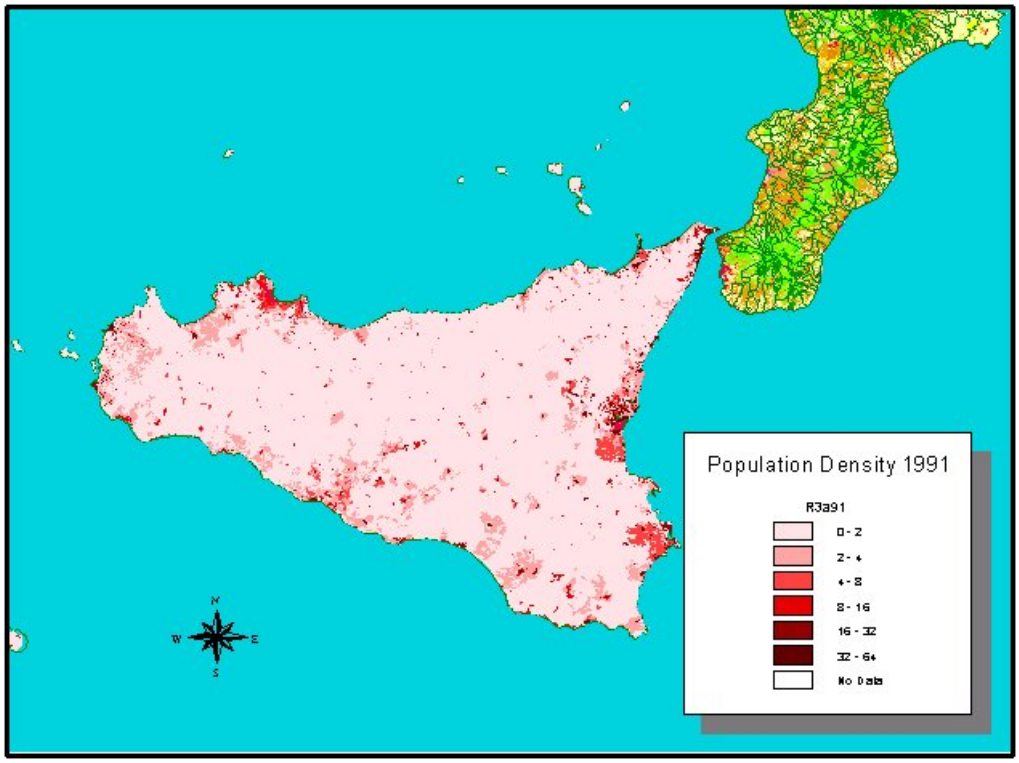
Mapping Population Density

Distribution of Population using CORINE Land Cover



Steve Peedell
Agriculture and Regional Information Systems Unit
Space Applications Institute
Joint Research Centre
Ispra

March 1999

CONTENTS

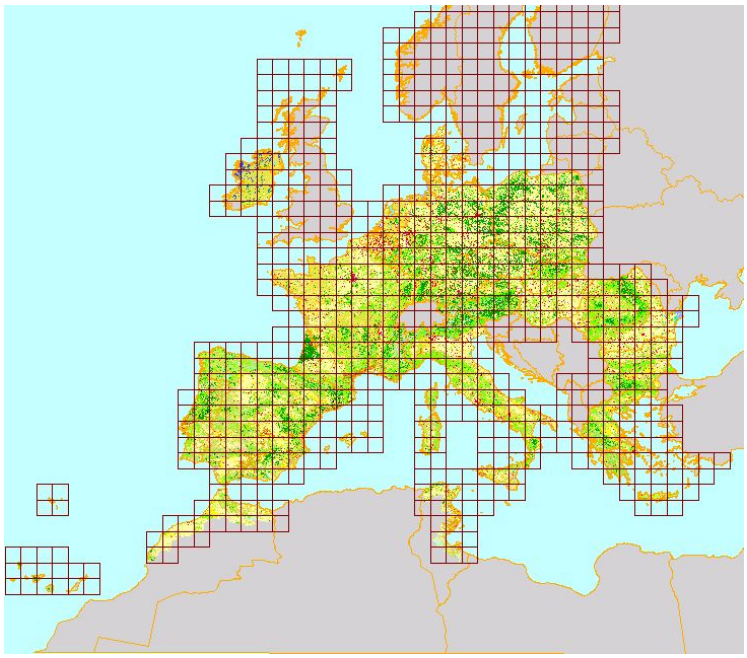
Introduction

Following discussions with the European Environment Agency in October 1998, a project was launched to investigate the feasibility of disaggregating population data, assigning different population densities according to the classes of CORINE Land Cover. The population data, held at commune level, are available for base years 1981 and 1991 from Eurostat-GISCO.

