“Creating Contiguous Cartograms in ArcGIS 9”

Eric B. Wolf

Abstract:

Cartograms are powerful tools for visualizing information. Many people have seen the cartograms created after the 2000 and 2004 US Presidential Election and can attest to the power of their presentation. Unfortunately, cartograms are too time consuming and difficult to create to be used as common map making tools. Several algorithms for creating cartograms will be reviewed. Implementations as scripts for ArcGIS Desktop 9 will be presented.
Introduction

Value-by-area cartograms communicate the geographic distribution of a statistic by replacing geographic area with the value of concern. Population density is a statistic commonly represented as a cartogram, where geographic features are resized according to the number of people living within that area. Sparsely populated areas are shrunken while densely populated areas are expanded. Contiguous cartograms attempt to preserve boundaries between features while non-contiguous cartograms allow the boundaries to dissolve. In either case, a general sense of the overall geographic area must be preserved to allow the viewer to recognize the area represented.

Non-contiguous cartograms are relatively easy to create. Each feature in a map can be resized to the new area based on the desired density of the statistic. This can be done easily, although somewhat time-consuming, in ESRI ArcGIS 9 by converting features to graphics and resizing each graphic. Contiguous cartograms are almost impossible to create by hand in ArcGIS 9.

Several computer algorithms have been created to generate contiguous cartograms. The “rubber sheet distortion method” (Dougenik et al., 1985) has been very popular. Adrian Herzog’s free Java application, Mapresso, implements this algorithm (Herzog 2005) as does Charles Jackel’s Avenue script for ArcView 3 (Jackel 1997). This paper describes a new script for ArcGIS 9 written in the Visual Basic for Applications (VBA) programming environment that implements Dougenik’s algorithm for contiguous cartograms as well as methods for creating non-contiguous and circular cartograms. While this script brings new functionality to ArcGIS 9, the intent is to provide a foundation to be used to implement algorithms for creating cartograms of various types.
Cartogram Creator – User’s Perspective

The Cartogram Creator interface is designed for the novice GIS user to easily create various kinds of cartograms based on any numeric value associated with a feature layer. When the user loads the Cartogram Creator template, they are presented with the familiar ArcGIS 9 interface with the addition of a “Cartogram” button on the Tools Toolbar (to the right of the Measure and Hyperlink tools – see Figure 1). The user should add the feature layer that is to be used as a base for the cartogram and establish any joins to any tables. It is very important that the user set the Map Units and Display Units to match the units of the feature set. Cartogram Creator relies on functions built into ArcObjects for calculating areas and distances. ArcGIS assumes all calculations are in the same units and does not automatically convert. The unit of the new area-value is not important as it is used simply to determine the proportion of resulting features.

![Figure 1 - ArcMap Interface](image)

The user is encouraged to carefully select a map projection for the feature set. Cartograms alter the apparent area while attempting to preserve the shape of geographic features. Selecting a map projection that preserves area and shape is recommended.

It is best to set the symbology of the features before starting the script (see Figure 2). The symbology will be preserved by all cartogram methods but some methods convert the features to graphics. The conversion to graphics disassociates the features with their attribute table making later changes to the symbology more challenging.
The user should next select the feature layer to base the cartogram on and press the Cartogram button. If a layer is not selected or if the layer does not have any numeric fields in the attribute table, a warning message will be displayed helping the user resolve the error. If a layer with at least one numeric field in its attribute table is selected, the Cartogram Creator dialog box will be displayed (Figure 3).

The user is presented with a drop-down box used to select the Area-Value. When a value is selected, the large text box will redraw showing the statistics to be used to create the cartogram. The user can review the statistics by clicking in the text box and using the arrow keys to move around. If the Area-Value is changed, the text box will redraw based on the new value (see Figure 4).
Radio-buttons on the right of the dialog control the type of cartogram to be created. Currently Cartogram Creator supports three types of cartograms:

1. Feature Shape: each feature is converted to a graphic and simply resized according to the new area-value.
2. Circular: a graphic circle will be created centered on each feature’s centroid with the desired area.
3. Rubber Sheet: using the “rubber sheet” method for creating contiguous cartograms, each vertex of each feature will be shifted according to a “force” exerted on it by every other feature.

