

Technical description of mapping historical, current, and future housing densities in the US using Census block-groups

David M. Theobald

Natural Resource Ecology Laboratory

Colorado State University

Historical and current housing density

Fine-grained, spatially-detailed housing data are available by employing Census block-groups and blocks, which are subdivisions of the familiar census tract. A typical block-group contains between 250 and 550 housing units, and there are over one-quarter million block-groups in the US. Question H17 from the STF3 Census Bureau long form questionnaire codes the responses to the question: approximately what year was your house built? These responses were aggregated to each decennial year so that the number of housing units between 1940 and 1990 can be mapped. Because they are estimates of units present in April 1990, there are a number of potential reasons why these data will underestimate the number of actual historical units. These reasons include housing units that were demolished or destroyed before 1990; mis-reporting the age of a house, perhaps due to substantial renovation of earlier house; and changes in the definition of housing unit, particularly the handling of vacant units. Nationwide, the number of housing units was underestimated by 8.3% in 1980 with 1.8% of counties less, by 14.4% in 1970, and by 27.0% in 1960 (Table 2).

To correct for these potential errors, I corrected historical housing unit estimates for block-groups using county-level estimates from historical decennial census per Hammer et al. (*in press*). A correction factor for each county was computed as the ratio of the number of units in the historical census, divided by the total housing units summed from the H17 block-group estimates:

$$U_{G'} = U_G \times (U_{CD}/U_{1990}) \quad \text{Eq. 1}$$

where $U_{G'}$ is the correction factor and U_G is the un-adjusted housing units for a block-group, U_{CD} is the total housing units in the block-group's county for decade D , and U_{1990} is the total housing units in the county for the 1990 Census. Then, the adjusted number of housing units in each block-group was computed by multiplying the number of housing units times the correction

factor for the county that the block-group lies in. However, the adjusted housing units for some block-groups can exceed the number of units in the next decade. To preclude overestimation of units, I used an iterative technique that calculates the number of underestimated housing units in each county, then spreads them equally over the block-groups in a county, weighted by the number of units in each block-group. The spreading function was constrained so that the number of units in a decade did not exceed the number of units in the next decade (see Appendix 1). In some areas, particularly urban core areas that are “hollowing” out, may actually be losing housing units over a decade, this violates a basic assumption that the number of units do not decline over time for the vast majority of block-groups. Although these data estimate housing units back to 1940, here I compiled correction data only from 1960 to 1990.

To ease the description and portrayal of development patterns, I classify housing density into four general classes: urban, suburban, exurban, and rural (see file blockgroups.avl). Urban densities are typically defined as areas with greater than 1,000 people per square mile (1.6 people per acre). Assuming an average of 2.5 people per housing unit, this translates to roughly 0.7 units per acre (~1 unit per 1.6 acres). I define *urban housing density* as greater than 0.5 units per acre (>1 unit per 2.0 acres), which is slightly more relaxed than the Census definition of urban. *Suburban densities* are defined as from 0.1 to 0.5 units per acre (1 unit per 2 to 10 acres). This identifies areas that are lower-density subdivisions. *Exurban densities* range from 0.025 to 0.1 units per acre (1 unit per 10 to 40 acres). This class identifies very low-density development, including “ranchette” development in Colorado, which occurs at 1 per 35 – 45 acres. *Rural density* then is defined as housing density below 0.025 units per acre (1 per 40 acres and more). Typically this includes working farmsteads and ranches, but also includes remote vacation and second houses on the public lands interface.

Forecasting future densities

A number of approaches have been developed to forecast future growth patterns. Most of these efforts focus on *urban* growth and changes to urban or built-up cover types. The approach I developed here explicitly recognizes and represents land use changes beyond urban and built-up areas, into rural areas, which is especially important for regional planning purposes. This goal provides a foundation for understanding the consequences of alternative planning scenarios on land use patterns. Because “all models are wrong, but some are useful”, then what criteria

should be used to determine the utility for regional-scale forecasting of development patterns?