Structure and Availability of CORINE Land Cover Data

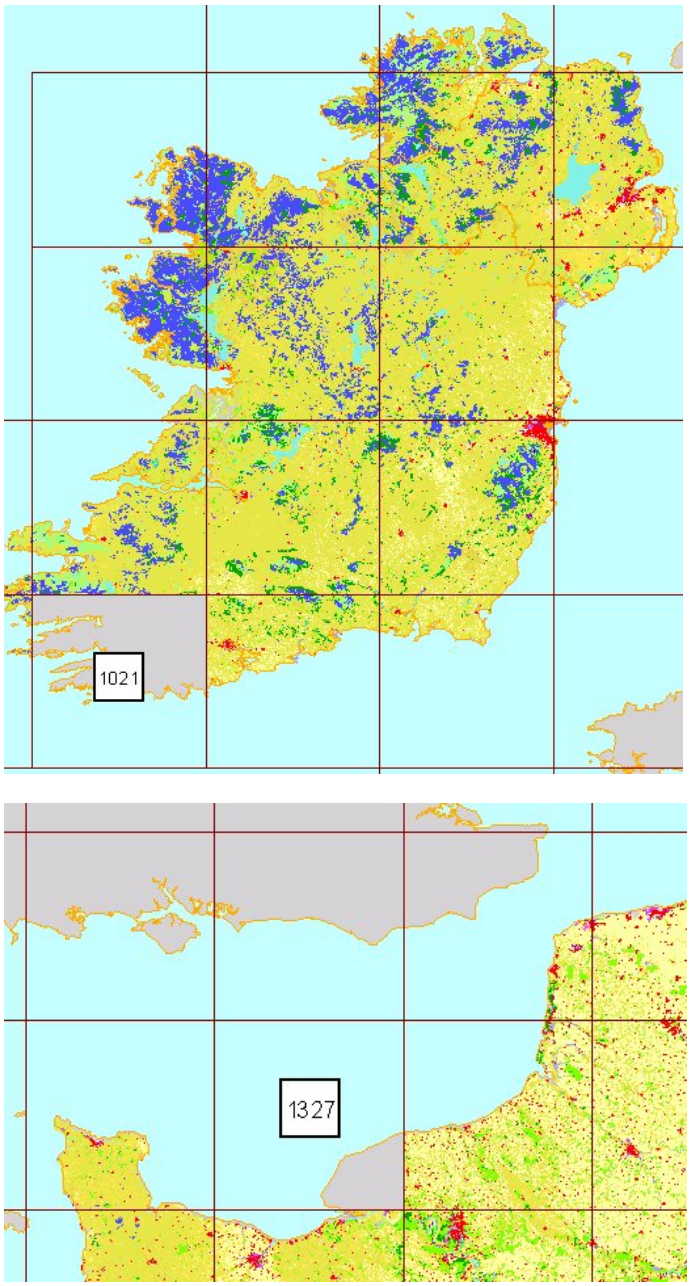
CORINE Land Cover is made available to SAI either directly through contact with the ETCLC, or via the GISCO Reference Database. The data are available in either raster or vector data models. One option currently only available with GISCO is to provide a single ArcInfo GRID format dataset, at both 100m and 250m resolution, which is extremely convenient for analytical purposes. Despite the fact that it is approximately 100Mb in this raster format, it remains an efficient alternative to using and managing a set of many discrete raster files, or even more awkward a set of discrete vector tiles. The difficulties of using vector data are primarily related to the complex and therefore time consuming nature of the geometric calculation of the relationship between the commune polygons and the land cover polygons, resulting in the creation of a third vector dataset of the result. In addition, the organisation of the vector data into tiles means that where either land cover or commune polygons span more than one tile, the integrity of a single commune or land cover object must be maintained. It is for this reason that the vector CORINE Land Cover data are organised as an ArcInfo LIBRARIAN layer, but whilst this gives a basis for the coherent display of multiple tiles, this approach still has limitations as a basis for analytical use of the data.

Missing CORINE Data



The collection and harmonisation of CORINE Land Cover data is still in progress. In some cases entire countries are missing (for example Scotland, England, Wales, Sweden, Finland, Norway), or individual tiles (as in Western Ireland). Figure 1 shows the current distribution of the raster dataset in GISCO, together with the tile structure as provided in October 1998 by ETC/LC.

Figure 1 – CORINE Land Cover Availability



During the course of this project, two tiles were found to be missing from the 100m GRID, located in Western Ireland (tile 1021) and Northern France (tile 1327). These cases are illustrated in figure 2. A check of the individual tiles provided by ETC/LC showed that in fact these tiles do exist, yet for some reason they are missing from the data provided by GISCO. For both these cases, the tiles were merged in to the overall dataset.

Figure 2 – Missing Tiles

In Greece there are areas of missing CORINE data which do not correspond to the tile boundaries (figure 3). As the reasons for this were not clear, Greece was not included in the scope of the present study.

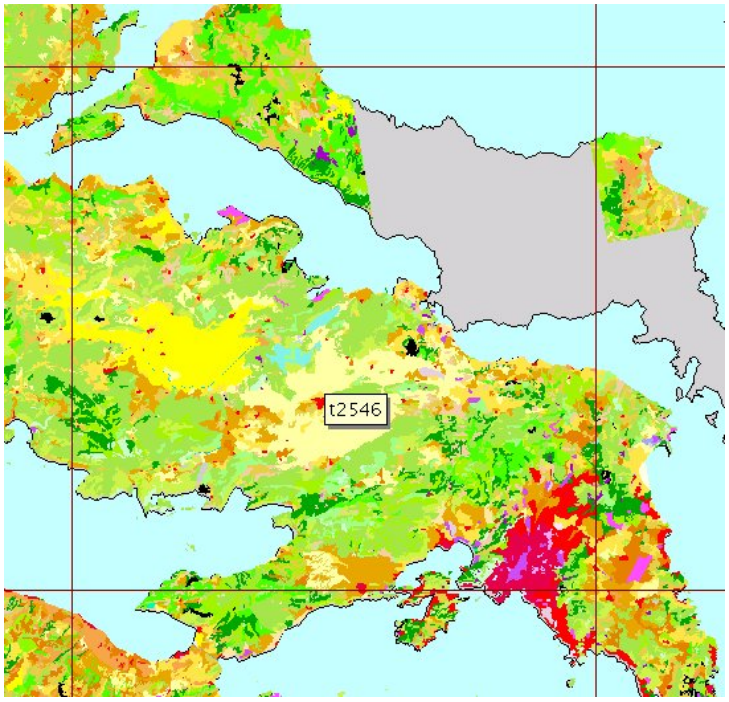
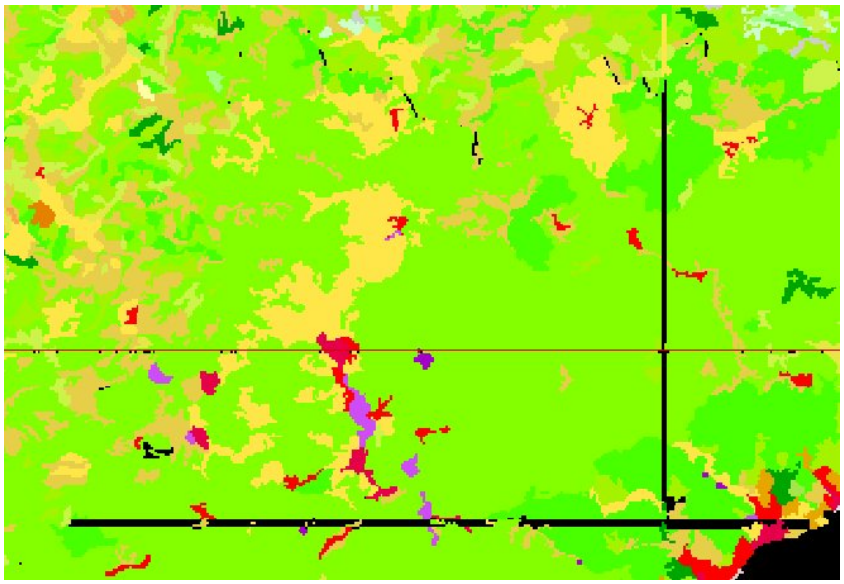


