER TABLE $data_tbl ADD PRIMARY KEY (stfid);");
  exec_sql($db_hti,"CREATE INDEX st_co_ndx_$data_tbl  ON $data_tbl (state_county ASC NULLS LAST);");
#
# fill the cbsa right now
#
  echo "Filling cbsa \n";
  $cbsa_state_county=get_array_from_db($db_acs,"select distinct state||county as state_county,cbsa
from $acs_scheme.g_2012_5 where sumlevel='313'");
  foreach($cbsa_state_county as $v){
     $updt="update $data_tbl set cbsa = '".$v[cbsa]."' where state_county = '".$v[state_county]."'";
     exec_sql($db_hti,$updt);
  }
}
#
# Get all the columns we need to keep updated
#
# pick tab_order range so you don't do the whole thing over again...
#
$sql = "select * from variable_definitions
 where tab_order >= 900 and tab_order<1000
 order by tab_order asc ";
$rS=get_array_from_db($db_hti,$sql);
#
$imax=-1;
#
# start building the query to get the data from acs_2012
#
$sql = "select a.state||a.county||a.tract||a.blkgrp as stfid";

$from = "from $acs_scheme.g_2012_5 a";

$where ="where a.sumlevel = '$sumlev' ";
#
##
$sql_mk_dt="";

foreach($rS as $r){
  $tab_col = $r['var_name'];
  $a_c = explode("+",$r['acs_field_name']);
  $tab_des = $r['description'];
  $plus=",";
  foreach($a_c as $acs_col){
    $sql_dd = "select * from information_schema.columns where table_schema='$acs_scheme' and
```

```
column_name ='$acs_col' and substring(table_name,1,9) = 'e_2012_5_'";
    $aS=get_array_from_db($db_acs,$sql_dd);
    $acs_tab = 'g_2012_5';
    if($aS){$acs_tab = $aS[0]['table_name'];}
#
# loop through tables and see if this is already here and assign the correct table index
#
    $j=0;
    $tab_here = -1;
    while($j<=$imax){
      if($acs_tab == $a_t[$j]){$tab_here = $j;}
      $j++;
    }
    if($tab_here>=0){$i=$tab_here;}
    else{
      $i=$j;
      $imax=$j;
      $a_t[$j]=$acs_tab;
      $from = "$from, $acs_scheme.$acs_tab a$i";
      echo("Building tables, $acs_scheme.$acs_tab a$i for acs_col = $acs_col \n");
      $where = "$where and a.stusab = a$i.stusab and a.logrecno = a$i.logrecno";
    }

    $sql = "$sql $plus a$i.$acs_col ";

    $plus="+";
#
# get the variable type - just use the last one...
#
    $sql_col_def = "select data_type from information_schema.columns where table_schema='$acs_scheme'
and column_name ='$acs_col' and table_name = '$acs_tab'";
    $dT=get_array_from_db($db_acs,$sql_col_def);
    $d_type = $dT[0]['data_type'];
  }
  $sql = "$sql as $tab_col";
#
```

```
# Now check to make sure they are in $data_tbl
#  if not add them to the table
#
  $sql_col_check = "select data_type from information_schema.columns where column_name ='$tab_col' and 
table_name = '$data_tbl'";
  $cc=get_array_from_db($db_hti,$sql_col_check);
  if(count($cc[0])==0){
    $sql_mk_dt = "$sql_mk_dt ALTER TABLE $data_tbl ADD COLUMN $tab_col $d_type;\n";
  }
#
  $i++;
}
#
# echo resulting queries
#
echo "sql_mk_dt -> $sql_mk_dt<br>\n";

echo "sql -> $sql<br>\n";
echo " $from<br>\n";
echo " $where<br>\n";

#
# now execute the adding collumns query
#
if($sql_mk_dt){$recordSet = get_array_from_db($db_hti,$sql_mk_dt);}
#
# Get all the states and loop through them
#
$sql_st = "SELECT distinct substring(stfid,1,2) AS state FROM $data_tbl order by state asc;";
$st_id=get_array_from_db($db_hti,$sql_st);
foreach($st_id as $st){
  $state = $st['state'];
  $sql_state = "$sql\n    $from\n    $where and a.state = '$state'";
  $recordSet = get_array_from_db($db_acs,$sql_state);
  if($recordSet){
    foreach($recordSet as $rs){
      $sql_update = "--".$rs['stfid'].";\n update $data_tbl set ";
      $comma="";
      foreach($rs as $t => $v){
      if($t <> 'stfid'){
        if($v==""){$v='NULL';}
        $sql_update = "$sql_update$comma$t=$v";
          $comma=", ";
          }
      }
      $sql_update="$sql_update where $data_tbl.stfid = '".$rs['stfid']."';";
      exec_sql($db_hti,$sql_update);
    }
  }
  echo "Just updated Data State = $state\n";

}

?>
```

Appendix B: Employment Data

This appendix contains programs and SQL that were used to generate the employment variables for the entire country, including the Massachusetts fixes.

Program 2: PHP Script to Build Employment Gravity Measures

```
<?php
$db_hti="hud_lai_2014";
include "../../rmodel/dbFunctions.inc.php";
ini_set("memory_limit","5120M");
#
# This is what we want to calculate gravity for
#
#        'cr01','cr02','cr03','cr04','cr05','cr07',
#        'ct01','ct02',
#        'cd01','cd02','cd03','cd04',
#        'cs01','cs02',
#        'cfa01','cfa02','cfa03','cfa04','cfa05',
#        'ca01','ca02','ca03',
$emp_types =  array(
         'c000',
         'ce01','ce02','ce03',
         'cns01','cns02','cns03','cns04','cns05','cns06','cns07','cns08','cns09','cns10',
         'cns11','cns12','cns13','cns14','cns15','cns16','cns17','cns18','cns19','cns20',
         'cfs01','cfs02','cfs03','cfs04','cfs05',
         'population','hu','occupied_hu','vacant_hu');
$start=time('now');
$l=0;

$st=$argv[1];
$stmax=$argv[2];
#if($st > 56){die;}
if($st == ""){die;}
if($st < 0){die;}
if($stmax == ""){$stmax=$st;}

echo "Okay will run states $st to $stmax \n";
while($st<=$stmax){
  $st_fp="$st";
  if($st<10000){ $st_fp="0$st"; }
  $blkgrps= get_array_from_db($db_hti,"select stfid,occ_hu_lat as use_centroid_lat,occ_hu_lon as
use_centroid_lon
                                               from blkgrp_gravity_2010
                                    where county='$st_fp' and c000 is null
                                    order by stfid");
  if(is_array($blkgrps)){
    $k=count($blkgrps);
    echo "About to start county $st_fp that has $k block groups: ".(-$start+time('now'))." seconds\n";
    $i=0;
    $j=0;
    foreach($blkgrps as $bg){
      $gravity =
build_gravity($db_hti,$emp_types,$bg['stfid'],$bg['use_centroid_lat'],$bg['use_centroid_lon']);
      $us = "-- ".$bg['stfid']." \n
            update blkgrp_gravity_2010 set \n";
      $comma="";
      foreach($emp_types as $v){
        $us = "$us $comma $v=".$gravity[$v];
      $comma=",\n";
      }
      $us = "$us where stfid = '".$bg['stfid']."'";
      exec_sql($db_hti,$us);
      if($j==100){
        echo " Have just completed $i of $k blockgroups in county $st_fp a total of $l this took ".(-
$start+time('now'))." seconds\n";
        $j=0;
      }
      $i++;
      $l++;
      $j++;
    }
  }
  $st++;
}
#
#/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/
\
#
# Functions for doing this
#
function build_gravity($db_hti,$emp_types,$this_stfid,$centroid_lat,$centroid_lon){
#
# Initialize gravity array
# and build sql
#
$sql = "-- $this_stfid \n select stfid,emp_lon as centroid_lon,emp_lat as centroid_lat";
foreach($emp_types as $et){
  $grv_full[$et]=0.0;
  $sql = "$sql, $et";
}
#
# on average
#   blocks are 0.563194016 miles apart
#   block groups are 4.002733441 miles apart
#   tracts are 6.907310439 miles apart
#   counties are 33.06612693 miles apart
#   states are 251.280847 miles apart
#
# so make it so we get 2.5 block around us on average...
#
$fudge=2.5*0.56/4.00;
$blkgrp_d=$fudge*4.00;
$tract_d=$fudge*6.91;
$county_d=$fudge*33.07;
$state_d=$fudge*251.28;
```

```
#
# so loop over all combinations of things
#
# first get the distance to the job center of each state
#
$state_dist = get_array_from_db($db_hti,"--state \n $sql from emp_pop_hu_2010_state;");
$gv_all=0;
foreach($state_dist as $st){
  $state=$st['stfid'];
  $dist =  getDistance($centroid_lat, $centroid_lon, $st['centroid_lat'], $st['centroid_lon']);
#
# check to see if we need to run this at a county level
#
  if($dist < $state_d or $state == substr($this_stfid,0,2)){
    $cnty_sql = "--county state = $state \n $sql from emp_pop_hu_2010_county where state='$state';";
    $cnty_dist = get_array_from_db($db_hti,$cnty_sql);
    foreach($cnty_dist as $ct){
      $c=$ct['stfid'];
      $dist =  getDistance($centroid_lat, $centroid_lon, $ct['centroid_lat'], $ct['centroid_lon']);
#
# check to see if we need to run this at a tract level
#
      if($dist < $county_d or $c ==  substr($this_stfid,0,5)){
        $tract_sql = "$sql from emp_pop_hu_2010_tract where county='$c' ;";
        $tract_dist = get_array_from_db($db_hti,$tract_sql);
        foreach($tract_dist as $tr){
          $t = $tr['stfid'];
          $dist =  getDistance($centroid_lat, $centroid_lon, $tr['centroid_lat'],
$tr['centroid_lon']);
#
# check to see if we need to run this at the blkgrp level
#
          if($dist < $tract_d or $t ==  substr($this_stfid,0,11)){
            $blkgrp_sql = "$sql from emp_pop_hu_2010_blkgrp where tract='$t' ;";
            $blkgrp_dist = get_array_from_db($db_hti,$blkgrp_sql);
            foreach($blkgrp_dist as $bg){
              $g = $bg['stfid'];
              $dist =  getDistance($centroid_lat, $centroid_lon, $bg['centroid_lat'],
$bg['centroid_lon']);
#
# check to see if we need to run this at the block level
#
              if($dist < $blkgrp_d or $g ==  substr($this_stfid,0,12)){
                $block_sql = "$sql from emp_pop_hu_2010_block where blkgrp='$g' ;";

                $block_dist = get_array_from_db($db_hti,$block_sql);
                foreach($block_dist as $bl){
                  $dist =  getDistance($centroid_lat, $centroid_lon, $bl['centroid_lat'],
$bl['centroid_lon']);
                  $grv_full = increment_grv($grv_full,$dist,$bl);
                }
              }
              else{$grv_full = increment_grv($grv_full,$dist,$bg);}
            }
          }
          else{$grv_full = increment_grv($grv_full,$dist,$tr);}
        }
      }
      else{$grv_full = increment_grv($grv_full,$dist,$ct);}
    }
  }
  else{$grv_full = increment_grv($grv_full,$dist,$st);}
}
return $grv_full;
}
function increment_grv($grv_full,$dist,$dat){
  $d2=$dist*$dist;
  if($d2 < 1.0){$d2=1.0;}
  foreach($grv_full as $p => $v){
    $grv_full[$p]=$grv_full[$p]+$dat[$p]/$d2;
  }
  return $grv_full;
```

```
}
#
# This calculates the distance between two points
#
function getDistance($latitude1, $longitude1, $latitude2, $longitude2) {
#    $earth_radius = 6371;  # kilometers
#    $earth_radius = 3958.761;  # miles
    $dLat = deg2rad($latitude2 - $latitude1);
    $dLon = deg2rad($longitude2 - $longitude1);

    $a = sin($dLat/2) * sin($dLat/2) + cos(deg2rad($latitude1)) * cos(deg2rad($latitude2)) * 
sin($dLon/2) * sin($dLon/2);
    $c = 2 * asin(sqrt($a));
    return 3958.761 * $c;
}

?>
```

Program 3: PHP Code to Create Local Employment

```
<?php
$db_hti="hud_lai_2014";
include "../../rmodel/dbFunctions.inc.php";
ini_set("memory_limit","5120M");
#
# This is what we want to calculate local job density for
#
$emp_types =  array(
          'c000',
          'cns01','cns02','cns03','cns04','cns05','cns06','cns07','cns08','cns09','cns10',
          'cns11','cns12','cns13','cns14','cns15','cns16','cns17','cns18','cns19','cns20',
          'population','occupied_hu','aland');
$cols="";
```

```
$comma="";
foreach($emp_types as $et){
  $cols="$cols$comma $et = e.$et";
  $col_list="$col_list$comma e.$et";
  $sum_list="$sum_list$comma sum(e.$et) as $et";
  $st_sum_list="$st_sum_list$comma sum($et*inarea/flarea) as $et";
  $comma=",\n";
}
echo "want to update these: $cols \n";

$start=time('now');
$l=0;