The primary goals pursued in this approach were to:

1. create a straightforward, easy-to-interpret model;
2. have model parameters accessible so they can be tweaked when simulating policy alternatives;
3. use national datasets rather than customized data;
4. be easily extendable as better understanding of the assumptions and processes is gained.

The modeling approach developed in this paper is a simplified version of the supply/demand/allocation (SDA) model. The SDA model is not driven by a particular economic theory, but is rooted in practical assumptions and limitations of development. The number of units available to be developed in an area is described by the supply component, while the demand component defines the number of units that are likely to be needed in the future, to meet the demands of a projected population. The locations where new housing units will be placed first, assuming that supply exceeds demand, are identified within the allocation component. If demand exceeds supply, then allocation is essentially irrelevant. Because the model is designed to forecast patterns for a 25 to 50 year time horizon, then most areas will be mapped so that they approach their “build-out” densities.

Broadly defined, supply is the number of units that can be developed on a piece of land. A number of coarse-scale factors help to determine whether land can be developed in the first place. Developable land is defined here as private land not occupied by water bodies such as lakes, swamps, or rivers. Additional fine-scale factors are typically considered in modeling land use change as well, such as hazardous areas (e.g., flooding, steep slopes, unstable soils, etc.) and provision of basic services (e.g., domestic wells or water and septic or sewer), though these were not used to create the block-group housing density dataset.

Initially, all developable land is assumed to be suitable for housing development. A critical factor in accurately portraying the spatial pattern of growth, however, is to consider the maximum density that an area will attain. A typical, recurrent characteristic of development is that the housing density is roughly homogeneous at a scale that corresponds roughly to subdivision patterns (~160 ac – 640 ac). A number of fine-scale factors typically determine the “build-out” density of an area, most importantly zoning. Zoning regulations typically restrict the land use and intensity (i.e. housing density) of use that can occur on a given parcel. Many

counties, particularly those in the West, do not have zoning regulations, however. For example, in Colorado, roughly one-third of counties have zoning regulations in place.

Probably the most challenging, but important, part of modeling future growth patterns is to determine the allocation of housing units. This is even more challenging when representing development as a continuum (housing density), rather than as a distinct class (e.g., urban/non-urban). The question answered here is: at what density does a given area (block-group) fill up with housing units? This determines at what point new development “spills-over” into adjacent areas. The strongest influence on these build-out densities are zoning regulations that restriction development types and densities.

In lieu of detailed information (such as zoning) on build-out densities, one alternative is to assume that future development will continue to occur in a similar pattern as it has in recent years. The approach used here is to assume that in any given decade, a block-group’s density will not exceed the average density of its neighboring block-groups. This allows urban areas to organically grow up and spread out over time. Furthermore, an advantage of this approach is that the average density is calculated locally, so it is specific to each county, indeed sub-sections of counties can have markedly different patterns. Pseudo-code that describes the logic of the spread model is provided below.

A common method to generate demand for development is to use population projections. There are three general sources of population projections. The first is detailed population projections based on economic models of job growth and trends. Because of the detail and effort taken, these are generally the best source of population projections and are often provided by state demographers. However, for regional- and national-level modeling that spans state boundaries, compiling state-level projection trends is prohibitive. A second source of population projection data is commercial databases that provide population projections based on economic models as well. The methods used to create these data are generally “black box”, that is, detailed information about the assumptions and inputs to the model are generally not available. A further limitation of both data sources is that only short-term (e.g., up to 25 or 30 year) projections are available. Because of these limitations, we developed our own projections using state-level population projections to the year 2025 from the US Census Bureau. We then projected growth for each county out to 2025 using their 1990-99 growth rates, but constrained them so that the sum of the county population did not exceed the state-level projection from the Census data.

Population estimates for 2050 were derived by a simple linear extension of growth from 2025 to 2050 to equal the same additional people as had occurred from 2000 to 2025. See the Excel spreadsheet (“county-1990-99-2025-2050.xls”) for additional information. The dBASE file (county-1990-2050gp.dbf”) was linked to a county shapefile to run the forecast model.