Figure 3 – Missing Greek Data

Errors in CORINE Data

In addition to the problem of missing data, errors may exist in the CORINE Land Cover dataset used as the basis of this study. These may include:

- False classification / interpretation
- Geometric errors
- Mosaicing errors



It is beyond the scope of this study to quantify the importance of these potential sources of error. An example in Piemonte, Italy shows however that errors do exist (Figure 4). This artefact may come from a mosaicing error between the imagery used for the original photo-interpretation of CORINE.

Figure 4 – Mosaicing Artefacts

Structure and Availability of Commune Data

Commune data are provided by GISCO at 1:1million scale in ArcInfo coverage format. Although more detailed data are available at 1:100,000 scale, these are too complex to use as a single coverage, hence GISCO provide the communes as a series of tiled coverages, subdivided into a regular 100km grid. ArcInfo Librarian is then used to treat these separate coverages as a single logical entity. This approach is the same as that used for the 1:100000 scale vector CORINE Land Cover data. However, data extraction from a library can be slow, and for this reason the communes at 1:1million were selected and converted from a coverage located on a network disk to a single shapefile located on the PC used for the processing. Prior to conversion to shapefiles, the attributes of population which exist in a separate table provided by GISCO were joined to the communes' attribute table.

Since the communes were to be rasterised to 100m resolution anyway, the decision to use the 1:1million scale data was not thought to have significant impact on the quality of the output results.

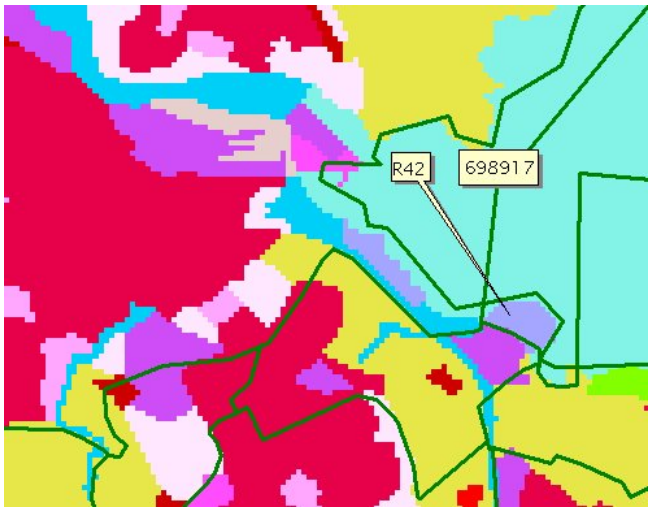
The communes provided by GISCO are available for 1995, 1991 and 1981, although not all communes are represented for all three years. Note that these years are an average baseline for all countries, and there may be fluctuation of several years either side of this norm. 1991 data were used for this study as these most closely match the dates of CORINE Land Cover data acquisition.

Errors in Commune Data

- Miscoding of communes
- Geometric errors
- Inclusion of higher level regions

The first 2 errors cannot be quantified in the context of this study – an important assumption is therefore made that the communes are correctly digitised, coded and have the correct 1991 population figure assigned to them.

In theory, the polygons in the commune dataset provided by GISCO should represent only those regions which are in the correct hierarchy of administrative regions, i.e. with a 10 character commune code. A standard commune code should be in the format "CCnnnnnnnn", where the first 2 characters are the NUTS Level 0 country code, the first 3 characters represent NUTS Level 1, the first 4 characters represent NUTS Level 2 and the first 5 characters NUTS Level 3 - followed by a serial number of n digits to give a unique reference.



However, in some countries polygons exist within this dataset which have a national or regional code, and which therefore also have a population figure for the entire country or region. These polygons often represent water bodies, but occasionally, due to slight geometric differences between the communes and the land cover, may also include pixels which could have a population allocated to them. This is illustrated in Figure 5, where a polygon is assigned a NUTS Level 1 code and the population for that entire NUTS 1 region (698917). This anomaly was not identified in the initial set of communes processed, which therefore had to be repeated.