$st=$argv[1];
$stmax=$argv[2];
if($st > 56){die;}
if($st == ""){die;}
if($st < 0){die;}
if($stmax == ""){$stmax=$st;}

echo "Okay will run states $st to $stmax \n";
while($st<=$stmax){
  $st_fp="$st";
  if($st<10){ $st_fp="0$st"; }
  echo "About to update unchaged ones in state = $st_fp \n";
  $up_same="update local_employment_2010 l set $cols from emp_pop_hu_2010_blkgrp e where
e.stfid=l.stfid and e.state='$st_fp'";

  exec_sql($db_hti,$up_same);
  $need_same=get_array_from_db($db_hti,"select count(*) as cnt from local_employment_2010 where same
is NULL and substring(stfid,1,2)='$st_fp'");
  if($need_same[0]['cnt'] > 0){
   exec_sql($db_hti,"update local_employment_2010 set same = st_equals(geom,combined_geom) where
substring(stfid,1,2)='$st_fp'");
  }
  $blkgrps= get_array_from_db($db_hti,"select
stfid,$col_list,occ_hu_lon,occ_hu_lat,ST_Astext(combined_geom) as geom
                                               from local_employment_2010 e
                                        where substring(stfid,1,2)='$st_fp' and not(same)
                                        order by stfid");
  if(is_array($blkgrps)){
    $k=count($blkgrps);
    echo "About to start state $st_fp that has $k block groups: ".(-$start+time('now'))." seconds\n";
    $i=0;
    $j=0;
    foreach($blkgrps as $bg){
      $stfid=$bg[stfid];
      $lon=$bg[occ_hu_lon];
      $lat=$bg[occ_hu_lat];
      $geom="st_geomfromtext('".$bg[geom]."',4326)";
#
      $bd_sql = "select $sum_list
                from emp_pop_hu_2010_block e
                join census_blocks_2012 c on c.geoid=e.stfid
             where not(e.blkgrp='$stfid') and st_intersects($geom,c.geom)";
#       echo "try this $bd_sql \n";
      $all_in=get_array_from_db($db_hti,$bd_sql);
#
      $up_sql="update local_employment_2010 set ";
      $comma="";
      foreach($emp_types as $et){
        $jobs=$bg[$et] + $all_in[0][$et];
        $up_sql="$up_sql$comma $et=$jobs";
        $comma=",";
      }
#       echo "update > $up_sql \n";
      exec_sql($db_hti,"$up_sql where stfid = '$stfid'");
      $i++;
      $j++;
      if($j == 100){
        echo "In state $st_fp you have done $i out of $k in ".(-$start+time('now'))." seconds\n";
        $j=0;
      }
    }
  }
  $st++;
}

?>
```

Program 4: SQL to Generate Massachusetts County Employment Ratios

```
-- View: ctpp_2000_es202_2010_compare

-- DROP VIEW ctpp_2000_es202_2010_compare;

CREATE OR REPLACE VIEW ctpp_2000_es202_2010_compare AS
 SELECT bg2000.county, es202_2010.emp_type_01_2010_no_military::numeric / (bg2000.emp_type_01 - bg2000.emp_type_15)::numeric AS rat_01_no_mil,
    CASE
      WHEN bg2000.emp_type_02 > 0 THEN es202_2010.emp_type_02_2010::numeric / bg2000.emp_type_02::numeric
      ELSE 0::numeric
    END AS rat_02,
    CASE
      WHEN bg2000.emp_type_03 > 0 THEN es202_2010.emp_type_03_2010::numeric / bg2000.emp_type_03::numeric
      ELSE 0::numeric
    END AS rat_03,
    CASE
      WHEN bg2000.emp_type_04 > 0 THEN es202_2010.emp_type_04_2010::numeric / bg2000.emp_type_04::numeric
      ELSE 0::numeric
    END AS rat_04,
    CASE
      WHEN bg2000.emp_type_05 > 0 THEN es202_2010.emp_type_05_2010::numeric / bg2000.emp_type_05::numeric
      ELSE 0::numeric
    END AS rat_05,
    CASE
      WHEN bg2000.emp_type_06 > 0 THEN es202_2010.emp_type_06_2010::numeric / bg2000.emp_type_06::numeric
      ELSE 0::numeric
    END AS rat_06,
    CASE
      WHEN bg2000.emp_type_07 > 0 THEN es202_2010.emp_type_07_2010::numeric / bg2000.emp_type_07::numeric
      ELSE 0::numeric
    END AS rat_07,
    CASE
      WHEN bg2000.emp_type_08 > 0 THEN es202_2010.emp_type_08_2010::numeric / bg2000.emp_type_08::numeric
      ELSE 0::numeric
    END AS rat_08,
    CASE
      WHEN bg2000.emp_type_09 > 0 THEN es202_2010.emp_type_09_2010::numeric / bg2000.emp_type_09::numeric
      ELSE 0::numeric
    END AS rat_09,
    CASE
      WHEN bg2000.emp_type_10 > 0 THEN es202_2010.emp_type_10_2010::numeric / bg2000.emp_type_10::numeric
      ELSE 0::numeric
    END AS rat_10,
    CASE
      WHEN bg2000.emp_type_11 > 0 THEN es202_2010.emp_type_11_2010::numeric / bg2000.emp_type_11::numeric
      ELSE 0::numeric
    END AS rat_11,
    CASE
      WHEN bg2000.emp_type_12 > 0 THEN es202_2010.emp_type_12_2010::numeric / bg2000.emp_type_12::numeric
      ELSE 0::numeric
```

```
        END AS rat_12,
        CASE
           WHEN bg2000.emp_type_13 > 0 THEN es202_2010.emp_type_13_2010::numeric / bg2000.emp_type_13::numeric
           ELSE 0::numeric
        END AS rat_13,
        CASE
           WHEN bg2000.emp_type_14 > 0 THEN es202_2010.emp_type_14_2010::numeric / bg2000.emp_type_14::numeric
           ELSE 0::numeric
        END AS rat_14,
        CASE
           WHEN bg2000.emp_type_15 > 0 THEN es202_2010.emp_type_15_2010::numeric / bg2000.emp_type_15::numeric
           ELSE 0::numeric
        END AS rat_15, bg2000.emp_type_01, bg2000.emp_type_02, bg2000.emp_type_03, bg2000.emp_type_04, bg2000.emp_type_05,
bg2000.emp_type_06, bg2000.emp_type_07, bg2000.emp_type_08, bg2000.emp_type_09, bg2000.emp_type_10, bg2000.emp_type_11,
bg2000.emp_type_12, bg2000.emp_type_13, bg2000.emp_type_14, bg2000.emp_type_15, bg2000.emp_type_01 - bg2000.emp_type_15 AS
emp_type_01_no_military, es202_2010.emp_type_01_2010_no_military, es202_2010.emp_type_02_2010, es202_2010.emp_type_03_2010,
es202_2010.emp_type_04_2010, es202_2010.emp_type_05_2010, es202_2010.emp_type_06_2010, es202_2010.emp_type_07_2010,
es202_2010.emp_type_08_2010, es202_2010.emp_type_09_2010, es202_2010.emp_type_10_2010, es202_2010.emp_type_11_2010,
es202_2010.emp_type_13_2010, es202_2010.emp_type_14_2010, es202_2010.emp_type_15_2010
   FROM (        ( SELECT mass_emp_by_industry_ctpp_2000_bg.county, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x1) AS emp_type_01,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x2) AS emp_type_02, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x3) AS emp_type_03,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x4) AS emp_type_04, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x5) AS emp_type_05,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x6) AS emp_type_06, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x7) AS emp_type_07,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x8) AS emp_type_08, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x9) AS emp_type_09,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x10) AS emp_type_10, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x11) AS emp_type_11,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x12) AS emp_type_12, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x13) AS emp_type_13,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x14) AS emp_type_14, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x15) AS emp_type_15
                 FROM mass_emp_by_industry_ctpp_2000_bg
                GROUP BY mass_emp_by_industry_ctpp_2000_bg.county
                ORDER BY mass_emp_by_industry_ctpp_2000_bg.county)
        UNION
        SELECT 'state'::character varying AS county, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x1) AS emp_type_01,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x2) AS emp_type_02, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x3) AS emp_type_03,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x4) AS emp_type_04, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x5) AS emp_type_05,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x6) AS emp_type_06, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x7) AS emp_type_07,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x8) AS emp_type_08, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x9) AS emp_type_09,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x10) AS emp_type_10, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x11) AS emp_type_11,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x12) AS emp_type_12, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x13) AS emp_type_13,
sum(mass_emp_by_industry_ctpp_2000_bg.tab4x14) AS emp_type_14, sum(mass_emp_by_industry_ctpp_2000_bg.tab4x15) AS emp_type_15
         FROM mass_emp_by_industry_ctpp_2000_bg) bg2000
 JOIN ( SELECT x01.county, x01.emp AS emp_type_01_2010_no_military,
       CASE
         WHEN x02.emp IS NULL THEN 0::bigint
         ELSE x02.emp
       END AS emp_type_02_2010,
       CASE
         WHEN x03.emp IS NULL THEN 0::bigint
         ELSE x03.emp
       END AS emp_type_03_2010,
       CASE
         WHEN x04.emp IS NULL THEN 0::bigint
         ELSE x04.emp
       END AS emp_type_04_2010,
       CASE
         WHEN x05.emp IS NULL THEN 0::bigint
         ELSE x05.emp
       END AS emp_type_05_2010,
       CASE
         WHEN x06.emp IS NULL THEN 0::bigint
         ELSE x06.emp
       END AS emp_type_06_2010,
       CASE
```

```
      WHEN x07.emp IS NULL THEN 0::bigint
      ELSE x07.emp
    END AS emp_type_07_2010,
    CASE
      WHEN x08.emp IS NULL THEN 0::bigint
      ELSE x08.emp
    END AS emp_type_08_2010,
    CASE
      WHEN x09.emp IS NULL THEN 0::bigint
      ELSE x09.emp
    END AS emp_type_09_2010,
    CASE
      WHEN x10.emp IS NULL THEN 0::bigint
      ELSE x10.emp
    END AS emp_type_10_2010,
    CASE
      WHEN x11.emp IS NULL THEN 0::bigint
      ELSE x11.emp
    END AS emp_type_11_2010,
    CASE
      WHEN x12.emp IS NULL THEN 0::bigint
      ELSE x12.emp
    END AS emp_type_12_2010,
    CASE
      WHEN x13.emp IS NULL THEN 0::bigint
      ELSE x13.emp
    END AS emp_type_13_2010,
    CASE
      WHEN x14.emp IS NULL THEN 0::bigint
      ELSE x14.emp
    END AS emp_type_14_2010,
            CASE
               WHEN x15.emp IS NULL THEN 0::bigint
               ELSE x15.emp
            END AS emp_type_15_2010
        FROM ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
              FROM mass_es202
             WHERE NOT (mass_es202.description::text = ANY (ARRAY['DUR - Durable Goods Manufacturing'::character varying::text, 'NONDUR -
Non-Durable Goods Manufacturing'::character varying::text]))
             GROUP BY mass_es202.county
             ORDER BY mass_es202.county) x01
    LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
              FROM mass_es202
             WHERE mass_es202.description::text = ANY (ARRAY['11 - Agriculture, Forestry, Fishing & Hunting'::character varying::text, '21 -
Mining'::character varying::text])
             GROUP BY mass_es202.county
             ORDER BY mass_es202.county) x02 ON x01.county::text = x02.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
          FROM mass_es202
         WHERE mass_es202.description::text = '23 - Construction'::text
         GROUP BY mass_es202.county
         ORDER BY mass_es202.county) x03 ON x01.county::text = x03.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
      FROM mass_es202
     WHERE mass_es202.description::text = '31-33 - Manufacturing'::text
     GROUP BY mass_es202.county
     ORDER BY mass_es202.county) x04 ON x01.county::text = x04.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
   FROM mass_es202
  WHERE mass_es202.description::text = '42 - Wholesale Trade'::text
  GROUP BY mass_es202.county
  ORDER BY mass_es202.county) x05 ON x01.county::text = x05.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = '44-45 - Retail Trade'::text
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x06 ON x01.county::text = x06.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = ANY (ARRAY['22 - Utilities'::character varying::text, '48-49 - Transportation and Warehousing'::character
varying::text])
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x07 ON x01.county::text = x07.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = '51 - Information'::text
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x08 ON x01.county::text = x08.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = ANY (ARRAY['52 - Finance and Insurance'::character varying::text, '53 - Real Estate and Rental and
Leasing'::character varying::text])
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x09 ON x01.county::text = x09.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = ANY (ARRAY['54 - Professional and Technical Services'::character varying::text, '55 - Management of
Companies and Enterprises'::character varying::text, '56 - Administrative and Waste Services'::character varying::text])
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x10 ON x01.county::text = x10.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = ANY (ARRAY['61 - Educational Services'::character varying::text, '62 - Health Care and Social
Assistance'::character varying::text])
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x11 ON x01.county::text = x11.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = ANY (ARRAY['71 - Arts, Entertainment, and Recreation'::character varying::text, '72 - Accommodation
and Food Services'::character varying::text])
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x12 ON x01.county::text = x12.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = '81 - Other Services, Ex. Public Admin'::text
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x13 ON x01.county::text = x13.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = '92 - Public Administration'::text
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x14 ON x01.county::text = x14.county::text
  LEFT JOIN ( SELECT mass_es202.county, sum(replace(mass_es202.avg_month_employment::text, ','::text, ''::text)::integer) AS emp
  FROM mass_es202
 WHERE mass_es202.description::text = 'army'::text
 GROUP BY mass_es202.county
 ORDER BY mass_es202.county) x15 ON x01.county::text = x15.county::text) es202_2010 ON bg2000.county::text = es202_2010.county::text
 ORDER BY bg2000.county;

ALTER TABLE ctpp_2000_es202_2010_compare
 OWNER TO nobody;
```

Program 5: SQL to Create Massachusetts Employment Data at Block Group Level

```
SELECT bg.stfid, bg.county,
      tab4x1-tab4x15 as tab_4x1_no_mil, tab4x1, tab4x2, tab4x3, tab4x4, tab4x5,
      tab4x6, tab4x7, tab4x8, tab4x9, tab4x10, tab4x11, tab4x12, tab4x13,
      tab4x14, tab4x15,
      (tab4x1-tab4x15)*rat.rat_01_no_mil as tab_4x1_no_mil_projected_2010,
      tab4x2*rat.rat_02 as  tab4x2_projected_2010,
      tab4x3*rat.rat_03 as  tab4x3_projected_2010,
      tab4x4*rat.rat_04 as  tab4x4_projected_2010,
      tab4x5*rat.rat_05 as  tab4x5_projected_2010,
      tab4x6*rat.rat_06 as  tab4x6_projected_2010,
      tab4x7*rat.rat_07 as  tab4x7_projected_2010,
      tab4x8*rat.rat_08 as  tab4x8_projected_2010,
      tab4x9*rat.rat_09 as  tab4x9_projected_2010,
      tab4x10*rat.rat_10 as  tab4x10_projected_2010,
      tab4x11*rat.rat_11 as  tab4x11_projected_2010,
      tab4x12*rat.rat_12 as  tab4x12_projected_2010,
      tab4x13*rat.rat_13 as  tab4x13_projected_2010,
      tab4x14*rat.rat_14 as  tab4x14_projected_2010,
      tab4x15*rat.rat_15 as  tab4x15_projected_2010,
      ST_X(ST_centroid(g.the_geom)) as x_centroid,
      ST_Y(ST_centroid(g.the_geom)) as y_centroid
  into mass_employment_2010
  FROM mass_emp_by_industry_ctpp_2000_bg bg
  join blkgrps_2000 g on bg.stfid=g.stfid
  join ctpp_2000_es202_2010_compare as rat on bg.county = rat.county;
ALTER TABLE mass_employment_2010
  ADD PRIMARY KEY (stfid);
```

Appendix C: Creating Data for Fitting

Program 6: SQL to make View for Fitting