Spread model pseudo code

1. Calculate where new units go
 - a. $I = (G_{D-1} - G_{D-2}) / U_{D-1}$
 - b. $N = (P_D - P_{D-1}) * k * I$
 - c. $G_D = (G_{D-1} + N) \max G_{D-1}$
2. Calculate maximum density
 - a. Find adjacent BGs (.adj file)
 - b. $M_D = A / G_{D-1}$, average
3. Spread excess units
 - a. If $(G_D > M_D)$ then
 - b. $E = (G_D - M_D)$
 - c. $G_D = M_D$
 - d. Find adjacent BGs
 - e. $G_D = G_D + (E/W)$
 - f. End
4. Iteration
 - a. If $(E > 0)$ then
 - b. GOTO 3
 - c. End

where:

A = block group acres

U = total units in county

P = total population in county

p = population in block-group

C = county units

G = block-group units

M = max density in current decade

$k = G_{1990} / p_{1990}$, units to pop ratio
 W = weighted by area

Acknowledgements

Parts this analysis were supported by the William and Flora Hewlett Foundation, through the Center of the American West, University of Colorado-Boulder.

Appendix 1. Avenue script to iteratively adjust estimated housing units per block-group.

```
timeStart = Date.Now.AsSeconds

vTabBGUnits = av.FindDoc ( "bgunits.dbf" ).GetVTab
if ( vTabBGUnits.StartEditingWithRecovery.Not ) then
    msgBox.Warning ( "Couldn't edit table!","")
    return FALSE
end

vTabBGUnits.BeginTransaction

vTabCoFips = av.FindDoc ( "cofips.dbf" ).GetVTab
fldCoFips = vTabCoFips.FindField ( "coFips" )
lstDecades = { 1980, 1970, 1960 }
'lstDecades = { 1980 }

' initialize adjusted units
bmSelection = vTabBGUnits.GetSelection
bmSelection.SetAll
vTabBGUnits.SetSelection ( bmSelection )
vTabBGUnits.UpdateSelection
vTabBGUnits.Calculate ( "[Hu1980]", vTabBGUnits.FindField ( "AHu1980" ) )
vTabBGUnits.Calculate ( "[Hu1970]", vTabBGUnits.FindField ( "AHu1970" ) )
vTabBGUnits.Calculate ( "[Hu1960]", vTabBGUnits.FindField ( "AHu1960" ) )

' for each COUNTY
for each rec in vTabCoFips.GetSelection ' respects selected counties!!!
    coFips = vTabCoFips.ReturnValue ( fldCoFips, rec )

    ' select the records for each county here...
    vTabBGUnits.Query ( "([cofips] = " + coFips.Quote + " )", bmSelection,
#VTAB_SELTYPE_NEW)
    vTabBGUnits.SetSelection ( bmSelection )
    vTabBGUnits.UpdateSelection

    ' for each DECADE
    for each xDecade in lstDecades
        av.SetName ( "Adjusting: " + coFips ++ xDecade.asString )

        ' housing units from decadal census must be joined
        fldCUnits = vTabBGUnits.FindField ( "CUnits"+ xDecade.asString )
        fldAHUnits = vTabBGUnits.FindField ( "AHU"+ xDecade.asString )
        fldHUnits = vTabBGUnits.FindField ( "Hu"+ xDecade.asString )
        fldCf = vTabBGUnits.FindField ( "Cf"+ xDecade.asString )

        bmSelection = vTabBGUnits.GetSelection
        ' sum total housing units
        xAUnits = 0
        for each rec2 in vTabBGUnits.GetSelection
            xAUnits = xAUnits + vTabBGUnits.ReturnValue ( fldAhunits, rec2 )
        end

        xCUnits = vTabBGUnits.ReturnValue ( fldCUnits, bmSelection.GetNextSet ( -1
) )
        xLoop = 0
```

```

' while there is greater than 1% underestimated units
while ( ( xCUnits * 0.999 ) > xAUnits )
  xNumRecs = vTabBGUnits.GetNumSelRecords
  '
  xAdjUnits = ( ( xCUnits - xAUnits ) / xNumRecs ) max 1
  xCessUnits = ( xCUnits - xAUnits )

  ' calc adjusted units, maxing out so can't exceed next decades units
  vTabBGUnits.Calculate ( "((( "+ xCessUnits.asString + "
*([Ahu"+xDecade.asString+"]) / "+ xAUnits.asString + ") ) max 1
+[Ahu"+xDecade.asString+"]) min [Ahu"+(xDecade+10).asString + "]", fldAHUnits
)

  ' calculate new correction ratio
  '
  vTabBGUnits.Calculate ( "([CUnits"+ xDecade.asString + "] /
"+xAUnits.asString + ")", fldCf )

  ' calculate new units
  '
  vTabBGUnits.Calculate ( "([Cf"+ xDecade.asString + "] *
[Hu"+xDecade.asString+"]) min [Ahu"+(xDecade+10).asString + "]", fldAHUnits )

  ' sum total housing units
  xAUnits = 0
  for each rec2 in vTabBGUnits.GetSelection
    xAUnits = xAUnits + vTabBGUnits.ReturnValue ( fldAhunits, rec2 )
  end

  xLoop = xLoop + 1
  if ( xLoop > 15 ) then
    lf = LineFile.Make( "ahu-errors.txt".AsFileName, #FILE_PERM_APPEND )
    lf.WriteElt( "Stuck in loop"++ coFips ++ xDecade.asString + NL )
    lf.Close
    break
  end
end ' while
end ' DECADE
end ' COUNTY

vTabBGUnits.EndTransaction
vTabBGUnits.StopEditingWithRecovery ( TRUE )

av.SetName ( av.GetProject.GetName )
msgBox.info("Took: "+(Date.Now.AsSeconds - timeStart).asString,"")