The user also may select “Overlap Polygons” with the checkbox. This affects how the desired areal density of the cartogram is calculated. If the Overlap Polygons box is checked, the average density for the entire feature set will be used. The result is polygons that overlap but the total area of all features is equal to the total area of the original features (see Figure 5).
If the box is unchecked, the density of the highest density feature will be used to calculate the density of the resulting features. The highest density feature maintains its size and all other features reduce in size (see Figure 6).

If the user selects the contiguous cartogram method, the dialog box will change giving the user a chance to select the base of the Output Filename and the Number of iterations (see Figure 7). The Rubber Sheet method of creating cartograms is executed iteratively on the features in order to reduce the error in the resulting features. Each iteration results in less error.
Cartogram Creator creates a new shape file for each iteration. The shape file is created with the file name as the base followed by an underscore and a number indicating the iteration number. These shape files are added to the current data frame. If you check the Delete Temp Files box, the shape files for intermediate iterations will be removed from the Data Layer and disk once all iterations are complete.

The common-control File Dialog is used to select the output file. This dialog is multi-purpose, but uses the notation “Open” for select (see Figure 8). You are not opening an existing file; you are selecting a new file to output to. If you select an existing shapefile, it will be overwritten.

The original attribute table and any joined tables will be copied to each intermediate shape file. The original symbology will also be preserved at each step. Each shape file will be added to the current Data Frame (see Figure 9).
Once all parameters are set, the user presses the “Calculate Cartogram” button. A dialog box will notify the user when the cartogram is complete and the user will be returned to the Cartogram Creator dialog. The user can then press the “Exit” to leave Cartogram Creator and assess the results.

If a non-contiguous or circular cartogram was generated, the graphics layer of the active Data Frame will contain the resulting polygons. If the user wishes to regenerate the cartogram, all of the polygons must be selected and deleted.

If a contiguous cartogram was generated, the shapefile created during each iteration will be shown as an active layer in the current Data Frame. The user may turn off all layers except for the last generated shape file to clearly see the final cartogram (See Figure 10).

The Cartogram Creator script is a work in progress. Some of these features may function differently when you try them. Please refer to the documentation distributed with the script.
Cartogram Creator – Programmer’s Perspective

Cartogram Creator is written to provide a framework for developing new methods of generating cartograms by leveraging the ArcObjects environment. Creating a cartogram with VBA in ArcGIS 9 requires solving several programming challenges. Most of these challenges are addressed in the well-documented code of the Cartogram Creator script. Some of these programming challenges and details related to adding a new cartogram algorithm are reviewed here.

Global variables are used heavily in the script to facilitate the process of creating a new cartogram method. Because of the nature of the script execution, globals are not persistent. The use of globals over locals in this case make sense: The environment external to the script is not expected to change, so interfaces like IMap do not need to be re-instantiated during execution. The script runs straight through exactly once; even if the globals are modified, it will only affect the execution of that one method and not the others.

To provide this environment for the programmer, many commonly-used COM interfaces are predefined as global variables. The variables are initialized during the form initialization process. These include:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g_Map</td>
<td>IMap</td>
<td>Current map interface</td>
</tr>
<tr>
<td>g_pMxDoc</td>
<td>IMxDocument</td>
<td>Current document</td>
</tr>
<tr>
<td>g_pActiveView</td>
<td>IActiveView</td>
<td>Current active view</td>
</tr>
<tr>
<td>g_pGraphCont</td>
<td>IGraphicsContainer</td>
<td>Current graphics container</td>
</tr>
<tr>
<td>g_pSelFLayer</td>
<td>IFeatureLayer</td>
<td>Selected feature layer interface</td>
</tr>
<tr>
<td>g_pGFL</td>
<td>IGeoFeatureLayer</td>
<td>Selected GEO feature layer interface</td>
</tr>
<tr>
<td>g_pDispFClass</td>
<td>IFeatureClass</td>
<td>Selected display feature class interface</td>
</tr>
<tr>
<td>g_pFRenderer</td>
<td>IFeatureRenderer</td>
<td>Render to the selected feature</td>
</tr>
<tr>
<td>g_pStatusBar</td>
<td>IStatusBar</td>
<td>ArcMap status bar interface</td>
</tr>
<tr>
<td>g_pProgbar</td>
<td>IStepProgressor</td>
<td>ArcMap status bar step progressor</td>
</tr>
</tbody>
</table>