Figure 5 - Dutch Commune Allocated NUTS 1 Code

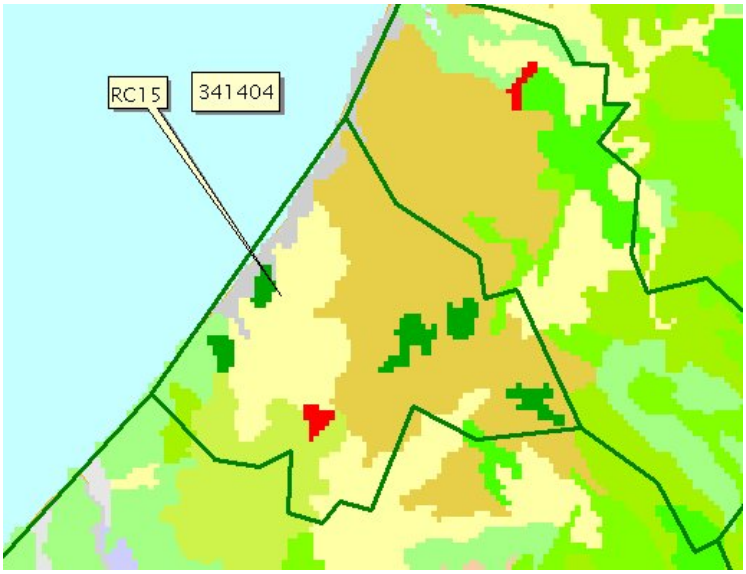


Figure 6 illustrates an example in Portugal where a seemingly valid commune has a 4 character code, and where the population value of 341404 clearly represents the number of inhabitants of a much larger region than that covered by the individual polygon.

To avoid this problem, a filter was first applied to the commune dataset to include only those polygons which had a complete 10 character commune code.

Figure 6 – Portugese Commune Allocated NUTS 2 Code

Note that the commune codes are based on Version 5 of the NUTS standard, whilst the NUTS regions themselves are based on an updated Version 6. For example, a commune "R251200135" in France would belong to a NUTS region commencing "FR" in version 6 as opposed to "R2" in Version 5.

Scale Differences

The commune data used in this study are the 1:1 million scale communes from the GISCO reference database. 1:100,000 scale polygons are also available, but these are provided in ArcInfo LIBRARIAN format and were not used for performance reasons. Scale differences can be seen in the above example figure 6, where the coastline at 1:1 million scale does not match that represented by the 100m pixels of CORINE Land Cover.

Algorithm Description

A simple algorithm has been developed to allocate population according to the Land Cover classes of CORINE.

Initially we can assume that:

$$X_m = \sum_c S_{cm} Y_{cm}$$

Where :-
X_m is the population in municipality *m*
S_{cm} is the area of Land Cover type *c* in municipality *m*

Y_{cm} is the density of population for Land Cover type *c* in municipality *m*. Inside each municipality Y_{cm} is assumed to be proportional to given parameters U_c for each Land Cover type. The values for U_c are those contained in the master table of population weightings (Appendix A), provided by EEA, and which are based on a global assessment of the study area.

We need to calculate Y_{cm}, which will be the basis for reclassifying the original CORINE Land Cover pixels to estimated population values.

$$Y_{cm} = U_c W_m$$

W_m is an adjustment factor to ensure that the total calculated population in each municipality matches the total from the administrative data. This will be necessary in any commune which has a different proportion of land cover classes than the entire study area. For example, if a commune is 100% urban class 1.1.1, one cannot distribute just 32% of the total population – the weighting must be factored accordingly.

$$X_m = \sum_c S_{cm} U_c W_m \Rightarrow W_m = \frac{X_m}{\sum_c S_{cm} U_c}$$

Hence the densities may be computed as $Y_{cm} = U_c \frac{X_m}{\sum_c S_{cm} U_c}$

Algorithm Implementation

The algorithm has been implemented in the language Avenue provided with ArcView GIS. The Spatial Analyst extension of ArcView was used to allow direct working on the GRID format. Spatial Analyst also provides a range of Avenue classes and methods to allow custom processing of GRID data.

The procedure is summarised in figure 7. The fundamental unit of analysis is the commune ($_m$), since population weightings vary on a commune by commune basis. Essentially the procedure involves calculation of population density Y_m for each class c, which can then be used to make an output GRID of Y_{cm}.

Once a given region has been processed, the individual GRID's are merged together to provide a single result GRID for that region.

Avenue scripts to undertake this procedure are given in Appendix B.