```

Appendix II. Avenue code to forecast future housing densities.

```
theView = av.GetActiveDoc
thmUnits = theView.GetActiveThemes.Get(0)
fTabUnits = thmUnits.GetFTab
numFeatures = fTabUnits.GetSelection.GetSize
if ( fTabUnits.StartEditingWithRecovery.Not ) then
    msgBox.Warning ( "Couldn't edit table!","")
    return FALSE
end

timeStart = Date.Now.AsSeconds
fTabUnits.BeginTransaction

mDensityFloor = 0.025    ' minimum maximum density possible is 1 per 40 acres

' lstDecades = { 2000, 2010, 2020, 2030, 2040, 2050 }
lstDecades = { 2000, 2010, 2020, 2030, 2040, 2050 }

' -----
' create fields if they aren't here yet
lstFldsToAdd = {}

strFldName = "cu2000"
fldCU2000 = fTabUnits.FindField ( strFldName )
if ( fldCU2000 = NIL ) then
    lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "cu2010"
fldCU2010 = fTabUnits.FindField ( strFldName )
if ( fldCU2010 = NIL ) then
    lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "cu2020"
fldCU2020 = fTabUnits.FindField ( strFldName )
if ( fldCU2020 = NIL ) then
    lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "cu2030"
fldCU2030 = fTabUnits.FindField ( strFldName )
if ( fldCU2030 = NIL ) then
    lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "cu2040"
fldCU2040 = fTabUnits.FindField ( strFldName )
if ( fldCU2040 = NIL ) then
    lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "cu2050"
fldCU2050 = fTabUnits.FindField ( strFldName )
if ( fldCU2050 = NIL ) then
    lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end

strFldName = "mu2000"
fldMU2000 = fTabUnits.FindField ( strFldName )
if ( fldMU2000 = NIL ) then
    lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "mu2010"
fldMU2010 = fTabUnits.FindField ( strFldName )
if ( fldMU2010 = NIL ) then
    lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "mu2020"
```

```

fldMU2020 = fTabUnits.FindField ( strFldName )
if ( fldMU2020 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "mu2030"
fldMU2030 = fTabUnits.FindField ( strFldName )
if ( fldMU2030 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "mu2040"
fldMU2040 = fTabUnits.FindField ( strFldName )
if ( fldMU2040 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "mu2050"
fldMU2050 = fTabUnits.FindField ( strFldName )
if ( fldMU2050 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end

strFldName = "bgu2000"
fldBGU2000 = fTabUnits.FindField ( strFldName )
if ( fldBGU2000 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "bgu2010"
fldBGU2010 = fTabUnits.FindField ( strFldName )
if ( fldBGU2010 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "bgu2020"
fldBGU2020 = fTabUnits.FindField ( strFldName )
if ( fldBGU2020 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end

strFldName = "bgu2030"
fldBGU2030 = fTabUnits.FindField ( strFldName )
if ( fldBGU2030 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "bgu2040"
fldBGU2040 = fTabUnits.FindField ( strFldName )
if ( fldBGU2040 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "bgu2050"
fldBGU2050 = fTabUnits.FindField ( strFldName )
if ( fldBGU2050 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0
) )
end

strFldName = "cp2000"
fldCP2000 = fTabUnits.FindField ( strFldName )
if ( fldCP2000 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "cp2010"
fldCP2010 = fTabUnits.FindField ( strFldName )
if ( fldCP2010 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "cp2020"