```
-- View: place_blkgrp_data

-- DROP VIEW place_blkgrp_data;

CREATE OR REPLACE VIEW place_blkgrp_data AS
 SELECT t.stfid, "substring"(t.stfid::text, 1, 2) AS state, "substring"(t.stfid::text, 1, 5) AS
county, c.cbsa, cb.tci_ready, t.households, t.occupied_hu, t.owner_occupied_hu,
t.renter_occupied_hu, bgi.occupied_hu_trk, bgi.owner_occupied_hu_trk, bgi.renter_occupied_hu_trk,
        CASE
            WHEN t.occupied_hu > 0::numeric::double precision THEN t.renter_occupied_hu /
t.occupied_hu
            ELSE NULL::numeric::double precision
        END AS frac_renters, t.median_hh_income::numeric AS median_hh_income,
bgi.prop_owner_median_hh_income * bgi.beta_hh_income::numeric::double precision AS
median_hh_income_owners, bgi.prop_renter_median_hh_income * bgi.beta_hh_income::numeric::double
precision AS median_hh_income_renters, t.median_hh_income / cnty.median_hh_income AS
med_income_frac_county, t.median_hh_income / c.median_hh_income AS med_income_frac_cbsa,
cnty.median_hh_income AS county_median_income, c.median_hh_income AS cbsa_median_income,
bgi.prop_owner_median_hh_income * bgi.beta_hh_income / c.median_hh_income AS
cbsa_income_owners_frac, bgi.prop_renter_median_hh_income * bgi.beta_hh_income /
c.median_hh_income AS cbsa_income_renters_frac, bgi.prop_owner_median_hh_income *
bgi.beta_hh_income / cnty.median_hh_income AS county_income_owners_frac,
bgi.prop_renter_median_hh_income * bgi.beta_hh_income / cnty.median_hh_income AS
county_income_renters_frac,
        CASE
            WHEN c.cbsa IS NULL THEN t.median_hh_income / cnty.median_hh_income
            ELSE t.median_hh_income / c.median_hh_income
        END AS area_income_frac,
        CASE
            WHEN c.cbsa IS NULL THEN bgi.prop_owner_median_hh_income * bgi.beta_hh_income /
```

```
cnty.median_hh_income
            ELSE bgi.prop_owner_median_hh_income * bgi.beta_hh_income / c.median_hh_income
        END AS area_income_owner_frac,
        CASE
            WHEN c.cbsa IS NULL THEN bgi.prop_renter_median_hh_income * bgi.beta_hh_income /
cnty.median_hh_income
            ELSE bgi.prop_renter_median_hh_income * bgi.beta_hh_income / c.median_hh_income
        END AS area_income_renter_frac,
        CASE
            WHEN c.cbsa IS NULL THEN cnty.median_hh_income
            ELSE c.median_hh_income
        END AS area_median_hh_income,
        CASE
            WHEN c.cbsa IS NULL THEN cnty.median_hh_income
            ELSE c.median_hh_income
        END AS ami_median_hh_income,
        CASE
            WHEN c.cbsa IS NULL THEN 1.0::double precision
            ELSE c.median_hh_income / cnty.median_hh_income
        END AS ami_med_income_frac_county,
        CASE
            WHEN c.cbsa IS NULL THEN cnty.median_hh_income_owners
            ELSE c.median_hh_income_owners
        END AS ami_median_hh_income_owners,
        CASE
            WHEN c.cbsa IS NULL THEN
            CASE
                WHEN cnty.median_hh_income > 0::numeric::double precision THEN
cnty.median_hh_income_owners / cnty.median_hh_income
                ELSE NULL::numeric::double precision
            END
            ELSE
            CASE
                WHEN c.median_hh_income > 0::numeric::double precision THEN
```

```
c.median_hh_income_owners / c.median_hh_income
                ELSE NULL::numeric::double precision
            END
        END AS ami_median_hh_income_owners_frac,
        CASE
            WHEN c.cbsa IS NULL THEN cnty.median_hh_income_renters
            ELSE c.median_hh_income_renters
        END AS ami_median_hh_income_renters,
        CASE
            WHEN c.cbsa IS NULL THEN
            CASE
                WHEN cnty.median_hh_income > 0::numeric::double precision THEN
cnty.median_hh_income_renters / cnty.median_hh_income
                ELSE NULL::numeric::double precision
            END
            ELSE
            CASE
                WHEN c.median_hh_income > 0::numeric::double precision THEN
c.median_hh_income_renters / c.median_hh_income
                ELSE NULL::numeric::double precision
            END
        END AS ami_median_hh_income_renters_frac,
        CASE
            WHEN c.cbsa IS NULL THEN cnty.avg_hh_size
            ELSE c.avg_hh_size
        END AS ami_avg_hh_size,
        CASE
            WHEN c.cbsa IS NULL THEN cnty.avg_hh_size_owners
            ELSE c.avg_hh_size_owners
        END AS ami_avg_hh_size_owners,
        CASE
            WHEN c.cbsa IS NULL THEN cnty.avg_hh_size_renters
            ELSE c.avg_hh_size_renters
        END AS ami_avg_hh_size_renters,
        CASE
            WHEN t.avg_hh_size IS NOT NULL THEN t.avg_hh_size
            ELSE
            CASE
                WHEN t.households > 0::numeric::double precision THEN t.hh_population::numeric / 
t.households::numeric
                ELSE NULL::numeric
            END::double precision
        END AS avg_hh_size,
        CASE
            WHEN t.avg_hh_size_owners IS NOT NULL THEN t.avg_hh_size_owners::numeric
            ELSE
            CASE
                WHEN t.owner_occupied_hu > 0::numeric::double precision THEN 
t.population_owner_occ_hu::numeric / t.owner_occupied_hu::numeric
                ELSE NULL::numeric
            END
        END AS avg_hh_size_owners,
        CASE
            WHEN t.avg_hh_size_renters IS NOT NULL THEN t.avg_hh_size_renters::numeric
            ELSE
            CASE
                WHEN t.renter_occupied_hu > 0::numeric::double precision THEN 
t.population_renter_occ_hu::numeric / t.renter_occupied_hu::numeric
                ELSE NULL::numeric
            END
        END AS avg_hh_size_renters, bgi.commuters_per_hh, bgi.commuters_per_hh_trk, 
bgi.commuters_per_hh_owners_trk, bgi.commuters_per_hh_renters_trk, 
bgi.prop_commuters_per_hh_owners * bgi.beta_commuters_per_hh AS commuters_per_hh_owners, 
bgi.prop_commuters_per_hh_renters * bgi.beta_commuters_per_hh AS commuters_per_hh_renters, 
        CASE
            WHEN c.cbsa IS NULL THEN
            CASE
                WHEN (cnty.occupied_hu * cnty.hh_population) > 0::numeric::double precision THEN
```

```
(cnty.workers_by_tenure_total - cnty.workers_by_tenure_home) * cnty.population / (cnty.occupied_hu
* cnty.hh_population)
                ELSE NULL::numeric::double precision
            END
            ELSE
            CASE
                WHEN (c.occupied_hu * c.hh_population) > 0::numeric::double precision THEN
(c.workers_by_tenure_total - c.workers_by_tenure_home) * c.population / (c.occupied_hu *
c.hh_population)
                ELSE NULL::numeric::double precision
            END
        END AS ami_commuters_per_hh,
        CASE
            WHEN c.cbsa IS NULL THEN
            CASE
                WHEN (cnty.owner_occupied_hu * cnty.hh_population) > 0::numeric::double precision
THEN (cnty.workers_by_tenure_total_owner - cnty.workers_by_tenure_home_owner) * cnty.population /
(cnty.owner_occupied_hu * cnty.hh_population)
                ELSE NULL::numeric::double precision
            END
            ELSE
            CASE
                WHEN (c.owner_occupied_hu * c.hh_population) > 0::numeric::double precision THEN
(c.workers_by_tenure_total_owner - c.workers_by_tenure_home_owner) * c.population /
(c.owner_occupied_hu * c.hh_population)
                ELSE NULL::numeric::double precision
            END
        END AS ami_commuters_per_hh_owners,
        CASE
            WHEN c.cbsa IS NULL THEN
            CASE
                WHEN (cnty.renter_occupied_hu * cnty.hh_population) > 0::numeric::double precision
THEN (cnty.workers_by_tenure_total_renters - cnty.workers_by_tenure_home_renters) *
cnty.population / (cnty.renter_occupied_hu * cnty.hh_population)
```

```
                ELSE NULL::numeric::double precision
            END
            ELSE
            CASE
                WHEN (c.renter_occupied_hu * c.hh_population) > 0::numeric::double precision THEN
(c.workers_by_tenure_total_renters - c.workers_by_tenure_home_renters) * c.population /
(c.renter_occupied_hu * c.hh_population)
                ELSE NULL::numeric::double precision
            END
        END AS ami_commuters_per_hh_renters, (d.number_intersections::numeric * 4046.86)::double
precision / a.arealand AS intersection_density, ((d.number_blocks_total::numeric *
4046.86)::double precision / a.arealand)::numeric AS block_density, d.total_block_perimeter_meters
/ d.number_blocks_perimeter::numeric AS avg_block_perimeter, d.total_block_perimeter_meters /
d.number_blocks_perimeter::numeric * ((d.number_blocks_total::numeric * 4046.86)::double precision
/ a.arealand)::numeric AS avg_block_perimeter_per_acre,
        CASE
            WHEN a.arealand > 0::numeric::double precision THEN (t.households::numeric *
4046.86)::double precision / a.arealand
            ELSE NULL::numeric::double precision
        END AS gross_hh_density,
        CASE
            WHEN d.land_acres_gt05 > 0::numeric THEN (t.households *
d.households_gt05::numeric::double precision)::numeric / (d.land_acres_gt05 *
d.households_gt0::numeric)
            ELSE 0::numeric
        END AS res_density_gt05, l.c000::numeric + m.tab_4x1_no_mil_projected_2010 AS emp_gravity,
l.cns07::numeric + m.tab4x6_projected_2010 AS retail_gravity,
        CASE
            WHEN cb.tci_ready THEN s.tci
            ELSE
            CASE
                WHEN c.cbsa IS NULL THEN 0::numeric
                ELSE NULL::numeric
            END
        END AS tci,
        CASE
            WHEN "substring"(t.stfid::text, 1, 2) = '25'::text THEN md.avg_d
            ELSE cd.median_distance_s000::numeric
        END AS avg_d, c.median_smoc_mortgage AS cbsa_median_smoc, c.median_gross_rent AS
cbsa_median_gr, cnty.median_smoc_mortgage AS county_median_smoc, cnty.median_gross_rent AS
county_median_gr,
        CASE
            WHEN t.households > 0::numeric::double precision THEN t.agg_hh_vehicles::numeric /
t.households::numeric
            ELSE NULL::numeric
        END AS autos_per_hh,
        CASE
            WHEN t.renter_occupied_hu > 0::numeric::double precision THEN
t.agg_hh_vehicles_renters::numeric / t.renter_occupied_hu::numeric
            ELSE NULL::numeric
        END AS autos_per_hh_renters,
        CASE
            WHEN t.owner_occupied_hu > 0::numeric::double precision THEN
t.agg_hh_vehicles_owners::numeric / t.owner_occupied_hu::numeric
            ELSE NULL::numeric
        END AS autos_per_hh_owners,
        CASE
            WHEN (t.total_workers - t.workers_home) > 0::numeric::double precision THEN 100.0 *
t.workers_transit::numeric / (t.total_workers - t.workers_home)::numeric
            ELSE NULL::numeric
        END AS pct_transit_commuters,
        CASE
            WHEN ((trk.owner_workers_total - trk.owner_workers_home) * trk.workers_transit *
(t.total_workers - t.workers_home)) > 0::numeric::double precision THEN 100.0 *
(trk.owner_workers_transit::numeric * (trk.total_workers - trk.workers_home)::numeric *
t.workers_transit::numeric) / ((trk.owner_workers_total - trk.owner_workers_home) *
trk.workers_transit * (t.total_workers - t.workers_home))::numeric
            ELSE
            CASE
                WHEN trk.workers_transit = 0::numeric::double precision THEN 0::numeric
                ELSE NULL::numeric
            END
        END AS pct_transit_commuters_owners,
        CASE
            WHEN ((trk.renter_workers_total - trk.renter_workers_home) * trk.workers_transit *
(t.total_workers - t.workers_home)) > 0::numeric::double precision THEN 100.0 *
(trk.renter_workers_transit::numeric * (trk.total_workers - trk.workers_home)::numeric *
t.workers_transit::numeric) / ((trk.renter_workers_total - trk.renter_workers_home) *
trk.workers_transit * (t.total_workers - t.workers_home))::numeric
            ELSE
            CASE
                WHEN trk.workers_transit = 0::numeric::double precision THEN 0::numeric
                ELSE NULL::numeric
            END
        END AS pct_transit_commuters_renters, (t.total_workers - t.workers_home)::numeric AS
commuters, v.avg_vmt_per_year,
        CASE
            WHEN t.households > 0::numeric::double precision THEN v.avg_vmt_per_year *
t.agg_hh_vehicles::numeric / t.households::numeric
            ELSE NULL::numeric
        END AS vmt_per_hh, v.num_autos, v.avg_model_year, v.stdev, v.eom, t.median_smoc,
t.median_smoc_mortgage, t.median_gross_rent,
        CASE
            WHEN t.gr_000_100 > 0::numeric::double precision THEN 0
            WHEN t.gr_100_149 > 0::numeric::double precision THEN 100
            WHEN t.gr_150_199 > 0::numeric::double precision THEN 150
            WHEN t.gr_200_249 > 0::numeric::double precision THEN 200
            WHEN t.gr_250_299 > 0::numeric::double precision THEN 150
            WHEN t.gr_300_349 > 0::numeric::double precision THEN 300
            WHEN t.gr_350_399 > 0::numeric::double precision THEN 350
            WHEN t.gr_400_449 > 0::numeric::double precision THEN 400
            WHEN t.gr_450_499 > 0::numeric::double precision THEN 450
```