Some values from the user interface are saved as globals so the programmer does not have to refer to the form controls directly:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g_strAVattr</td>
<td>String</td>
<td>Name of the new area-value variable</td>
</tr>
<tr>
<td>g_lAVattr</td>
<td>Long</td>
<td>Index in the feature values for the area-value</td>
</tr>
<tr>
<td>g_lIteration</td>
<td>Long</td>
<td>Current iteration in the process</td>
</tr>
<tr>
<td>g_lIteration_count</td>
<td>Long</td>
<td>Number of iterations requested</td>
</tr>
<tr>
<td>g_iType</td>
<td>Integer</td>
<td>Type of cartogram requested (see constants)</td>
</tr>
<tr>
<td>g_bOverlap</td>
<td>Boolean</td>
<td>True if features should overlap</td>
</tr>
<tr>
<td>g_strFilepath</td>
<td>String</td>
<td>Fully qualified path name for the output file</td>
</tr>
<tr>
<td>g_strFilebase</td>
<td>String</td>
<td>Just the root name (no path, no .shp)</td>
</tr>
<tr>
<td>g_bDelete_temp</td>
<td>Boolean</td>
<td>True if temporary shape files should be deleted</td>
</tr>
<tr>
<td>g_astrTempfiles()</td>
<td>String</td>
<td>Array of temporary files, FQNs</td>
</tr>
</tbody>
</table>
Other values that are important for generating cartograms are provided as globals and defined as the user modifies the settings in the Cartogram Creator dialog:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>g_lFeature_count</td>
<td>Long</td>
<td>Number of features in the selected layer</td>
</tr>
<tr>
<td>g_dTotal_area</td>
<td>Double</td>
<td>Sum of all source feature areas</td>
</tr>
<tr>
<td>g_dTotal_value</td>
<td>Double</td>
<td>Sum of all new area-values</td>
</tr>
<tr>
<td>g_dTotal_points</td>
<td>Double</td>
<td>Total of all vertices</td>
</tr>
<tr>
<td>g_dDensity</td>
<td>Double</td>
<td>Desired density of the resulting cartogram</td>
</tr>
<tr>
<td>g_dTotal_density</td>
<td>Double</td>
<td>Average density of all resulting features</td>
</tr>
<tr>
<td>g_dMax_density</td>
<td>Double</td>
<td>Highest density of all resulting features</td>
</tr>
</tbody>
</table>

A few constants have been defined and one new type:

The type of cartogram to be created:

```
NONCONTIGUOUS As Integer = 1
RUBBERSHEET As Integer = 2
CIRCULAR As Integer = 3
```

And a numerical constant:

```
PI As Double = 3.14159265358979
```

The new type is defined as follows:

```vbnet
Public Type myFeature
    lFID As Long            ' The FID of the feature
    sName As String         ' The name of the feature. This is filled
    in                     ' in by looking for an attribute with "name"
    in                     ' it. otherwise, it's FID #
    dValue As Double        ' The feature's new area value
    dDesired As Double      ' The desired area of the feature
    dDensity As Double      ' The original density of the feature
    dFactor As Double       ' This is the factor that the current area can be multiplied by to get the desired area
    dVertices As Double     ' This is the total number of vertices in the polygon
    lGElemPos As Long       ' This is the position of the feature when you loop through the FeatureClass. It's probably 1 to 1 with the array index, but I don't trust it
    ptCenter As IPoint      ' The centroid of the polygon
```
' This gets a little whacky, but I'm trying to duplicate the
' c_carto2.ave code as closely as possible

    dArea As Double
dRadius As Double
dMass As Double

End Type

A global array of MyFeature is initialized and filled as the user modifies the settings in the
Cartogram Creator form.