```

```

fldCP2020 = fTabUnits.FindField ( strFldName )
if ( fldCP2020 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 )
)
end
strFldName = "cp2030"
fldCP2030 = fTabUnits.FindField ( strFldName )
if ( fldCP2030 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "cp2040"
fldCP2040 = fTabUnits.FindField ( strFldName )
if ( fldCP2040 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end
strFldName = "cp2050"
fldCP2050 = fTabUnits.FindField ( strFldName )
if ( fldCP2050 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_SHORT, 6, 0 ) )
end

end

strFldName = "bgu2cu"      ' percentage of block group population in county
fldBG2Co = fTabUnits.FindField ( strFldName )
if ( fldBG2Co = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_DECIMAL, 8, 6 ) )
end

strFldName = "uppl1990"   ' units per population
fldUPerPop1990 = fTabUnits.FindField ( strFldName )
if ( fldUPerPop1990 = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_DECIMAL, 8, 4 ) )
end

strFldName = "bgDens"     ' block group density
fldBgDens = fTabUnits.FindField ( strFldName )
if ( fldBGDens = NIL ) then
  lstFldsToAdd.Add ( Field.Make ( strFldName, #FIELD_DECIMAL, 8, 4 ) )
end

' add the new fields
fTabUnits.AddFields ( lstFldsToAdd )
fldBG2Co = fTabUnits.FindField ( "bgu2cu" )

' get required fields
fldBGU1980 = fTabUnits.FindField ( "ahul1980" )
fldbgp1990 = fTabUnits.FindField ( "pop1990" )
fldCP1980 = fTabUnits.FindField ( "cp1980" )
fldCP1990 = fTabUnits.FindField ( "cp1990" )
fldUPP1990 = fTabUnits.FindField ( "uppl1990" )
fldAcres = fTabUnits.FindField ( "acres" )
fldBGDens = fTabUnits.FindField ( "BgDens" )

'-----
' find the .adj file
fnAdjacent = ( thmUnits.GetSrcName.GetFileName.AsString.Substitute ( ".shp",
".adj" ) ).asFilename
vTabAdjacent = VTab.Make ( fnAdjacent, FALSE, FALSE)
if ( vTabAdjacent.HasError ) then
  msgBox.Warning ( "Couldn't find .adj file:
"+thmUnits.GetSrcName.AsString.Substitute ( ".shp", ".adj" ) ,"" )
  return NIL
end
end