```
      WHEN t.gr_500_549 > 0::numeric::double precision THEN 500
      WHEN t.gr_550_599 > 0::numeric::double precision THEN 550
      WHEN t.gr_600_649 > 0::numeric::double precision THEN 600
      WHEN t.gr_650_699 > 0::numeric::double precision THEN 650
      WHEN t.gr_700_749 > 0::numeric::double precision THEN 700
      WHEN t.gr_750_799 > 0::numeric::double precision THEN 750
      WHEN t.gr_800_899 > 0::numeric::double precision THEN 800
      WHEN t.gr_900_999 > 0::numeric::double precision THEN 900
      WHEN t.gr_1000_1249 > 0::numeric::double precision THEN 1000
      WHEN t.gr_1250_1499 > 0::numeric::double precision THEN 1250
      WHEN t.gr_1500_1999 > 0::numeric::double precision THEN 1500
      WHEN t.gr_2000_plus > 0::numeric::double precision THEN 2000
      ELSE 0
    END AS gr_min,
    CASE
      WHEN t.gr_2000_plus > 0::numeric::double precision THEN 2500
      WHEN t.gr_1500_1999 > 0::numeric::double precision THEN 2000
      WHEN t.gr_1250_1499 > 0::numeric::double precision THEN 1500
      WHEN t.gr_1000_1249 > 0::numeric::double precision THEN 1250
      WHEN t.gr_900_999 > 0::numeric::double precision THEN 1000
      WHEN t.gr_800_899 > 0::numeric::double precision THEN 900
      WHEN t.gr_750_799 > 0::numeric::double precision THEN 800
      WHEN t.gr_700_749 > 0::numeric::double precision THEN 750
      WHEN t.gr_650_699 > 0::numeric::double precision THEN 700
      WHEN t.gr_600_649 > 0::numeric::double precision THEN 650
      WHEN t.gr_550_599 > 0::numeric::double precision THEN 600
      WHEN t.gr_500_549 > 0::numeric::double precision THEN 550
      WHEN t.gr_450_499 > 0::numeric::double precision THEN 500
      WHEN t.gr_400_449 > 0::numeric::double precision THEN 450
      WHEN t.gr_350_399 > 0::numeric::double precision THEN 400
      WHEN t.gr_300_349 > 0::numeric::double precision THEN 350
      WHEN t.gr_250_299 > 0::numeric::double precision THEN 300
      WHEN t.gr_200_249 > 0::numeric::double precision THEN 150
      WHEN t.gr_150_199 > 0::numeric::double precision THEN 200
```

```
            WHEN t.gr_100_149 > 0::numeric::double precision THEN 150
            WHEN t.gr_000_100 > 0::numeric::double precision THEN 100
            ELSE 9999999
        END AS gr_max,
        CASE
            WHEN t.smoc_mort_000_200 > 0::numeric::double precision THEN 0
            WHEN t.smoc_mort_200_299 > 0::numeric::double precision THEN 200
            WHEN t.smoc_mort_300_399 > 0::numeric::double precision THEN 300
            WHEN t.smoc_mort_400_499 > 0::numeric::double precision THEN 400
            WHEN t.smoc_mort_500_599 > 0::numeric::double precision THEN 500
            WHEN t.smoc_mort_600_699 > 0::numeric::double precision THEN 600
            WHEN t.smoc_mort_700_799 > 0::numeric::double precision THEN 700
            WHEN t.smoc_mort_800_899 > 0::numeric::double precision THEN 800
            WHEN t.smoc_mort_900_999 > 0::numeric::double precision THEN 900
            WHEN t.smoc_mort_1000_1249 > 0::numeric::double precision THEN 1000
            WHEN t.smoc_mort_1250_1499 > 0::numeric::double precision THEN 1250
            WHEN t.smoc_mort_1500_1999 > 0::numeric::double precision THEN 1500
            WHEN t.smoc_mort_2000_2499 > 0::numeric::double precision THEN 2000
            WHEN t.smoc_mort_2500_2999 > 0::numeric::double precision THEN 2500
            WHEN t.smoc_mort_3000_plus > 0::numeric::double precision THEN 3000
            ELSE 0
        END AS smoc_min,
        CASE
            WHEN t.smoc_mort_3000_plus > 0::numeric::double precision THEN 3500
            WHEN t.smoc_mort_2500_2999 > 0::numeric::double precision THEN 3000
            WHEN t.smoc_mort_2000_2499 > 0::numeric::double precision THEN 2500
            WHEN t.smoc_mort_1500_1999 > 0::numeric::double precision THEN 2000
            WHEN t.smoc_mort_1250_1499 > 0::numeric::double precision THEN 1500
            WHEN t.smoc_mort_1000_1249 > 0::numeric::double precision THEN 1250
            WHEN t.smoc_mort_900_999 > 0::numeric::double precision THEN 1000
            WHEN t.smoc_mort_800_899 > 0::numeric::double precision THEN 900
            WHEN t.smoc_mort_700_799 > 0::numeric::double precision THEN 800
            WHEN t.smoc_mort_600_699 > 0::numeric::double precision THEN 700
            WHEN t.smoc_mort_500_599 > 0::numeric::double precision THEN 600
            WHEN t.smoc_mort_400_499 > 0::numeric::double precision THEN 500
            WHEN t.smoc_mort_300_399 > 0::numeric::double precision THEN 400
            WHEN t.smoc_mort_200_299 > 0::numeric::double precision THEN 300
            WHEN t.smoc_mort_000_200 > 0::numeric::double precision THEN 200
            ELSE 9999999
        END AS smoc_max,
        CASE
            WHEN t.gr_000_100::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 50
            WHEN (t.gr_000_100 + t.gr_100_149)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN
125
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199)::numeric > (0.10 *
t.gr_cash_rent::numeric) THEN 175
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249)::numeric > (0.10 *
t.gr_cash_rent::numeric) THEN 225
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 +
t.gr_250_299)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 175
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 325
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 375
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 425
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499)::numeric > (0.10 *
t.gr_cash_rent::numeric) THEN 475
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549)::numeric > (0.10 *
t.gr_cash_rent::numeric) THEN 525
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599)::numeric
> (0.10 * t.gr_cash_rent::numeric) THEN 575
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 625
```

```
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 675
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 725
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799)::numeric > (0.10 *
t.gr_cash_rent::numeric) THEN 775
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899)::numeric > (0.10 *
t.gr_cash_rent::numeric) THEN 850
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999)::numeric
> (0.10 * t.gr_cash_rent::numeric) THEN 950
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999 +
t.gr_1000_1249)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 1125
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999 +
t.gr_1000_1249 + t.gr_1250_1499)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN 1375
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999 +
t.gr_1000_1249 + t.gr_1250_1499 + t.gr_1500_1999)::numeric > (0.10 * t.gr_cash_rent::numeric) THEN
1750
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999 +
t.gr_1000_1249 + t.gr_1250_1499 + t.gr_1500_1999 + t.gr_2000_plus)::numeric > (0.10 *
t.gr_cash_rent::numeric) THEN 2250
            ELSE 0
        END AS gr_10pctile,
        CASE
            WHEN t.gr_000_100::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 50
            WHEN (t.gr_000_100 + t.gr_100_149)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN
125
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199)::numeric > (0.90 *
t.gr_cash_rent::numeric) THEN 175
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249)::numeric > (0.90 *
t.gr_cash_rent::numeric) THEN 225
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 +
t.gr_250_299)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 175
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 325
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 375
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 425
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499)::numeric > (0.90 *
t.gr_cash_rent::numeric) THEN 475
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549)::numeric > (0.90 *
t.gr_cash_rent::numeric) THEN 525
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599)::numeric
> (0.90 * t.gr_cash_rent::numeric) THEN 575
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 625
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 675
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 725
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799)::numeric > (0.90 *
t.gr_cash_rent::numeric) THEN 775
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899)::numeric > (0.90 *
t.gr_cash_rent::numeric) THEN 850
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999)::numeric
> (0.90 * t.gr_cash_rent::numeric) THEN 950
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999 +
t.gr_1000_1249)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 1125
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999 +
t.gr_1000_1249 + t.gr_1250_1499)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN 1375
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999 +
t.gr_1000_1249 + t.gr_1250_1499 + t.gr_1500_1999)::numeric > (0.90 * t.gr_cash_rent::numeric) THEN
1750
            WHEN (t.gr_000_100 + t.gr_100_149 + t.gr_150_199 + t.gr_200_249 + t.gr_250_299 +
t.gr_300_349 + t.gr_350_399 + t.gr_400_449 + t.gr_450_499 + t.gr_500_549 + t.gr_550_599 +
t.gr_600_649 + t.gr_650_699 + t.gr_700_749 + t.gr_750_799 + t.gr_800_899 + t.gr_900_999 +
t.gr_1000_1249 + t.gr_1250_1499 + t.gr_1500_1999 + t.gr_2000_plus)::numeric > (0.90 *
t.gr_cash_rent::numeric) THEN 2250
```

```
            ELSE 9999999
        END AS gr_90pctile,
        CASE
            WHEN t.smoc_mort_000_200::numeric > (0.10 * t.smoc_hu_mort::numeric) THEN 100
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299)::numeric > (0.10 *
t.smoc_hu_mort::numeric) THEN 250
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399)::numeric >
(0.10 * t.smoc_hu_mort::numeric) THEN 350
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499)::numeric > (0.10 * t.smoc_hu_mort::numeric) THEN 450
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599)::numeric > (0.10 * t.smoc_hu_mort::numeric) THEN 550
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699)::numeric > (0.10 *
t.smoc_hu_mort::numeric) THEN 650
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799)::numeric >
(0.10 * t.smoc_hu_mort::numeric) THEN 750
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899)::numeric > (0.10 * t.smoc_hu_mort::numeric) THEN 850
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999)::numeric > (0.10 * t.smoc_hu_mort::numeric) THEN 950
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249)::numeric > (0.10 *
t.smoc_hu_mort::numeric) THEN 1125
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 +
t.smoc_mort_1250_1499)::numeric > (0.10 * t.smoc_hu_mort::numeric) THEN 1375
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 + t.smoc_mort_1250_1499 +
t.smoc_mort_1500_1999)::numeric > (0.10 * t.smoc_hu_mort::numeric) THEN 1750
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 + t.smoc_mort_1250_1499 +
t.smoc_mort_1500_1999 + t.smoc_mort_2000_2499)::numeric > (0.10 * t.smoc_hu_mort::numeric) THEN
2250
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 + t.smoc_mort_1250_1499 +
t.smoc_mort_1500_1999 + t.smoc_mort_2000_2499 + t.smoc_mort_2500_2999)::numeric > (0.10 *
t.smoc_hu_mort::numeric) THEN 2750
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 + t.smoc_mort_1250_1499 +
t.smoc_mort_1500_1999 + t.smoc_mort_2000_2499 + t.smoc_mort_2500_2999 +
t.smoc_mort_3000_plus)::numeric > (0.10 * t.smoc_hu_mort::numeric) THEN 3250
            ELSE 0
        END AS smoc_10pctile,
        CASE
            WHEN t.smoc_mort_000_200::numeric > (0.90 * t.smoc_hu_mort::numeric) THEN 100
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299)::numeric > (0.90 *
t.smoc_hu_mort::numeric) THEN 250
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399)::numeric >
(0.90 * t.smoc_hu_mort::numeric) THEN 350
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499)::numeric > (0.90 * t.smoc_hu_mort::numeric) THEN 450
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599)::numeric > (0.90 * t.smoc_hu_mort::numeric) THEN 550
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699)::numeric > (0.90 *
t.smoc_hu_mort::numeric) THEN 650
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799)::numeric >
```

```
(0.90 * t.smoc_hu_mort::numeric) THEN 750
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899)::numeric > (0.90 * t.smoc_hu_mort::numeric) THEN 850
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999)::numeric > (0.90 * t.smoc_hu_mort::numeric) THEN 950
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249)::numeric > (0.90 *
t.smoc_hu_mort::numeric) THEN 1125
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 +
t.smoc_mort_1250_1499)::numeric > (0.90 * t.smoc_hu_mort::numeric) THEN 1375
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 + t.smoc_mort_1250_1499 +
t.smoc_mort_1500_1999)::numeric > (0.90 * t.smoc_hu_mort::numeric) THEN 1750
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 + t.smoc_mort_1250_1499 +
t.smoc_mort_1500_1999 + t.smoc_mort_2000_2499)::numeric > (0.90 * t.smoc_hu_mort::numeric) THEN
2250
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 + t.smoc_mort_1250_1499 +
t.smoc_mort_1500_1999 + t.smoc_mort_2000_2499 + t.smoc_mort_2500_2999)::numeric > (0.90 *
t.smoc_hu_mort::numeric) THEN 2750
            WHEN (t.smoc_mort_000_200 + t.smoc_mort_200_299 + t.smoc_mort_300_399 +
t.smoc_mort_400_499 + t.smoc_mort_500_599 + t.smoc_mort_600_699 + t.smoc_mort_700_799 +
t.smoc_mort_800_899 + t.smoc_mort_900_999 + t.smoc_mort_1000_1249 + t.smoc_mort_1250_1499 +
t.smoc_mort_1500_1999 + t.smoc_mort_2000_2499 + t.smoc_mort_2500_2999 +
t.smoc_mort_3000_plus)::numeric > (0.90 * t.smoc_hu_mort::numeric) THEN 3250
```