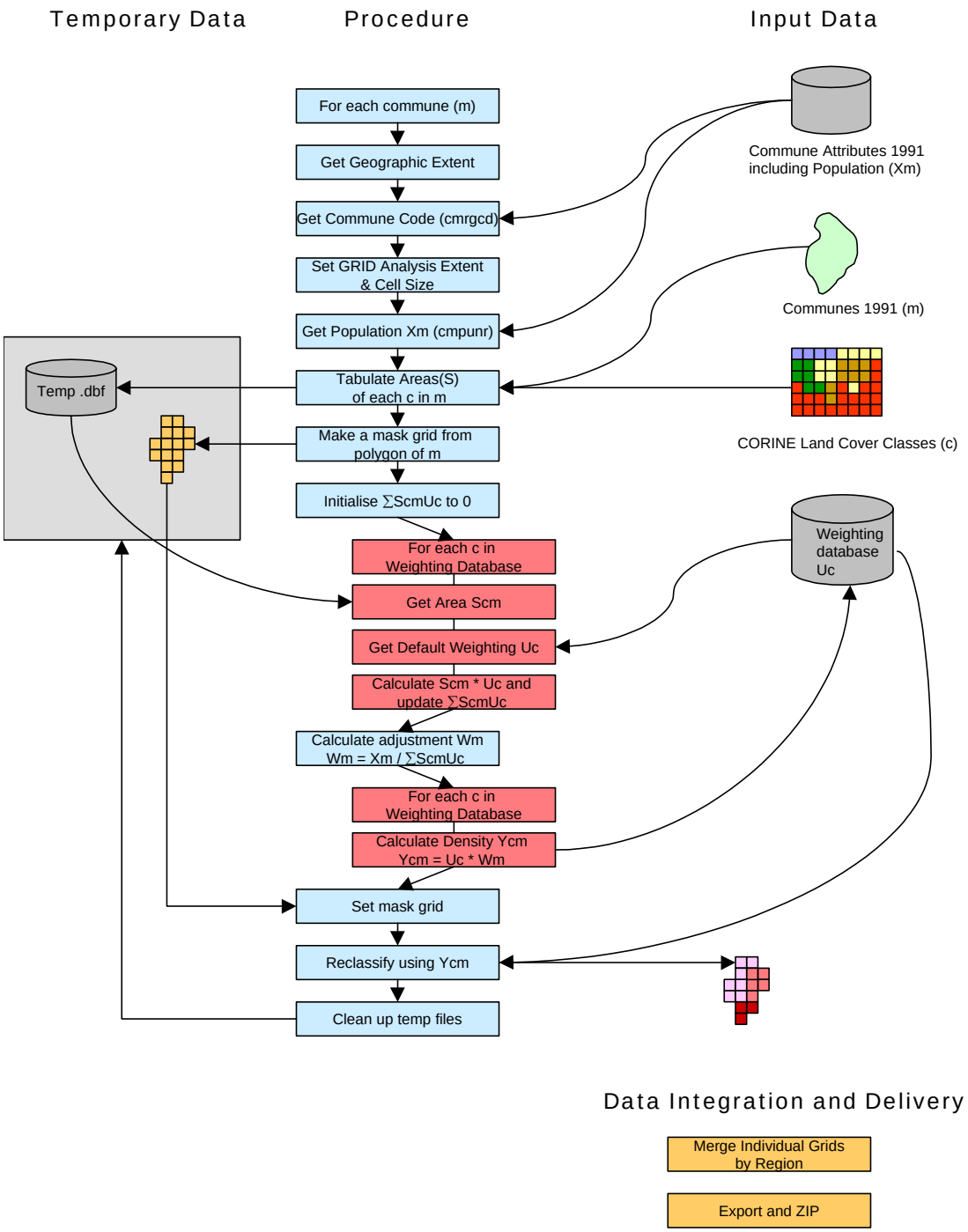


Figure 7 – Algorithm Implementation Workflow

Data Processing Considerations

Because of the sheer number of communes and land cover pixels involved in this analysis, consideration must also be given to the problems of data management. Once the algorithm was implemented, initial attempts to work at country level were tested. This should work in those countries which have less than 5000 communes. This is because for each commune, a GRID format dataset of population values is written to disk. One of the characteristics of GRID data is that all GRIDs in the same directory share a common "info" directory, where attribute and statistics files are written. In the case of the floating point GRIDs created by this analysis, 2 files are written per GRID into the info directory, which has an internal limitation of 10,000 files. Thus Italy for example, which has 10000 communes, was broken down into 9 regions, each with its own directory.

Since the output data are floating point GRIDs, no compression is applied. This means that even by breaking down countries into regions, output file sizes can be very large. An implication of this is that whilst the underlying CORINE Land Cover data can be handled on a reasonable specification PC as a single integer GRID, it would be unfeasible to mosaic the population results into a single floating point GRID. There are two alternative approaches which might be considered given this problem:

Firstly, instead of storing the results as pixels in a GRID, they could be stored in a database ready for on the fly reclassification. For example, the algorithm result is per CORINE class per commune. Therefore, in a database record the following information could be stored:

CMRGCD	***POPLC1***	***POPLC2***	***POPLCn***	***POPLC49***
RC13507005	20.79	13.28	n.nn	0
RC13507006	16.43	8.65	n.nn	0

Where POPLC1..POPLC49 columns store the calculated population for the corresponding CORINE Land Cover class in that commune. Thus, if there are 100000 communes, the database will have 100000 records. When the data need to be represented as a GRID, the underlying CORINE integer GRID can be reclassified. Since the output of a reclassify operation is always an integer GRID, it may also make sense to combine the database approach with a second method, which is to multiply out the floating point values so that they are stored as integers. In the above example, with values stored to two decimal places, multiplication of the values by 100 would result in integer results. With this method, it may be practical to create a single mosaic GRID. Obviously, any analysis and/or presentation of the data would have to include the subsequent division of the data by the same denominator.

Also a bug was discovered in Spatial Analyst which cost a significant amount of lost time. There appears to be a limitation of 32767 commands which can be submitted to the GRID engine at any one time. At this point, Avenue returns the following error: "GRD ERROR - Syntax error at or near symbol NL", and ArcView subsequently crashes.

There are 8 GRID commands performed for each iteration of this analysis, thus any sequence of processing in excess of 4000 communes may result in this error. Note that this limitation would apply if more than 2 batches of 2000 communes were analysed, as the submission list will only be cleared on exiting the ArcView project and re-entering. This made a one-off batch process impossible to implement in Avenue, and the cause of the problem was only identified towards the end of the project.

Results and Limitations

As stated previously, in the discussion of the algorithm used, this analysis is based on a simplistic approach. Obviously the driving force of the values obtained is the initial weightings table, which is itself based on very broad assumptions of how populations behave. The algorithm does not attempt to resolve population distribution differences within a given class in a commune, thus ignoring the fact that such distributions are not even. Refinement of the algorithm to incorporate the effects of other factors influencing population distribution would involve collection and harmonisation of a wide range of ancillary information, including:

- Transport networks
- Elevation
- Climate
- Economic and cultural factors
- Development restrictions

The algorithm does not attempt any morphological analysis, for example to include distance decay effects within urban areas, nor does it tackle the relationships between adjoining classes. For example, an urban area adjacent to water may have a very different distribution of population than an urban area adjacent to a suburban area.