```

```

' test to see if .adj needs to be updated....
if ( fTabUnits.GetNumRecords <> ( vTabAdjacent.ReturnValue (
vTabAdjacent.FindField ("ConnFeat" ), 0 ) - 1 ) ) then
  msgBox.Warning ( "Number of features .adj and records in theme did not
match, probably need to re-create .adj file!","")
' return FALSE
end

fldConnFeature = vTabAdjacent.FindField ( "Connfeat" )

' read in adj into list
lstAdj = {}
lstT1 = {}
av.ShowMsg ( "Reading adjacency list..." )
av.ShowStopButton

for each rec in fTabUnits
  if (av.SetStatus(100*( rec.Clone / numFeatures )).Not) then
    return NIL
  end

  lstT1.Empty
  for each rec2 in (vTabAdjacent.ReturnValue ( fldConnFeature, rec.Clone
))..(vTabAdjacent.ReturnValue ( fldConnFeature, rec.Clone + 1 )-1)
    lstT1.Add ( vTabAdjacent.ReturnValue ( fldConnFeature, rec2 ) )
  end
  lstAdj.Add ( lstT1.Clone )
end

'-----
' for each decade
for each xD in lstDecades
  ' initialize the projected units
  bmSelection = fTabUnits.GetSelection
  bmSelection.SetAll

  av.SetName ( "Forecasting: "+xD.asString )

  DAgol = ( xD - 10 ).asString
  DAgol2 = ( xD - 20 ).asString
  fldMU = fTabUnits.FindField ( "mu"+xD.asString )
  fldBGU = fTabUnits.FindField ( "bgu"+xD.AsString )
  fldBGUDAgol = fTabUnits.FindField ( "bgu"+DAgol.AsString )
  fldCU = fTabUnits.FindField ( "cu"+xD.AsString )

  ' calculate the proportion of new units to go to each blockgroup
  if ( fTabUnits.Calculate ( "([cp"+xD.asString+"]*[upp1990])", fldCU ).Not )
then ' new units proportional to NEW units
  msgBox.Warning ( "Error in calc: "+"([cp"+xD.asString+"]*[upp1990])", "" )
  return FALSE
  end

  if ( fTabUnits.Calculate ( "([bgu"+DAgol+"]-[bgu"+DAgol2+"]) / ([cu"+DAgol+"]-
[cu"+DAgol2+"]) max 0", fldBG2Co ).Not ) then ' new units proportional to NEW
units
  msgBox.Warning ( "Error in calc: "+"([bgu"+DAgol+"]-
[bgu"+DAgol2+"]) / ([cu"+DAgol+"]-[cu"+DAgol2+"]) max 0", "" )
  return FALSE
  end

  ' number of new units this decade
  if ( fTabUnits.Calculate ( "(((([bgu2cu]*([cp"+xD.AsString+"]-
[cp"+DAgol+"]))*[upp1990])) + [bgu"+DAgol+"]) max [bgu"+DAgol+"])", fldBGU
).Not ) then ' new units proportional to NEW units

```

```

    msgBox.Warning ( "Error in calc: "+"((( [bgu2cu]*([cp"+xD.AsString+]-
[cp"+DAgo1+]))*[upp1990])) + [bgu"+DAgo1+]]", "" )
    return FALSE
end

' density
if ( fTabUnits.Calculate ( "([bgu"+xD.asString+]" / [acres] )",fldBGDens
).Not ) then
    msgBox.Warning ( "Error in calc: "+"([bgu"+xD.asString+]" / [acres] )",
"")
    return FALSE
end

' calculate the max density for each decade by finding maximum of adjacent
(nearby) block group
av.showStopButton
av.ShowMsg ( "Calculating neighborhood max density..." )
notVisited = {}
for each rec in fTabUnits
    if (av.SetStatus(100*( rec.Clone / numFeatures )).Not) then
        return NIL
    end

' initialize
notVisited.Add ( TRUE )

' calculates average of neighbors and itself, should do it area
weighted!!!
mDensity = fTabUnits.ReturnValue ( fldBGDens, rec )
for each recAdj in lstAdj.Get(rec)
    mDensity = mDensity + ( fTabUnits.ReturnValue ( fldBGDens, recAdj ) )
end

aveDensity = mDensity / ( lstAdj.Get(rec).Count + 1 )
mUnits = aveDensity * fTabUnits.ReturnValue ( fldAcres, rec )
' Max units can not go below the density of block group last decade
if ( mUnits > fTabUnits.ReturnValue ( fldBGUDAgol, rec ) ) then
    fTabUnits.SetValue ( fldMU, rec, mUnits)
else
    fTabUnits.SetValue ( fldMU, rec, fTabUnits.ReturnValue ( fldBGUDAgol,
rec ) )
end
end ' for each rec in fTabUnits

'-----
' for each polygon, see if exceed max dens then spread to adjacent and add
them back to list to check to see if excess
fTabUnits.Query ( ("([bgu"+xD.asString+]" > [mu"+xD.asString+]])" ),
bmSelection, #VTAB_SELTYPE_NEW )
fTabUnits.UpdateSelection

rec = bmSelection.GetNextSet ( -1 )
numExceed = bmSelection.Count
iterations = 0
lstRevisit = {} ' indexes to bitmap to respread if necessary
' while ( ( numExceed > 0 ) AND ( iterations < 15 ) )

while ( ( numExceed > 1 ) AND ( iterations < 15 ) )
    lstRevisit.Empty

    av.ShowMsg ( "Spreading units
("+iterations.asString+)"..."+numExceed.asString )
    av.ShowStopButton