```
            ELSE 9999999
        END AS smoc_90pctile, s.tas / 4046.86 AS tas_acres, s.trips_per_week AS tas_trips,
        CASE
            WHEN le_mass.tab_4x1_no_mil_projected_2010 IS NULL THEN le.c000::double precision / 
(0.000247105::double precision * le.aland)
            ELSE le.c000::double precision / (0.000247105::double precision * le.aland) + 
le_mass.tab_4x1_no_mil_projected_2010::double precision / le_mass.acresland
        END AS le_jobs_total_per_acre, le.cns01::double precision / (0.000247105::double precision 
* le.aland) AS le_job_type_01_per_acre, le.cns02::double precision / (0.000247105::double 
precision * le.aland) AS le_job_type_02_per_acre, le.cns03::double precision / 
(0.000247105::double precision * le.aland) AS le_job_type_03_per_acre, le.cns04::double precision 
/ (0.000247105::double precision * le.aland) AS le_job_type_04_per_acre, le.cns05::double 
precision / (0.000247105::double precision * le.aland) AS le_job_type_05_per_acre, 
le.cns06::double precision / (0.000247105::double precision * le.aland) AS 
le_job_type_06_per_acre,
        CASE
            WHEN le_mass.tab4x6_projected_2010 IS NULL THEN le.cns07::double precision / 
(0.000247105::double precision * le.aland)
            ELSE le.cns07::double precision / (0.000247105::double precision * le.aland) + 
le_mass.tab4x6_projected_2010::double precision / le_mass.acresland
        END AS le_job_type_07_per_acre, le.cns08::double precision / (0.000247105::double 
precision * le.aland) AS le_job_type_08_per_acre, le.cns09::double precision / 
(0.000247105::double precision * le.aland) AS le_job_type_09_per_acre, le.cns10::double precision 
/ (0.000247105::double precision * le.aland) AS le_job_type_10_per_acre, le.cns11::double 
precision / (0.000247105::double precision * le.aland) AS le_job_type_11_per_acre, 
le.cns12::double precision / (0.000247105::double precision * le.aland) AS 
le_job_type_12_per_acre, le.cns13::double precision / (0.000247105::double precision * le.aland) 
AS le_job_type_13_per_acre, le.cns14::double precision / (0.000247105::double precision * 
le.aland) AS le_job_type_14_per_acre, le.cns15::double precision / (0.000247105::double precision 
* le.aland) AS le_job_type_15_per_acre, le.cns16::double precision / (0.000247105::double 
precision * le.aland) AS le_job_type_16_per_acre, le.cns17::double precision / 
(0.000247105::double precision * le.aland) AS le_job_type_17_per_acre, le.cns18::double precision 
/ (0.000247105::double precision * le.aland) AS le_job_type_18_per_acre, le.cns19::double 
precision / (0.000247105::double precision * le.aland) AS le_job_type_19_per_acre,
le.cns20::double precision / (0.000247105::double precision * le.aland) AS
le_job_type_20_per_acre,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns01 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_01,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns02 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_02,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns03 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_03,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns04 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_04,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns05 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_05,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns06 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_06,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns07 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_07,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns08 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_08,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns09 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_09,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns10 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_10,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns11 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_11,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns12 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_12,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns13 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_13,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns14 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_14,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns15 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_15,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns16 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_16,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns17 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_17,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns18 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_18,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns19 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_19,
        CASE
            WHEN le.c000 > 0::numeric THEN le.cns20 / le.c000
            ELSE NULL::numeric
        END AS le_pct_type_20,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_1_detached::numeric * 
100::numeric / t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_1_detached,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_1_attached::numeric * 
100::numeric / t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_1_attached,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_2::numeric * 100::numeric / 
t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_2,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_3_4::numeric * 100::numeric / 
t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_3_4,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_5_9::numeric * 100::numeric /
t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_5_9,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_10_19::numeric * 100::numeric
/ t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_10_19,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_20_49::numeric * 100::numeric
/ t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_20_49,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_50_plus::numeric *
100::numeric / t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_50_plus,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_mobile::numeric *
100::numeric / t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_mobil,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.hu_brv::numeric * 100::numeric /
t.hu_total::numeric
            ELSE NULL::numeric
        END AS pct_hu_brvd,
        CASE
            WHEN t.rooms_total > 0::numeric::double precision THEN t.rooms_1::numeric *
100::numeric / t.rooms_total::numeric
            ELSE NULL::numeric
        END AS pct_rooms_1,
        CASE
            WHEN t.rooms_total > 0::numeric::double precision THEN t.rooms_2::numeric * 
100::numeric / t.rooms_total::numeric
            ELSE NULL::numeric
        END AS pct_rooms_2,
        CASE
            WHEN t.rooms_total > 0::numeric::double precision THEN t.rooms_3::numeric * 
100::numeric / t.rooms_total::numeric
            ELSE NULL::numeric
        END AS pct_rooms_3,
        CASE
            WHEN t.rooms_total > 0::numeric::double precision THEN t.rooms_4::numeric * 
100::numeric / t.rooms_total::numeric
            ELSE NULL::numeric
        END AS pct_rooms_4,
        CASE
            WHEN t.rooms_total > 0::numeric::double precision THEN t.rooms_5::numeric * 
100::numeric / t.rooms_total::numeric
            ELSE NULL::numeric
        END AS pct_rooms_5,
        CASE
            WHEN t.rooms_total > 0::numeric::double precision THEN t.rooms_6::numeric * 
100::numeric / t.rooms_total::numeric
            ELSE NULL::numeric
        END AS pct_rooms_6,
        CASE
            WHEN t.rooms_total > 0::numeric::double precision THEN t.rooms_7::numeric * 
100::numeric / t.rooms_total::numeric
            ELSE NULL::numeric
        END AS pct_rooms_7,
        CASE
            WHEN t.rooms_total > 0::numeric::double precision THEN t.rooms_8::numeric * 
100::numeric / t.rooms_total::numeric
```

```
            ELSE NULL::numeric
        END AS pct_rooms_8,
        CASE
            WHEN t.rooms_total > 0::numeric::double precision THEN t.rooms_9plus::numeric * 
100::numeric / t.rooms_total::numeric
            ELSE NULL::numeric
        END AS pct_rooms_9plus, t.median_number_rooms,
        CASE
            WHEN t.hu_total > 0::numeric::double precision THEN t.aggrigate_number_rooms::numeric 
/ t.hu_total::numeric
            ELSE NULL::numeric
        END AS avg_rooms, t.median_rooms_per_renter_hu,
        CASE
            WHEN t.renter_occupied_hu > 0::numeric::double precision THEN 
t.aggrigate_rooms_per_renter_hu::numeric / t.renter_occupied_hu::numeric
            ELSE NULL::numeric
        END AS avg_rooms_renters, t.median_rooms_per_owner_hu,
        CASE
            WHEN t.owner_occupied_hu > 0::numeric::double precision THEN 
t.aggrigate_rooms_per_owner_hu::numeric / t.owner_occupied_hu::numeric
            ELSE NULL::numeric
        END AS avg_rooms_owners, gpxt.gas_price, 20.7 AS mpg, ab.beta, ab.alpha, mv.vmt_transit AS 
model_vmt_with_transit, mv.vmt_no_transit AS model_vmt_no_transit, area.hh_income_hh1, 
area.hh_size_hh1, area.hh_commuters_hh1, area.hh_income_hh2, area.hh_size_hh2, 
area.hh_commuters_hh2, area.hh_income_hh3, area.hh_size_hh3, area.hh_commuters_hh3, 
area.hh_income_hh4, area.hh_size_hh4, area.hh_commuters_hh4, area.hh_income_hh5, area.hh_size_hh5, 
area.hh_commuters_hh5, area.hh_income_hh6, area.hh_size_hh6, area.hh_commuters_hh6, 
area.hh_income_hh7, area.hh_size_hh7, area.hh_commuters_hh7, area.hh_income_hh8, area.hh_size_hh8, 
area.hh_commuters_hh8
   FROM acs_bg_2012_data t
   JOIN blkgrp_gas_region_xtab gpxt ON t.stfid::text = gpxt.stfid::text
   JOIN blkgrp_gravity_2010 l ON t.stfid::text = l.stfid
   JOIN mass_emp_gravity_2010 m ON t.stfid::text = m.stfid::text
   LEFT JOIN mass_d_fix md ON t.stfid::text = md.stfid::text
   LEFT JOIN lai_blkgrp_2010_res_density d ON t.stfid::text = d.stfid
   JOIN blkgrp_2010_area_intpt a ON t.stfid::text = a.stfid
   LEFT JOIN acs_cbsa_2012_data c ON t.cbsa::text = c.cbsa::text
   LEFT JOIN cbsa_2010_data cb ON cb.cbsa::text = t.cbsa::text
   LEFT JOIN bg_tci tci ON t.stfid::text = tci.stfid::text
   LEFT JOIN il_vmt_blkgrp v ON t.stfid::text = v.stfid::text
   JOIN od_jt00_2010_distance_home_blkgrp cd ON t.stfid::bpchar = cd.stfid
   LEFT JOIN lai_blkgrps_tci_tas s ON t.stfid::text = s.stfid::text
   JOIN acs_county_2012_data cnty ON cnty.stfid::text = t.state_county::text
   LEFT JOIN local_employment_2010 le ON le.stfid::text = t.stfid::text
   LEFT JOIN mass_local_employment_2010 le_mass ON le_mass.stfid::text = t.stfid::text
   JOIN blkgrp_incomes bgi ON t.stfid::text = bgi.stfid::text
   LEFT JOIN acs_tract_2012_data trk ON "substring"(t.stfid::text, 1, 11) = trk.stfid::text
   JOIN blkgrp_model_vmt mv ON t.stfid::text = mv.stfid::text
   LEFT JOIN cbsa_transit_alpha ab ON ab.cbsa::text = t.cbsa::text
   JOIN area_hh_types area ON area.stfid = t.state_county::text
  WHERE le.aland >= 0::numeric::double precision;

ALTER TABLE place_blkgrp_data
  OWNER TO nobody;
GRANT ALL ON TABLE place_blkgrp_data TO nobody;
GRANT SELECT ON TABLE place_blkgrp_data TO sbecker;
```

Program 7: SLQ for Creating Data Table for R

SEM Fit (tenure_simple)	**select** stfid,county, area_income_owner_frac , area_income_renter_frac , area_median_hh_income , avg_d , avg_hh_size_owners , avg_hh_size_renters , block_density , commuters_per_hh_owners , commuters_per_hh_renters , emp_gravity , frac_renters , gross_hh_density , le_jobs_total_per_acre , le_job_type_07_per_acre , median_rooms_per_owner_hu , median_rooms_per_renter_hu , pct_hu_1_detached , retail_gravity , autos_per_hh_owners , autos_per_hh_renters , median_gross_rent , median_smoc_mortgage , pct_transit_commuters_owners ,

	pct_transit_commuters_renters **into** tenure_simple.data **from** place_blkgrp_data **where** gross_hh_density>0 and median_gross_rent < 2000 and median_smoc_mortgage < 4000 and county is not NULL and area_income_owner_frac is not NULL and area_income_renter_frac is not NULL and area_median_hh_income is not NULL and avg_d is not NULL and avg_hh_size_owners is not NULL and avg_hh_size_renters is not NULL and block_density is not NULL and commuters_per_hh_owners is not NULL and commuters_per_hh_renters is not NULL and emp_gravity is not NULL and frac_renters is not NULL and gross_hh_density is not NULL and le_jobs_total_per_acre is not NULL and le_job_type_07_per_acre is not NULL and median_rooms_per_owner_hu is not NULL and median_rooms_per_renter_hu is not NULL and pct_hu_1_detached is not NULL and retail_gravity is not NULL and autos_per_hh_owners is not NULL and autos_per_hh_renters is not NULL and median_gross_rent is not NULL and median_smoc_mortgage is not NULL and pct_transit_commuters_owners is not NULL and pct_transit_commuters_renters is not NULL; **ALTER TABLE** tenure_simple.data ADD PRIMARY KEY (stfid);

Autos Use per Household (vmt_no_transit)	**select** stfid,county,vmt_per_hh, area_income_frac , area_median_hh_income , avg_d , avg_hh_size , block_density , commuters_per_hh , emp_gravity , frac_renters , gross_hh_density , le_jobs_total_per_acre , le_job_type_07_per_acre , median_number_rooms , pct_hu_1_detached , retail_gravity
	into vmt_no_transit.data
	from place_blkgrp_data
	where gross_hh_density>0 and num_autos > 10 and not(vmt_per_hh is NULL) and not(county is NULL) and not(area_income_frac is NULL) and not(area_median_hh_income is NULL) and not(avg_d is NULL) and not(avg_hh_size is NULL) and not(block_density is NULL) and not(commuters_per_hh is NULL) and not(emp_gravity is NULL) and not(frac_renters is NULL) and not(gross_hh_density is NULL) and not(le_jobs_total_per_acre is NULL) and not(le_job_type_07_per_acre is NULL) and not(median_number_rooms is NULL) and not(pct_hu_1_detached is NULL) and not(retail_gravity is NULL);
	ALTER TABLE vmt_no_transit.data ADD PRIMARY KEY (stfid);

Program 8: R Code for Linear Transformations

```
trans_func<-function(x,tf){
  y<-x
  if(tf==1) {y<-sqrt(x)}
  if(tf==2) {y<-log(x)}
  y
}

invst_func<-function(y,tf){
  x<-y
  if(tf==1) {x<-(y*y)}
  if(tf==2) {x<-exp(y)}
  x
}
```

Program 9: Finding R^2 if a Distribution is Normal

```
is_norm<-function(x,xlb='x'){
        n<-rnorm(length(x),mean(x),sd(x))
        mn<-min(n)
        mx<-max(n)
        inv<-(mx-mn)/100.0
        mn_b<-0
        while(mn_b>mn){mn_b<- mn_b - inv}
        i<-0
        bmn<-mn_b
        while(bmn<mx){
          i<-i+1
          bmn<-bmn+inv
        }
        nx<-array(data=0,dim=i)
        nn<-array(data=0,dim=i)
        i<-0
        while(mn_b<mx){
          i<-i+1
          mx_b<-mn_b+inv
          nx[i]<-length(subset(x,x>=mn_b & x<mx_b))
          nn[i]<-length(subset(n,n>=mn_b & n<mx_b))
          mn_b<-mx_b
        }
        f<-lm(nx~ 0 + nn)
        r2<-find_r2(nx,f$residuals)
        r2<-round(r2*1000)/10
        mn<-min(c(min(x),min(n)))
        mx<-max(c(max(x),max(n)))
        hx <- hist(x,breaks=100,xlim=c(mn,mx),col=rgb(0,0,1,1/4),border=rgb(0,0,1,1/4), main =
paste("Normal R2 =" , r2,"%"),xlab=xlb)
        hn <- hist(n,breaks=100,xlim=c(mn,mx),col=rgb(1,0,0,1/8),border=rgb(1,0,0,1/8), add=T)
        r2
}

find_r2<-function(meas_y,resid){
        (1 - sum((resid)**2)/sum((meas_y-mean(meas_y))**2))
}
```

Appendix D: Choosing Variables and Interaction Terms

Program 10: R Code to Find Loop over Variables and Find An Optimal Set