However, given the short timescale of this project, the current algorithm does give a means of generating a regional view of population distribution with respect to Land Cover. This should allow the EEA to begin an analysis of where pressure on land resources is likely to be greatest, and where therefore a more sophisticated analysis might be appropriate.

As an illustration, Figure 8 shows the results of the analysis for Denmark.

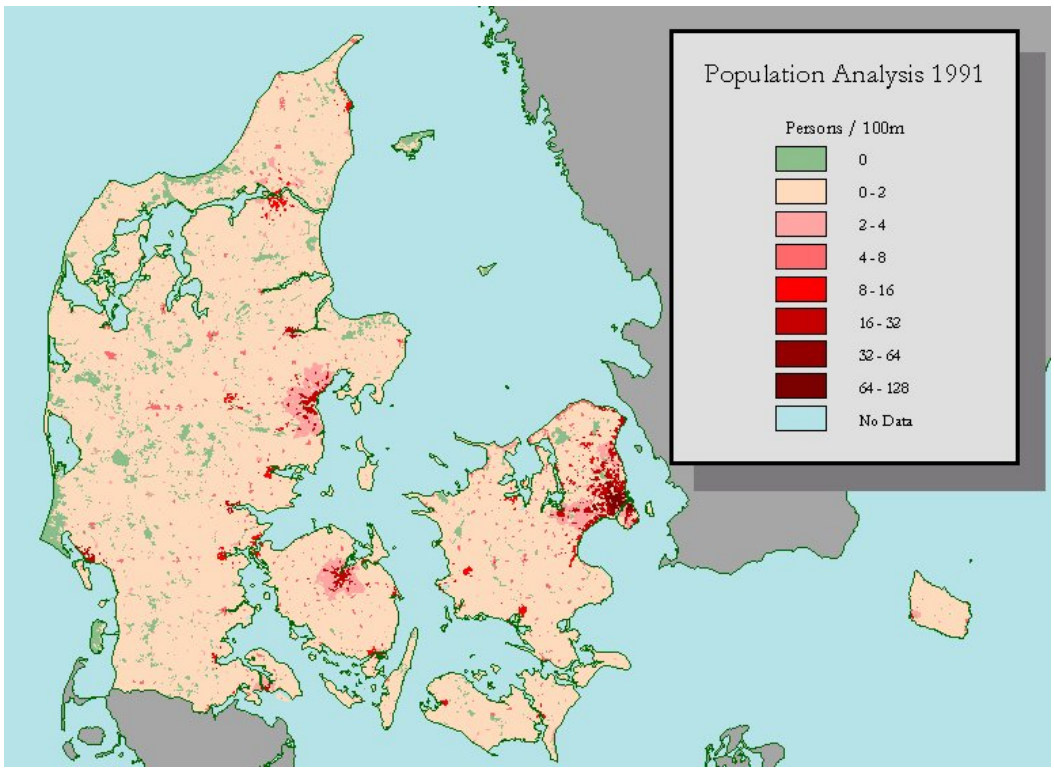


Figure 8 - 1991 Population Distribution Denmark

Appendix A – Master Population Weighting Table

Hypothetical Land Cover Category Population Weightings - **revised table CST 27/11/98**

(modifications indicated in bold)

Level1	%Popn Distribution	Level2	%Popn Distribution	Level3	%Popn Distribution
1	**63**	1.1	**57**	1.1.1	**32**
				1.1.2	**25**
		1.2	4	1.2.1	1
				1.2.2	1
				1.2.3	1
				1.2.4	1
		1.3	**0**	1.3.1	0
				1.3.2	0
				1.3.3	**0**
		1.4	**2**	1.4.1	**1**
				1.4.2	1
2	30	2.1	**7**	2.1.1	**3**
				2.1.2	**3**
				2.1.3	**1**
		2.2	**6**	2.2.1	**2**
				2.2.2	2
				2.2.3	2
		2.3	**3**	2.3.1	**3**
		2.4	**14**	2.4.1	**5**
				2.4.2	**5**
				2.4.3	**3**
				2.4.4	1
3	**7**	3.1	**3**	3.1.1	**1**
				3.1.2	**1**
				3.1.3	**1**
		3.2	4	3.2.1	1
				3.2.2	1
				3.2.3	1
				3.2.4	**1**
		3.3	**0**	3.3.1	**0**
				3.3.2	0
				3.3.3	0
				3.3.4	0
				3.3.5	0
4	0	4.1	0	4.1.1	0
				4.1.2	0
		4.2	0	4.2.1	0
				4.2.2	0
				4.2.3	0
5	0	5.1	0	5.1.1	0
				5.1.2	0
		5.2	0	5.2.1	0
				5.2.2	0
				5.2.3	0