```

```

while ( rec <> -1 )
  if (av.SetStatus(100*( rec.Clone / numFeatures )).Not) then
    return NIL
  end

  if ( fTabUnits.ReturnValue ( fTabUnits.FindField ( "BKG_Key" ), rec ) =
"080579556003" ) then
'***** DEBUG *****
av.Run ( "script.Debug", { Script.The, "Debug at line: 2" ++
fTabUnits.ReturnValue ( fldBGU, rec ).asString, NIL } )
'*****
end

  currentBGU = fTabUnits.ReturnValue ( fldBGU, rec )
  currentMU = fTabUnits.ReturnValue ( fldMU, rec )
  excessUnits = ( currentBGU - currentMU ) max 0

  ' set BGU to MU
  if ( currentMU > currentBGU ) then
    fTabUnits.SetValue ( fldBGU, rec, currentMU )
  else
    fTabUnits.SetValue ( fldBGU, rec, currentMU )
  end
  ' weight # spill over based on adjacent polygon area (for those polygons
not visited yet)
  xTotal = 0
  for each recAdj1 in lstAdj.Get(rec)
    if ( notVisited.Get( recAdj1 ) ) then
      xTotal = xTotal + fTabUnits.ReturnValue ( fldAcres, recAdj1 )
    end
  end

  '
  lstRevisit.Add ( rec )
  ' now spread based on % of area
  for each recAdj in lstAdj.Get(rec)
    if ( notVisited.Get( recAdj ) ) then
      newUnits = (( fTabUnits.ReturnValue ( fldAcres, recAdj) / xTotal) *
excessUnits ).Round
      ' no matter what you put in, can't go less than units last decade
      fTabUnits.SetValue ( fldBGU, recAdj, ( newUnits +
fTabUnits.ReturnValue ( fldBGU, recAdj)) max fTabUnits.ReturnValue (
fldBGUDAgol, recAdj ) )
    end
  end

  if ( fTabUnits.ReturnValue ( fTabUnits.FindField ( "BKG_Key" ), rec )
= "080579556003" ) then
end

    notVisited.Set ( rec, FALSE ) 'mark as visited
    ' start back at the top of the list
    rec = bmSelection.GetNextSet ( rec )
  end

  fTabUnits.Query ( ("([bgu"+xD.asString+"] > [mu"+xD.asString+"])" ),
bmSelection, #VTAB_SELTYPE_NEW )
  bmSelection = fTabUnits.GetSelection
  ftabUnits.SetSelection ( bmSelection )
  fTabUnits.UpdateSelection
  numExceed = bmSelection.Count
  iterations = iterations + 1
  rec = bmSelection.GetNextSet ( -1 )
end

```

```
end

av.SetStatus(100)
av.ClearMsg
fTabUnits.EndTransaction
fTabUnits.StopEditingWithRecovery ( TRUE )

av.SetName ( av.GetProject.GetName )
msgBox.info("Took: "+(Date.Now.AsSeconds - timeStart).asString,"")
```