```
flex_fit<-function(dt,i_ndx,i_cluster,i_y,nm_xs,avd=c(),clustering='not'){
  numx<-length(nm_xs)
  max_var<-numx + ((numx*numx)+numx)/2
  reslts<-
array(data=NA,dim=c((max_var+1),8),dimnames=list(1:(max_var+1),c('name','value','error','abs_t','p
rob','vif','avoid','noodled')))
  if(length(avd)==0){reslts[,'avoid']<-0}
  else{reslts[,'avoid']<-avd}
  nm<-names(dt)
  noodleing='start'
  times_through=0
  while(noodleing!='stop'){
    mx_p<-1
    mx_v<-100
    mx_vg<-20.00
    mx_pg<-0.05
    go="go"
    while(go!="stop"){
      j<-1
      jj<-2
      n_term<-1
      f<-paste(nm[i_y]," ~")
      plus<-""
      while(j<=numx){
        aa<-reslts[jj,'avoid']
        if(aa!=1){
          f<-paste(f,plus,"I(",nm_xs[j],"*1",")")
          plus<-"+"
        }
        jj<-jj+1
        i<-j
        while(i<=numx){
          aa<-reslts[jj,'avoid']
          if(aa!=1){
            f<-paste(f,plus,"I(",nm_xs[j],"*",nm_xs[i],")")
            plus<-"+"
          }
          jj<-jj+1
          i<-i+1
        }
        j<-j+1
      }
      fit<-lm(f,data=dt)
      s<-summary(fit)
      r2<-round((100*(s$r.squared)),4)
      if(clustering=='do'){
        cat('clustering','\n')
        ols<-ols_cluster_error(fit, dt, cluster=names(dt)[i_cluster],digits=3)
      }
      else{
        cat('not clustering','\n')
        ols<-s$coefficients
      }
      ols_nm<-names(ols[,1])
      if(numx>1){vf<-vif(fit)}
      mx_p<-0
      mx_pn<-0
      mx_v<-0
      mx_vn<-0
      j<-1
      i<-1
      while(j<=max_var+1){
        aa<-reslts[j,'avoid']
        if(aa!=1){
          reslts[j,'name']<-ols_nm[i]
          reslts[j,'value']<-ols[i,1]
          reslts[j,'error']<-ols[i,2]
          reslts[j,'abs_t']<-abs(ols[i,3])
          reslts[j,'prob']<-ols[i,4]
          if(i>1 && numx>1){reslts[j,'vif']<-vf[i-1]}
          else{reslts[j,'vif']<-0}
          if(j != 1){
            if(mx_v <= as.numeric(reslts[j,'vif'])){
              mx_v<-as.numeric(reslts[j,'vif'])
              mx_vn<-j
            }
            if(mx_p <= as.numeric(reslts[j,'prob'])){
              mx_p<-as.numeric(reslts[j,'prob'])
              mx_pn<-j
            }
          }
          i<-i+1
        }
        else{
          reslts[j,'value']<-0
          reslts[j,'error']<-0
          reslts[j,'abs_t']<-0
          reslts[j,'prob']<-0
          reslts[j,'vif']<-0
        }
        j<-j+1
      }
      cat("mx_p=",mx_p,mx_pg,"mx_pn=",mx_pn,"mx_v=",mx_v,mx_vg,"mx_vn=",mx_vn,"\n")
      cat("In Fiter ",reslts[,'avoid'],"\n")
      go<-"stop"
      if(mx_p>=mx_pg){
        reslts[mx_pn,'avoid']<-1
        go<-"go"
      }
      else{
        if(mx_v>=mx_vg){
        reslts[mx_vn,'avoid']<-1
        go<-"go"
      }
    }
    cat("Done with this Iteration ",reslts[,'avoid'],"\n")
    if(go=="stop"){if(clustering=='not'){
      go<-"go"
      clustering<-'do'
    }
  }
  cat(" r2=",r2,"% \n")
  if(times_through==0){r2_first=r2}
  times_through<-times_through+1
}
if(noodleing=='start'){
  noodle<-reslts[,'avoid']
  noodleing<-'go'
  try_noodle=0
  r2_old<-r2
  reslts_old<-reslts
  cat('beginning to noodle','\n')
  }
    if(noodleing=='go'){
      if(r2>r2_old){
        r2_old<-r2
        reslts_old<-reslts
      }
      else{
        r2<-r2_old
        reslts<-reslts_old
      }
      found_new_noodle='no'
      while(found_new_noodle=='no'){
        try_noodle=try_noodle+1
        if(try_noodle>max_var+1){
          found_new_noodle='nix'
          noodleing<-'stop'
        }
        else{if(noodle[try_noodle]==1){
          noodle[try_noodle]<-0
          reslts[try_noodle,'avoid']<-0
          found_new_noodle='yes'
        }
      }
    }
      cat('found noodel ',found_new_noodle,' | ',try_noodle,'\n')
    }
  }
  cat("final r2=",r2,'% there were ',times_through,' iterations and original r2 =
',r2_first,'%\n')
reslts
}
```

Appendix E: Final Fits

Program 11: R code for Final Fits of models

SEM

```
#!/usr/bin/env Rscript
source('model_sem_fit.r')
 con <- dbConnect(PostgreSQL(), host="localhost", user= "nobody", password="h1ggs",
dbname="hud_lai_2014")
 htd<- dbGetQuery(con,'select * from tenure_simple.data')
 tfd <- htd[c("stfid","county")]
 tfd$area_income_owner_frac <- trans_func((htd$area_income_owner_frac),2)
 this_mean <- mean(tfd$area_income_owner_frac)
 cat(this_mean)
 this_sd <- sd(tfd$area_income_owner_frac)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='area_income_owner_frac'"))
 tfd$area_income_owner_frac <- (tfd$area_income_owner_frac-this_mean)/this_sd
 tfd$area_income_renter_frac <- trans_func((htd$area_income_renter_frac),2)
 this_mean <- mean(tfd$area_income_renter_frac)
 cat(this_mean)
 this_sd <- sd(tfd$area_income_renter_frac)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='area_income_renter_frac'"))
 tfd$area_income_renter_frac <- (tfd$area_income_renter_frac-this_mean)/this_sd
 tfd$area_median_hh_income <- trans_func((htd$area_median_hh_income),2)
 this_mean <- mean(tfd$area_median_hh_income)
 cat(this_mean)
 this_sd <- sd(tfd$area_median_hh_income)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='area_median_hh_income'"))
 tfd$area_median_hh_income <- (tfd$area_median_hh_income-this_mean)/this_sd
 tfd$avg_d <- trans_func((htd$avg_d),2)
 this_mean <- mean(tfd$avg_d)
 cat(this_mean)
 this_sd <- sd(tfd$avg_d)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='avg_d'"))
 tfd$avg_d <- (tfd$avg_d-this_mean)/this_sd
 tfd$avg_hh_size_owners <- trans_func((htd$avg_hh_size_owners),2)
 this_mean <- mean(tfd$avg_hh_size_owners)
 cat(this_mean)
 this_sd <- sd(tfd$avg_hh_size_owners)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='avg_hh_size_owners'"))
 tfd$avg_hh_size_owners <- (tfd$avg_hh_size_owners-this_mean)/this_sd
 tfd$avg_hh_size_renters <- trans_func((htd$avg_hh_size_renters),2)
 this_mean <- mean(tfd$avg_hh_size_renters)
 cat(this_mean)
 this_sd <- sd(tfd$avg_hh_size_renters)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='avg_hh_size_renters'"))
 tfd$avg_hh_size_renters <- (tfd$avg_hh_size_renters-this_mean)/this_sd
 tfd$block_density <- trans_func((htd$block_density),1)
 this_mean <- mean(tfd$block_density)
 cat(this_mean)
 this_sd <- sd(tfd$block_density)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='block_density'"))
 tfd$block_density <- (tfd$block_density-this_mean)/this_sd
 tfd$commuters_per_hh_owners <- trans_func((htd$commuters_per_hh_owners),0)
 this_mean <- mean(tfd$commuters_per_hh_owners)
 cat(this_mean)
 this_sd <- sd(tfd$commuters_per_hh_owners)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='commuters_per_hh_owners'"))
```

```
 tfd$commuters_per_hh_owners <- (tfd$commuters_per_hh_owners-this_mean)/this_sd
 tfd$commuters_per_hh_renters <- trans_func((htd$commuters_per_hh_renters),0)
 this_mean <- mean(tfd$commuters_per_hh_renters)
 cat(this_mean)
 this_sd <- sd(tfd$commuters_per_hh_renters)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='commuters_per_hh_renters'"))
 tfd$commuters_per_hh_renters <- (tfd$commuters_per_hh_renters-this_mean)/this_sd
 tfd$emp_gravity <- trans_func((htd$emp_gravity),2)
 this_mean <- mean(tfd$emp_gravity)
 cat(this_mean)
 this_sd <- sd(tfd$emp_gravity)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='emp_gravity'"))
 tfd$emp_gravity <- (tfd$emp_gravity-this_mean)/this_sd
 tfd$frac_renters <- trans_func((htd$frac_renters),1)
 this_mean <- mean(tfd$frac_renters)
 cat(this_mean)
 this_sd <- sd(tfd$frac_renters)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='frac_renters'"))
 tfd$frac_renters <- (tfd$frac_renters-this_mean)/this_sd
 tfd$gross_hh_density <- trans_func((htd$gross_hh_density),1)
 this_mean <- mean(tfd$gross_hh_density)
 cat(this_mean)
 this_sd <- sd(tfd$gross_hh_density)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='gross_hh_density'"))
 tfd$gross_hh_density <- (tfd$gross_hh_density-this_mean)/this_sd
 tfd$le_jobs_total_per_acre <- trans_func((htd$le_jobs_total_per_acre),1)
 this_mean <- mean(tfd$le_jobs_total_per_acre)
 cat(this_mean)
 this_sd <- sd(tfd$le_jobs_total_per_acre)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='le_jobs_total_per_acre'"))
 tfd$le_jobs_total_per_acre <- (tfd$le_jobs_total_per_acre-this_mean)/this_sd
 tfd$le_job_type_07_per_acre <- trans_func((htd$le_job_type_07_per_acre),1)
 this_mean <- mean(tfd$le_job_type_07_per_acre)
 cat(this_mean)
 this_sd <- sd(tfd$le_job_type_07_per_acre)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='le_job_type_07_per_acre'"))
 tfd$le_job_type_07_per_acre <- (tfd$le_job_type_07_per_acre-this_mean)/this_sd
 tfd$median_rooms_per_owner_hu <- trans_func((htd$median_rooms_per_owner_hu),0)
 this_mean <- mean(tfd$median_rooms_per_owner_hu)
 cat(this_mean)
 this_sd <- sd(tfd$median_rooms_per_owner_hu)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='median_rooms_per_owner_hu'"))
 tfd$median_rooms_per_owner_hu <- (tfd$median_rooms_per_owner_hu-this_mean)/this_sd
 tfd$median_rooms_per_renter_hu <- trans_func((htd$median_rooms_per_renter_hu),0)
 this_mean <- mean(tfd$median_rooms_per_renter_hu)
 cat(this_mean)
 this_sd <- sd(tfd$median_rooms_per_renter_hu)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='median_rooms_per_renter_hu'"))
 tfd$median_rooms_per_renter_hu <- (tfd$median_rooms_per_renter_hu-this_mean)/this_sd
 tfd$pct_hu_1_detached <- trans_func((htd$pct_hu_1_detached),0)
 this_mean <- mean(tfd$pct_hu_1_detached)
 cat(this_mean)
 this_sd <- sd(tfd$pct_hu_1_detached)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='pct_hu_1_detached'"))
 tfd$pct_hu_1_detached <- (tfd$pct_hu_1_detached-this_mean)/this_sd
 tfd$retail_gravity <- trans_func((htd$retail_gravity),2)
 this_mean <- mean(tfd$retail_gravity)
 cat(this_mean)
 this_sd <- sd(tfd$retail_gravity)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='retail_gravity'"))
 tfd$retail_gravity <- (tfd$retail_gravity-this_mean)/this_sd
 tfd$autos_per_hh_owners <- trans_func((htd$autos_per_hh_owners),0)
 this_mean <- mean(tfd$autos_per_hh_owners)
 cat(this_mean)
 this_sd <- sd(tfd$autos_per_hh_owners)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='autos_per_hh_owners'"))
 tfd$autos_per_hh_owners <- (tfd$autos_per_hh_owners-this_mean)/this_sd
 tfd$autos_per_hh_renters <- trans_func((htd$autos_per_hh_renters),0)
 this_mean <- mean(tfd$autos_per_hh_renters)
 cat(this_mean)
 this_sd <- sd(tfd$autos_per_hh_renters)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='autos_per_hh_renters'"))
 tfd$autos_per_hh_renters <- (tfd$autos_per_hh_renters-this_mean)/this_sd
 tfd$median_gross_rent <- trans_func((htd$median_gross_rent),2)
 this_mean <- mean(tfd$median_gross_rent)
 cat(this_mean)
 this_sd <- sd(tfd$median_gross_rent)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='median_gross_rent'"))
 tfd$median_gross_rent <- (tfd$median_gross_rent-this_mean)/this_sd
 tfd$median_smoc_mortgage <- trans_func((htd$median_smoc_mortgage),2)
 this_mean <- mean(tfd$median_smoc_mortgage)
 cat(this_mean)
 this_sd <- sd(tfd$median_smoc_mortgage)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='median_smoc_mortgage'"))
 tfd$median_smoc_mortgage <- (tfd$median_smoc_mortgage-this_mean)/this_sd
 tfd$pct_transit_commuters_owners <- trans_func((htd$pct_transit_commuters_owners),0)
 this_mean <- mean(tfd$pct_transit_commuters_owners)
 cat(this_mean)
 this_sd <- sd(tfd$pct_transit_commuters_owners)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='pct_transit_commuters_owners'"))
 tfd$pct_transit_commuters_owners <- (tfd$pct_transit_commuters_owners-this_mean)/this_sd
 tfd$pct_transit_commuters_renters <- trans_func((htd$pct_transit_commuters_renters),0)
 this_mean <- mean(tfd$pct_transit_commuters_renters)
 cat(this_mean)
 this_sd <- sd(tfd$pct_transit_commuters_renters)
 dbGetQuery(con,paste('update tenure_simple.model_variables set v_mean=',this_mean,',
v_std=',this_sd," where v_name='pct_transit_commuters_renters'"))
 tfd$pct_transit_commuters_renters <- (tfd$pct_transit_commuters_renters-
this_mean)/this_sd
  nm <- names(tfd)[4:length(names(tfd))]
 print('------- getting in_model --------')
 in_model<- dbGetQuery(con,'select * from tenure_simple.model_form')
 print('------- about to fit--------')
 fit<-sem_fit(tfd,in_model,intercpt=FALSE)
 summary(fit, rsquare=TRUE)
 print('------- did fit--------')
 print('------- about to save Coefs and R2--------')
save_sem_fit(con,'tenure_simple',fit,in_model)
 print('------- Saved Coefs and R2--------')
png(filename="./hud_lai_2014/tenure_simple/path.png", height=2000, width=1600,bg="white")
 semPaths(fit, residuals = FALSE,intercepts = FALSE, layout = 'tree',rotation=2,what =
'std',color=c( 'lightyellow', 'lightyellow', 'lightyellow', 'lightyellow', 'lightyellow',
'lightyellow', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue',
'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue',
'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue'
),'no',exoCov=FALSE,sizeMan = 10, sizeMan2 = 3,nCharNodes=0, thresholds = FALSE, reorder
= TRUE, curvePivot = TRUE, structural=FALSE)
dev.off()
png(filename="./hud_lai_2014/tenure_simple/path_spring.png", height=2000,
width=1600,bg="white")
 semPaths(fit, residuals = FALSE,intercepts = FALSE, layout = 'spring',what =
```

	'std',color=c('lightyellow', 'lightyellow', 'lightyellow', 'lightyellow', 'lightyellow', 'lightyellow', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue'),'no',exoCov=FALSE,sizeMan = 10, sizeMan2 = 3,nCharNodes=0, thresholds = FALSE, reorder = TRUE, curvePivot = TRUE, structural=FALSE)
	dev.off()
	png(filename="./hud_lai_2014/tenure_simple/path_circle.png", height=2000, width=1600,bg="white")
	 semPaths(fit, residuals = FALSE,intercepts = FALSE, layout = 'circle2',what = 'std',color=c('lightyellow', 'lightyellow', 'lightyellow', 'lightyellow', 'lightyellow', 'lightyellow', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue', 'lightblue'),'no',exoCov=FALSE,sizeMan = 10, sizeMan2 = 3,nCharNodes=0, thresholds = FALSE, reorder = TRUE, curvePivot = TRUE, structural=FALSE)
	dev.off()
	 print('------- Everything Done--------')

Autos Use per Household	#!/usr/bin/env Rscript source('model_flex_fit.r') con <- dbConnect(PostgreSQL(), host="localhost", user= "nobody", password="h1ggs", dbname="hud_lai_2014") htd<- dbGetQuery(con,'select * from vmt_no_transit.data') tfd <- htd[c("stfid","county")] tfd$factor <- tfd$county tfd$vmt_per_hh <- trans_func((htd$vmt_per_hh),0) tfd$area_income_frac <- trans_func((htd$area_income_frac),3) tfd$area_median_hh_income <- trans_func((htd$area_median_hh_income),2) tfd$avg_d <- trans_func((htd$avg_d),2) tfd$avg_hh_size <- trans_func((htd$avg_hh_size),0) tfd$block_density <- trans_func((htd$block_density),1) tfd$commuters_per_hh <- trans_func((htd$commuters_per_hh),1) tfd$emp_gravity <- trans_func((htd$emp_gravity),2) tfd$frac_renters <- trans_func((htd$frac_renters),0)