Appendix B – ArcView Scripts

```
'-------------------------------------------------------
'- SCRIPT NAME : EEA.WeightedPopulation
'- DESCRIPTION : Calculate population densities according
'-               to distribution of CORINE Land Cover
'- AUTHOR      : Steve Peedell, ARIS Unit
'- LAST REVISED: 12th February 1999
'-------------------------------------------------------


'------------------------------
'- Called from a View document
'- which must have a Gtheme for
'- Land Cover and an FTheme for
'- commune polygons
'------------------------------
theView = av.GetActiveDoc


    '================================================
    '= Get the communes (FTheme)
    '= Field containing pop has already been joined
    '= Because of GRID limitation of 32767 commands
    '= theCMTheme is manually updated here for each
    '= country or region. In this case, the theme
    '= "cmrc.shp" is the whole of Portugal
    '================================================
        theCMTheme = theView.FindTheme("cmrc.shp")

        theCMVTab = theCMTheme.GetFTab
        theCMBitmap = theCMVTab.GetSelection
        thePopField = theCMVtab.FindField("cmpunr")
        theShapeField = theCMVtab.FindField("Shape")
        theCMIDField = theCMVtab.FindField("Cmrgcd")

    '================================================
    '= Get the landcover (GTheme)
    '================================================

        theLCTheme = theView.FindTheme("Lceugr100")
        theLCGrid = theLCtheme.GetGRID
        theVAT = theLCTheme.GetVTAB
        theValField = theVAT.FindField("Value")


    '================================================
    '= Get the default weighting VTAB
    '================================================

        theWeightTable = av.GetProject.FindDoc("weighting3.dbf")
        theWeightVTab = theWeightTable.GetVTab
        thePixelField = theWeightVTab.FindField("pixel")
        theWeightField = theWeightVTab.FindField("percpopn")
        theNewWeightField = theWeightVTab.FindField("newWeight")
        theResultPopField = theWeightVTab.FindField("weightpop")
        thePopPixelField = theWeightVTab.FindField("popperpix") '##this is Ycm * 1000
                                                                 '##to keep math integer
        theNumPixelField = theWeightVTab.FindField("numpixels")
        theAdjustField = theWeightVtab.FindField("adjust")
        theWeightVTab.SetEditable(TRUE)

    '\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
    '\ The Main Loop
    '\   Step through each commune, select it
    '\   Set extent to selected features (rounded to 100m)
```

```
    '\   Tabulate Areas to get area of each LCCD represented
    '\   Make a mask grid of the commune and set it
    '\   Calculate the proportional weightings for each class
    '\   Reclassify, output new grid
    '\ Next commune
    '\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
      for each c in theCMVTab

        theCMBitmap.ClearAll
        theCMBitmap.Set(c)
        theCommune = theCMVTab.ReturnValue(theCMIDField, c)
        '------------------------------------------------
        '- Check for multiple polygons
        '------------------------------------------------
        queryString = "([Cmrgcd] = "+theCommune.Quote+")"
        theCMVtab.Query(queryString, theCMBitmap, #VTAB_SELTYPE_NEW)
        theCMVTab.UpdateSelection
        if (FILE.Exists(("f:\steve\eea\results\rc\individual\"+theCommune).AsFileName))
then
          continue
        end

        '------------------------------------------------
        '- Tidy up the temp directory
        '------------------------------------------------
        rmpFileList = "C:\temp".AsFileName.ReadFiles("rmp*.txt")
          for each r in rmpFileList
            File.Delete(r)
          end 'for each r

        '** Get the extent of the commune poly(s), and round
        '** so that it fits exactly into the 100m grid resolution
        '** of the LandCover data

          theCMExtent = theCMTheme.GetSelectedExtent
          xmin = theCMExtent.GetLeft.Floor
          xmax = theCMExtent.GetRight.Ceiling
          ymin = theCMExtent.GetBottom.Floor
          ymax = theCMExtent.GetTop.Ceiling

          roundxmin = ((xmin / 100).Floor) * 100
          roundxmax = ((xmax / 100).Ceiling) * 100
          roundymin = ((ymin / 100).Floor) * 100
          roundymax = ((ymax / 100).Ceiling) * 100
          theExtent = Rect.MakeXY(roundXmin, roundYmin, roundXmax, roundYMax)
          '------------------------------------------------------------
          '- Set the Analysis Extent to this for performance
          '------------------------------------------------------------
          GRID.SetAnalysisExtent(#GRID_ENVTYPE_VALUE, theExtent)
          GRID.SetAnalysisCellSize(#GRID_ENVTYPE_VALUE, 100)
          '---------------------------------------
          '- Tabulate Areas (returns a VTab)
          '---------------------------------------
          tabareaVTab = theLCGrid.TabulateArea (theValField, theCMVTab,
                        Prj.MakeNull, theCMIDField, FALSE)
          numrecs = tabAreaVtab.Getnumrecords


         '------------------------------------------------
         '- Make the mask GRID
         '------------------------------------------------
          maskGrid = GRID.MakeFromFTab (theCMVTab, Prj.MakeNull,
                                        theCMIDField, NIL)

         '------------------------------------------------
         '- Get the Population (Xm)
         '------------------------------------------------
          Xm = theCMVTab.ReturnValue(thePopField, c)
```

```
  '=====================================================
  '==== Do the formula
  '=====================================================
    SigmaScmUc = 0
    for each w in theWeightVTab
      Number.SetDefFormat("")
      '-----------------------------------
      '- Get the area of each landcover Sc
      '-----------------------------------
      theClass = theWeightVTab.ReturnValue(thePixelField, w)
      '-------------------------------------------
      '- Check this class exists in the tabAreaVTab
      '-------------------------------------------
      if (tabAreaVTab.FindField("Value_"+theClass.AsString) <> NIL ) then
        Scm = tabAreaVTab.ReturnValue
              (tabAreaVTab.FindField("Value_"+theClass.AsString+"")
              , 0)/10000

        theWeightVTab.SetValue(theNumPixelField, w, Scm)
      else
        Scm = 0
        theWeightVTab.SetValue(theNumPixelField, w, Scm)
      end
      '-------------------------------------------