**Addressing the Feature Layer**

In order to maintain a simple interface that is consistent with much of ArcDesktop, the user
selects the feature layer to base the cartogram on and starts the Cartogram Creator script. When
the UserForm is initialized, the global variables pointing to the COM interfaces for the
IDocument, IMap, and selected layer are initialized. The standard interface to the selected
layer, retrieved from the IMxDocument.SelectedLayer method does not provide access to
joined fields in the attribute list. Instead, the DisplayFeatureClass must be retrieved from
an IGeoFeatureLayer interface to the SelectedLayer.

**Conversion to Graphics**

Some methods of creating cartograms work best as simple graphic layers. The non-contiguous
cartogram method converts all features to graphics first and then uses the ITransform2D
interface to resize each polygon. Maintaining the feature symbology is more of a challenge.
When a feature is converted to a graphic, the resulting graphic does not have attributes associated
with it. A n interface to the IFeatureRenderer is used to copy the symbology of the original
feature to the new graphic.

**Working with Feature Polygons**

Some methods of creating cartograms work best maintaining the feature layer. Contiguous
cartograms attempt to maintain the topology of the feature set. Modifying features, however, is
more challenging than adding graphics. You do not want to modify your original feature class, so
a function CopyFeatureClassToShapefile exists to create a new shapefile with the same
attributes as the source feature class. A n edit session for the new feature class can be started and
the features of the source class can be looped through and copied to the new feature class and
modified accordingly.

Each feature shape can be referenced as an IPolygon. Each IPolygon can contain multiple
IRings (separate polygons like islands). Y ou can get the ring count from the
ExteriorRingCount attribute of IPolygon and then request an array of pointers to the
rings with QueryExteriorRings. If you want to properly modify interior rings, like lakes
such as Florida’s Lake Okeechobee, you may need to loop through the Interior Rings as well.
Each ring contains an IPointCollection that delineates each vertex in the polygon. It is important to remember that while the number of elements in an IPointCollection is returned with the PointCount attribute, in a Polygon Ring, the first and last elements in IPointCollection are the same. That is, they are not two separate points with the same value, but in fact the same point. If you loop through the points starting with modifying the first one, by the time to get to the last one it will have already been modified. Because of this it is important to loop from 0 (the start of the array) to PointCount - 2.

After modifying the vertices in IPointCollection, the Update method should be called. After all rings have been modified, the IFeature.Store method should be called. Finally, the edit session should be stopped.
Conclusion

The Cartogram Creator script presented here allows the average ArcGIS to generate a variety of cartograms. Further, the script provides a solid foundation for implementing other methods of generating cartograms. The greatest programming challenges in manipulating features have been solved in clearly commented, easily reused code. The script can be modified in the Visual Basic for Applications environment built into ArcGIS 9. The availability of this script should make it possible for end-users to generate cartograms and cartographic researchers to work with new algorithms and methods for generating cartograms.

Availability of the Cartogram Creator Script

The Cartogram Creator script currently consists of two UserForms and two source modules. Printing even the source here would be prohibitive because of the length of the code. However, the entire script in an ArcGIS 9 template format is available from the ArcScripts section of ESRI.com. Simply go to http://arcscripts.esri.com and search for “cartogram”. Be sure to get the latest version of the script. The downloaded file will contain a README.txt describing the current status of the script.
Acknowledgements

I would like to acknowledge my colleague, Dr. R. Gary Litchford, at the University of Tennessee for giving me the time to work on this project. I would like to acknowledge two of my professors at Northwest Missouri State University, Dr. Yi Hwa Wu and Dr. Ming-Chih Hung, for encouraging me to explore cartograms and contiguous cartograms within the context of their classes. Finally, I would like to acknowledge my wife, Asha Wolf, DPT, for her patience with me as I tackled this project.
References


Author Information

Eric Bradway Wolf
GIS Application Support
University of Tennessee at Chattanooga
615 McCallie Avenue
Chattanooga, Tennessee 37403

(423)314-1792
eric-wolf@utc.edu