```
 tfd$gross_hh_density <- trans_func((htd$gross_hh_density),2)
 tfd$le_jobs_total_per_acre <- trans_func((htd$le_jobs_total_per_acre),2)
 tfd$le_job_type_07_per_acre <- trans_func((htd$le_job_type_07_per_acre),1)
 tfd$median_number_rooms <- trans_func((htd$median_number_rooms),0)
 tfd$pct_hu_1_detached <- trans_func((htd$pct_hu_1_detached),0)
 tfd$retail_gravity <- trans_func((htd$retail_gravity),2)
  nm <- names(tfd)[5:length(names(tfd))]
 opt<-'none'
 fit<-flex_fit(tfd,1,2,0,4,nm,'vmt_no_transit',con,opt)
 nm_clstr <-names(tfd)[2]
 if(opt == 'none') {nm_clstr <- '' }
 ff<-fit_results(fit,tfd,nm,nm_clstr)
 fff<-subset(ff,ff[,'avoid']==0)
 cat('---- OUT ----','
')
n_coef=length(fff[,'name'])
  n<-1
  while(n <= n_coef){
    cat('name|',n,'|',fff[n,'name'],'|')
    cat('value|',n,'|',fff[n,'value'],'|')
    cat('error|',n,'|',fff[n,'error'],'|')
    cat('abs_t|',n,'|',fff[n,'abs_t'],'|')
    cat('prob|',n,'|',fff[n,'prob'],'|')
    cat('vif|',n,'|',fff[n,'vif'],'|')
    n <- n+1
  }
   cat('---- STOP ---','
')
 nf<-load_coef_db(con,'vmt_no_transit',fff)
 y <- htd$vmt_per_hh
 pre_fit <- predict(fit, se.fit=TRUE)
 m <- invst_func(pre_fit$fit,0)
 m_err <- invst_func(pre_fit$se.fit,0)
 polyy<-array(data=c(0),dim=4)
 r <- y-m
 htd$r <- y-m
 r2<-find_r2(y,r)
 r2
 min_y <- min(y)
 max_y <- max(y)
 min_m <- min(m)
 max_m <- max(m)
png(filename="./hud_lai_2014/vmt_no_transit/mod_meas.png", height=250,
width=250,bg="white")
plot(m,y,type="p", pch='.',
col="blue",ylim=c(min_y,max_y),xlim=c(min_y,max_y),xlab="Modeled",ylab="Measured")
dev.off()
 print('------- Writing stuff --------')
png(filename="./hud_lai_2014/vmt_no_transit/resid_meas.png", height=250,
width=250,bg="white")
xx<-error_plot(y,r,xl='vmt_per_hh Measured',yl='vmt_per_hh Residual',sigma=2.5)
 prmatrix(xx,collab =paste("Fit Qual. Stat ~"))
 print('------- Done --------')
dev.off()
png(filename="./hud_lai_2014/vmt_no_transit/histp.png", height=250, width=250,bg="white")
sdev<-sd(m)
mn<-mean(m)
nn<-length(m)
tit<-paste('<x>|sd',round(mn,2),'|',round(sdev,2))
 hist(m, breaks=20,plot = TRUE,border="forestgreen",main=tit,
xlab='Model',xlim=c(min_y,max_y))
 dev.off()
png(filename="./hud_lai_2014/vmt_no_transit/histd.png", height=250, width=250,bg="white")
sdev<-sd(y)
mn<-mean(y)
nn<-length(y)
tit<-paste('<x>|sd|n',round(mn,2),'|',round(sdev,2),'|',nn,sep='')
 hist(y, breaks=20,plot = TRUE,border="blue",main=tit,
xlab='Measured',xlim=c(min_y,max_y))
 dev.off()
png(filename="./hud_lai_2014/vmt_no_transit/histr.png", height=250, width=250,bg="white")
sdev<-sd(r)
mn<-mean(r)
nn<-length(r)
tit<-paste('<x>|sd',round(mn,2),'|',round(sdev,2))
 hist(r, breaks=20,plot = TRUE,border="forestgreen",main=tit, xlab='Residual')
 dev.off()
 otpt <- htd[c("stfid","vmt_per_hh")]
 otpt$p <- round(m,digits = 6)
 otpt$r <- round(r,digits = 6)
 otpt$m_err <- round(m_err,digits = 6)
 print('------- Saving Residuals in DB --------')
 make_write_table(con,'vmt_no_transit','resid',1,otpt)
 print('------- Everything Done--------')
```

Program 12: OLS with Cluster Error Estimation

```
#
# this is a striped down version of the function I got from
http://diffuseprior.wordpress.com/2012/06/15/standard-robust-and-clustered-standard-errors-
computed-in-r
# on using clustered errors...
# with only the clutered error part
#
ols_cluster_error <- function(r1, data, cluster=NULL,digits=3){
#
  X <- model.matrix(r1)
  n <- dim(X)[1]
  k <- dim(X)[2]
  clus <- cbind(X,data[,cluster],resid(r1))
  colnames(clus)[(dim(clus)[2]-1):dim(clus)[2]] <- c(cluster,"resid")
  m <- dim(table(clus[,cluster]))
  dfc <-  (m/(m-1))*((n-1)/(n-k))
  uclust  <- apply(resid(r1)*X,2, function(x) tapply(x, clus[,cluster], sum))
  se <- sqrt(diag(solve(crossprod(X)) %*% (t(uclust) %*% uclust) %*% solve(crossprod(X)))*dfc)
  res <- cbind(coef(r1),se)
  res <- cbind(res,res[,1]/res[,2],(1-pnorm(abs(res[,1]/res[,2])))*2)
  res1 <-
matrix(as.numeric(sprintf(paste("%.",paste(digits,"f",sep=""),sep=""),res)),nrow=dim(res)[1])
  rownames(res1) <- rownames(res)
  colnames(res1) <- c("Estimate","Std. Error","t value","Pr(>|t|)")
  return(res1)
}
```

Program 13: SEM Fitting R Code

```
library(car)
library(RPostgreSQL)
library(lavaan)
library(semPlot)
sem_fit<-function(df,in_model,intercpt){
   mod<-array()
   mod['sem']<-"\n"
   ss<-subset(in_model,in_model$ex_lat_lv == 'lv_var')
   lat_v<-unique(ss$end_v)
   n_lat_v<-length(lat_v)
   i<-1
   while(i<=n_lat_v){
      this_lat<-lat_v[i]
      mod[this_lat]<-paste(this_lat," =~ ")
      this_model<-subset(in_model,in_model$end_v == this_lat & in_model$ex_lat_lv == 'lv_var')
      this<-this_model$d_var
      n_this<-length(this)
      j<-1
      plus<-""
      while(j<=n_this){
         mod[this_lat]<-paste(mod[this_lat],plus,this[j])
         plus<-" + "
         j<-j+1
      }
      mod['sem']=paste(mod['sem'],mod[this_lat],'\n')
      i<-i+1
   }

   ss<-subset(in_model,in_model$ex_lat_lv != 'lv_var')
   end_v<-unique(ss$end_v)
   n_end_v<-length(end_v)
   i<-1
   while(i<=n_end_v){
      this_end<-end_v[i]
      cat('now this_end=',this_end,'\n')
      if(intercpt) {mod[this_end]<-paste(this_end," ~ 1 + ")}
      else {mod[this_end]<-paste(this_end," ~ 0 + ")}
      this_model<-subset(in_model,in_model$end_v == this_end & in_model$ex_lat_lv != 'lv_var')
      this<-this_model$d_var
      n_this<-length(this)
      j<-1
      plus<-""
      while(j<=n_this){
         mod[this_end]<-paste(mod[this_end],plus,this[j])
         plus<-" + "
         j<-j+1
      }
      mod['sem']=paste(mod['sem'],mod[this_end],'\n')
      i<-i+1
   }

   cat('here is the model: ',mod['sem'],'\n')
```

```
    sem(mod['sem'],data=df)
}
```

Appendix F: Running Models and Calculation Costs

Program 14: PHP Code to Run Final Model and Costs

```
<?php
include "../../rpgmodel/dbFunctions.inc.php";
$db="hud_lai_2014";
ini_set("memory_limit","1280M");
#
# set up household type
#
# first the local
#
$hht[0]='local';
$ntype=1;
while($ntype<=8){
  $hht[$ntype]="hh$ntype";
  $ntype++;
}
#
# read inputs to see if you want to run model
#
$do_model=$argv[1];
#
# first build all the tables for each household type
#
foreach($hht as $hhtype){

  if($do_model == 'model'){
    echo "building table lai_model_$hhtype \n";
    $income = "hh_income_$hhtype as hh_income_owners,hh_income_$hhtype as hh_income_renters,";
    if($hhtype == 'local'){$income = "median_hh_income_owners as
hh_income_owners,median_hh_income_renters as hh_income_renters,";}
```

```
  exec_sql($db,"drop table lai_model_$hhtype;
   select stfid,state,cbsa,county,$income
              alpha,beta,gas_price,mpg,frac_renters,
    gr_min,
    gr_max,
    smoc_min,
    smoc_max,
    gr_10pctile,
    gr_90pctile,
      smoc_10pctile,
      smoc_90pctile,
--
-- type $hhtype model outputs
--
-- owners
--
      NULL::numeric as autos_per_hh_owners,
      NULL::numeric as model_smoc_with_mortgage,
      NULL::numeric as h_cost_owners,
      NULL::numeric as pct_transit_commuters_owners,
      NULL::numeric as vmt_per_hh_owners,
-- renters
      NULL::numeric as autos_per_hh_renters,
      NULL::numeric as model_gross_rent,
      NULL::numeric as h_cost_renters,
      NULL::numeric as pct_transit_commuters_renters,
      NULL::numeric as vmt_per_hh_renters,
-- combined
      NULL::numeric as autos_per_hh,
      NULL::numeric as h_cost,
      NULL::numeric as pct_transit_commuters,
      NULL::numeric as vmt_per_hh,
--
-- type $hhtype costs and indcies
```

```
--
-- owners
--
      NULL::numeric as auto_own_cost_owners,
      NULL::numeric as vmt_cost_owners,
      NULL::numeric as transit_cost_owners,
      NULL::numeric as transit_trips_owners,
      NULL::numeric as t_cost_owners,
      NULL::numeric as t_owners,
      NULL::numeric as h_owners,
      NULL::numeric as ht_owners,
-- renters
      NULL::numeric as auto_own_cost_renters,
      NULL::numeric as vmt_cost_renters,
      NULL::numeric as transit_cost_renters,
      NULL::numeric as transit_trips_renters,
      NULL::numeric as t_cost_renters,
      NULL::numeric as t_renters,
      NULL::numeric as h_renters,
      NULL::numeric as ht_renters,
-- combined
      NULL::numeric as auto_own_cost,
      NULL::numeric as vmt_cost,
      NULL::numeric as transit_cost,
      NULL::numeric as transit_trips,
      NULL::numeric as t_cost,
      NULL::numeric as t,
      NULL::numeric as h,
      NULL::numeric as ht
    into lai_model_$hhtype
    from lai_model_input_data
    order by stfid;
    ALTER TABLE lai_model_$hhtype
      ADD PRIMARY KEY (stfid);
      CREATE INDEX ON lai_model_$hhtype (cbsa ASC NULLS LAST);
      CREATE INDEX ON lai_model_$hhtype (county ASC NULLS LAST);
      CREATE INDEX ON lai_model_$hhtype (state ASC NULLS LAST);
      ALTER TABLE lai_model_$hhtype
       CLUSTER ON lai_model_".$hhtype."_state_idx;
    ");
#
  echo "modeling household type $hhtype \n";
#
# so now we have table - fill them with the model
#
    $from_owners = "from lai_sem_model('tenure_simple',
                              array['avg_hh_size_owners','area_income_owner_frac',
'commuters_per_hh_owners'],
                              array['hh_size_$hhtype',
'area_income_frac_$hhtype','hh_commuters_$hhtype'],
                              'where households>0','lai_model_input_data') as a(stfid varchar,a_o
numeric,a_r numeric,gr numeric,smoc numeric,t_o numeric,t_r numeric)";
    $from_renters = "from lai_sem_model('tenure_simple',
                              array['avg_hh_size_renters','area_income_renter_frac',
'commuters_per_hh_renters'],
                              array['hh_size_$hhtype',
'area_income_frac_$hhtype','hh_commuters_$hhtype'],
                              'where households>0','lai_model_input_data') as a(stfid varchar,a_o
numeric,a_r numeric,gr numeric,smoc numeric,t_o numeric,t_r numeric)";
    $from_vmt_owner = "from lai_model('vmt_no_transit',
                       array['commuters_per_hh','area_income_frac',
'avg_hh_size','median_number_rooms'],

array['hh_commuters_$hhtype','area_income_frac_$hhtype','hh_size_$hhtype','median_rooms_per_owner_
hu'],
                       'where households>0','lai_model_input_data') as a(stfid varchar ,vmt_per_hh
numeric)";
    $from_vmt_renter = " from lai_model('vmt_no_transit',
                       array['commuters_per_hh','area_income_frac',
'avg_hh_size','median_number_rooms'],

array['hh_commuters_$hhtype','area_income_frac_$hhtype','hh_size_$hhtype','median_rooms_per_renter
_hu'],
                       'where households>0','lai_model_input_data') as a(stfid varchar ,vmt_per_hh
numeric)";
    if($hhtype=='local'){
      $from_owners = "from lai_sem_model('tenure_simple',array['a'],array['b'],'where
households>0','lai_model_input_data')
                        as a(stfid varchar,a_o numeric,a_r numeric,gr numeric,smoc numeric,t_o
numeric,t_r numeric)";
      $from_renters = $from_owners;
      $from_vmt_owner = " from lai_model('vmt_no_transit',
                       array['commuters_per_hh',       'area_income_frac',      'avg_hh_size',
'median_number_rooms'],

array['commuters_per_hh_owners','area_income_owner_frac','avg_hh_size_owners','median_rooms_per_ow
ner_hu'],
                       'where households>0','lai_model_input_data') as a(stfid varchar ,vmt_per_hh
numeric)";
      $from_vmt_renter = " from lai_model('vmt_no_transit',
                       array['commuters_per_hh',       'area_income_frac',       'avg_hh_size',
'median_number_rooms'],

array['commuters_per_hh_renters','area_income_renter_frac','avg_hh_size_renters','median_rooms_per
_renter_hu'],
                       'where households>0','lai_model_input_data') as a(stfid varchar ,vmt_per_hh
numeric)";
    }
    echo "updating owners with hhtype $hhtype \n";
    exec_sql($db,"
  --
  --  do type $hhtype
  --
      update lai_model_$hhtype l set
          autos_per_hh_owners=round(a.a_o,2),
        model_smoc_with_mortgage=round(a.smoc),
          pct_transit_commuters_owners=round(a.t_o,1)
          $from_owners
        where l.stfid=a.stfid;

      update lai_model_$hhtype set autos_per_hh_owners = 0 where autos_per_hh_owners<0;
      update lai_model_$hhtype set model_smoc_with_mortgage = 0 where model_smoc_with_mortgage<0;
      update lai_model_$hhtype set h_cost_owners = model_smoc_with_mortgage;
      update lai_model_$hhtype set h_cost_owners = smoc_10pctile where
model_smoc_with_mortgage<smoc_10pctile;
      update lai_model_$hhtype set h_cost_owners = smoc_90pctile where
model_smoc_with_mortgage>smoc_90pctile;

      update lai_model_$hhtype set pct_transit_commuters_owners = 0 where cbsa is NULL;
      update lai_model_$hhtype set pct_transit_commuters_owners = 0 where
pct_transit_commuters_owners<0;
      update lai_model_$hhtype set pct_transit_commuters_owners = 100 where
pct_transit_commuters_owners>100;

      update lai_model_$hhtype l set vmt_per_hh_owners=round(a.vmt_per_hh)
         $from_vmt_owner
        where l.stfid=a.stfid;

      update lai_model_$hhtype set vmt_per_hh_owners=vmt_per_hh_owners*1.08;
      update lai_model_$hhtype set vmt_per_hh_owners = 0 where vmt_per_hh_owners<0;
    ");
#
    echo "updating renters with hhtype $hhtype \n";
    exec_sql($db,"
  --
  --  do type $hhtype
  --
      update lai_model_$hhtype l set
          autos_per_hh_renters=round(a.a_r,2),
          model_gross_rent=round(a.gr),
          pct_transit_commuters_renters=round(a.t_r,1)
          $from_renters
        where l.stfid=a.stfid;

      update lai_model_$hhtype set autos_per_hh_renters = 0 where autos_per_hh_renters<0;
      update lai_model_$hhtype set model_gross_rent = 0 where model_gross_rent<0;
      update lai_model_$hhtype set h_cost_renters = model_gross_rent;
      update lai_model_$hhtype set h_cost_renters = gr_10pctile where
model_smoc_with_mortgage<gr_10pctile;
      update lai_model_$hhtype set h_cost_renters = gr_90pctile where
model_smoc_with_mortgage>gr_90pctile;

      update lai_model_$hhtype set pct_transit_commuters_renters = 0 where cbsa is NULL;
      update lai_model_$hhtype set pct_transit_commuters_renters = 0 where
pct_transit_commuters_renters<0;
      update lai_model_$hhtype set pct_transit_commuters_renters = 100 where
pct_transit_commuters_renters>100;

      update lai_model_$hhtype l set vmt_per_hh_renters=round(a.vmt_per_hh)
          $from_vmt_renter
         where l.stfid=a.stfid;

      update lai_model_$hhtype set vmt_per_hh_renters=vmt_per_hh_renters*1.08;
      update lai_model_$hhtype set vmt_per_hh_renters = 0 where vmt_per_hh_renters<0;
    ");
#
# calc alpha and beta if this is local
#
    if($hhtype == 'local'){
#
```

```
      echo "about to do alpha and beta calculations \n";
#
# This is the transit factors
#
      $tr_rev = get_array_from_db($db,
        'select a.cbsa,
          a.stops_from_rev,
        a.trips_from_rev,
        a.rev_from_stops,
        a.rev_from_trips,
        a.pt_from_stops,
        a.pt_from_trips,
        t.tot_stops,
        t.tot_trips,
        a.stops_from_rev/t.tot_stops as frac_of_regions_stops,
        a.trips_from_rev/t.tot_trips as frac_of_regions_trips
         from(select cbsa,
                sum(stops) as stops_from_rev,
              sum(trips) as trips_from_rev,
                sum(rev_from_stops) as rev_from_stops,
              sum(rev_from_trips) as rev_from_trips,
                sum(pt_from_stops) as pt_from_stops,
              sum(pt_from_trips) as pt_from_trips
              from transit_revenue
              group by cbsa) a
         join (select cbsa,count(*) as tot_stops,sum(trips_per_week) as tot_trips from
stop_frequencies group by cbsa) t
           on a.cbsa=t.cbsa
         order by a.cbsa');
      $rev_tot=0.0;
      $ptr_tot=0.0;
      foreach($tr_rev as $tr){
            if($tr[frac_of_regions_stops] > 0.5 and $tr[frac_of_regions_trips] > 0.5){
             $cbsa=$tr[cbsa];
      $rev[$cbsa]=($tr[rev_from_trips]+$tr[rev_from_stops])/($tr[frac_of_regions_trips]+$tr[frac_
of_regions_stops]);

      $ptr[$cbsa]=($tr[pt_from_trips]+$tr[pt_from_stops])/($tr[frac_of_regions_trips]+$tr[frac_of
_regions_stops]);

             $rev_tot=$rev_tot+$rev[$cbsa];
             $ptr_tot=$ptr_tot+$ptr[$cbsa];
          }
      }
      echo "total revenue where we have it: $rev_tot total passanger trips: = $ptr_tot \n";
#
# sum up commuters, people and households - populations across all CBSAs
#
      $mdl_trs = get_array_from_db($db,
        "select l.cbsa,
          sum(l.pct_transit_commuters_owners*input.owner_occupied_hu +
l.pct_transit_commuters_renters*input.renter_occupied_hu) as transit_hh
        from lai_model_$hhtype l
        join lai_model_input_data input on l.stfid=input.stfid
        where l.cbsa is not null
        group by l.cbsa
        order by l.cbsa");
#
      foreach($mdl_trs as $tr){
        $cbsa=$tr['cbsa'];
        $region_th[$cbsa]=$tr['transit_hh'];
          echo "CBSA = $cbsa and use=".$region_th[$cbsa]."\n";
      }
#
# now calculate the alpha and beta and stick them in a db table
#
      $ngood=0;
      $avg_alpha_h=0;
      $avg_beta_h=0;
      foreach($rev as $cbsa => $rv){
        $tr=$ptr[$cbsa];
        if($region_th[$cbsa]){
        $a_h=$rv/$region_th[$cbsa];
        $b_h=$tr/$region_th[$cbsa];

               echo "$cbsa | $rv | $tr | $a_h \n";

               exec_sql($db,"update cbsa_alpha_beta set
           alpha_h=$a_h,
           beta_h=$b_h,
         flag='calc'
          where cbsa = '$cbsa'");
        $ngood++;
        $avg_alpha_h += $a_h;
        $avg_beta_h += $b_h;
        }
      }
#
# get averages for other regions
#
      $avg_alpha_h = $avg_alpha_h/$ngood;
      $avg_beta_h = $avg_beta_h/$ngood;
#
# update missing regions with average
#
      exec_sql($db,"update cbsa_alpha_beta set
        alpha_h=$avg_alpha_h,
      beta_h=$avg_beta_h,
        flag='avg'
       where alpha_h is NULL");
#
```

```
# first set alpha and beta to be the average value even in rural areas
#
      exec_sql($db,"update lai_model_input_data l set alpha=$avg_alpha_h,beta=$avg_beta_h");
#
# use the household based alpha and beta for block groups in CBSAs
#
      exec_sql($db,"update lai_model_input_data l set alpha=alpha_h,beta=beta_h from
cbsa_alpha_beta a where l.cbsa=a.cbsa");
      exec_sql($db,"update lai_model_local l set alpha=a.alpha,beta=a.beta from
lai_model_input_data a where l.stfid=a.stfid");
#
      echo "Just finished assigning alpha and beta to each blocks group. \n";
#
    }
  }
#
# now loop over hh types and merge renter and owner into combines
#
  echo "Running cost for $hhtype \n";
#
# calculate t costs etc...
#
  $tenure='owners';
  $i=0;
  while($i<=1){
    exec_sql($db,"update lai_model_$hhtype l set
                   auto_own_cost_$tenure = c1*1.052913,
                   vmt_cost_$tenure = c2*1.052913,
                   transit_cost_$tenure = c3*1.052913,
                   t_cost_$tenure = c4*1.052913,
                   t_$tenure = c5*1.052913,
                   transit_trips_$tenure = pct_transit_commuters_$tenure * beta,
                   h_$tenure =  (1200 * h_cost_$tenure/hh_income_$tenure),
                   ht_$tenure = (1200 * h_cost_$tenure/hh_income_$tenure) + c5*1.052913
```

```
          from (select * from lai_t_costs('autos_per_hh_$tenure',
                   'vmt_per_hh_$tenure',
                   'pct_transit_commuters_$tenure',
                   'hh_income_$tenure',
                   'gas_price',
                   'autos_per_hh_$tenure is not NULL',
                   'lai_model_$hhtype')
         as v(stfid varchar,c1 numeric,c2 numeric,c3 numeric,c4 numeric,c5 numeric)) v
           where v.stfid=l.stfid");
  $tenure='renters';
  $i++;
 }
#
# getting combined
#
 echo "Combining costs for $hhtype \n";
#
 $vars = array('autos_per_hh','h_cost',
  'pct_transit_commuters',
  'vmt_per_hh',
  'auto_own_cost',
  'vmt_cost',
  'transit_cost',
  'transit_trips',
  't_cost',
  't',
  'h',
  'ht');
 $sql = "update lai_model_$hhtype l set ";
 $comma="";
 $r="_renters";
 $o="_owners";
 foreach($vars as $v){
  $sql="$sql$comma $v = (frac_renters*$v$r + (1.0 - frac_renters)*$v$o)";
```

```
    $comma=",";
  }
  exec_sql($db,"$sql");
  $ntype++;
}
?>
```