      '- Get the default weighting for this class Uc
      '-------------------------------------------
      Uc = theWeightVtab.ReturnValue(theWeightField, w)
      ScmUc = Scm * Uc
      SigmaScmUc = SigmaScmUc + ScmUc

    end ' for each w

    '-------------------------------------------------
    '- Calculate the adjustment factor, Wm
    '-------------------------------------------------
    number.setdefformat("dddd.ddddd")
    Wm = Xm / SigmaScmUc

    calcString = Wm.AsString
    theWeightVtab.Calculate(calcString, theAdjustField)
    theWeightVTab.Refresh
    '-------------------------------------------------
    '- Step through the loop again, to calculate the
    '- density Ycm
    '-------------------------------------------------
    for each w in theWeightVtab
      Uc = theWeightVtab.ReturnValue(theWeightField, w)
      Ycm = Uc * Wm
      theWeightVTab.SetValue(theNewWeightField, w, Ycm)
      theWeightVTab.SetValue(thePopPixelField, w, Ycm * 1000)
    end 'for each w (second loop)
 Number.SetDefFormat("")
'- set all values to zero
theWeightVTab.Calculate("0", theResultPopField)
theWeightVTab.Refresh
'-------------------------------------------------
'- Set the analysis mask to the temp commune grid
'-------------------------------------------------
Grid.SetAnalysisMask (maskGrid)
'-------------------------------------------------
'- Reclassify using Ycm as pop density
'-------------------------------------------------
popGRID = theLCGrid.Reclass (theWeightVTab, thePixelField,
          thePixelField, thePopPixelField, TRUE)
popGRIDFloat = (popGRID.Float / 1000.0)
popGRIDFloat.SaveDataset(("f:\steve\eea\results\rc\individual\"
                          +theCommune).AsFileName)
number.setdefformat("")
'-------------------------------------------------
'- Tidy up the dbf files created by TabArea
'-------------------------------------------------
```

```
          tabAreaVTab.DeActivate
          av.PurgeObjects

          dbfFileList = "C:\temp".AsFileName.ReadFiles("*.dbf")
          for each f in dbfFileList
            File.Delete(f)
          end 'for each f


       end 'for each c
    '////////////////////////////////////////////////////////
    '/ End of main loop


'-----------------------------------------------------------
'- SCRIPT NAME : EEA.Merge
'- DESCRIPTION : Perform automatic merging of individual
'-               commune level pop grids.
'-               NOTE: Max of 49 input GRIDs for merge
'-               Subsequent merging done manually
'- AUTHOR      : Steve Peedell, ARIS Unit
'- LAST REVISED: 12th February 1999
'-----------------------------------------------------------
theView = av.GetActiveDoc

'-------------------------------------------------
'- As for the WeightedPopulation, regions
'- are hard coded in the script, so references
'- in this case to "cmrc" (Portugal) would need
'- to be changed here
'-------------------------------------------------
theCMTheme = theView.FindTheme("Cmrc.shp")
theCMVtab = theCMTheme.GetFTab
theCMField = theCMVTab.FindField("Cmrgcd")

'-------------------------------------------------
'- Make sure theCMVTab has a field called "analysed"
'- this just keeps a tab on which communes have been
'- merged in case of failure part way through.
'- Adding the field was done by hand, but of course
'- could be automated
'-------------------------------------------------
theAnalysedField = theCMVTab.FindField("analysed")

'-------------------------------------------------
'- Make an empty list which will be used to store
'- the GRIDs to be merged
'-------------------------------------------------
mergegridList = {}

'---------------------------------
'- hard-coded directory for path
'---------------------------------
resultpath = "f:\steve\eea\results\rc\individual\"
theCMVTab.SetEditable(TRUE)

'---------------------------------------
'- At this level, several merged
'- grids will be created. For example
'- if there are 450 communes, 10 merges
'- will be performed, resulting in
'- grids m1..m10
'---------------------------------------
MergeCounter = 1
'---------------------------------------
'- Gridcounter will have a max value of
'- 45 to be sure merge operation doesn't
'- fail
'---------------------------------------
Gridcounter = 1
```

```
totalrecs = theCMVTab.GetNumRecords
recordcounter = 1

for each c in theCMVTab
  '-----------------------------------
  '- if we have < 45 grids in the list
  '- keep adding them
  '-----------------------------------
  if (Gridcounter < 45) then
    theCM = theCMVtab.ReturnValue(theCmfield, c)
    theGridFileName = (resultpath+theCM).AsFileName
    if (GRID.IsValidDataSetFileName(theGridFileName)) then
      theGridSrcName = GRID.MakeSrcName(theGridFileName.AsString)
      theGRID = Grid.Make(theGridSrcName)
      '-----------------------------------
      '- We need floating point GRIDs
      '- But this check is probably
      '- redundant
      '-----------------------------------
      if (theGrid.IsInteger) then
        FailFn = ("f:\steve\eea\results\rc\fail\"+theCM).AsFileName
        GRID.copyDataSet(theGRIDFileName, failFn)
      else
        mergeGridList.Add(theGRID)
      end
      GridCounter = GridCounter + 1

      '-------------------------------------------
      '- Flag that this commune has been processed
      '-------------------------------------------
      theCMVtab.SetValue(theAnalysedField, c, "Y")
    end
    '-------------------------------------------------
    '- If on this iteration we hit the last commune
    '- but the counter is still < 45, then do the
    '- merge (a neater way to do this would be to call
    '- a separate program, as the code is duplicated
    '- below
    '-------------------------------------------------
    if (recordcounter = totalrecs) then
              Grid.SetAnalysisExtent (#GRID_ENVTYPE_MAXOF, Rect.MakeNull)
              '- get the first GRID
              g=mergegridList.Get(0)
              '- merge it to the rest
              mergegrid=g.MERGE(mergegridList)
              mergeFileName = "f:\steve\eea\results\rc\merge\m"+
                              (MergeCounter.AsString)
              Av.ShowMsg(mergefileName)
              mergeGRID.SaveDataset(MergeFileName.AsFileName)
              '- empty the list ready for the next batch
              mergeGRIDList.Empty
              '- Tidy up the temp directory
              av.PurgeObjects
                    rmpFileList = "C:\temp".AsFileName.ReadFiles("rmp*.txt")
                    for each r in rmpFileList
                      File.Delete(r)
                    end 'for each r
    end
  else
    Grid.SetAnalysisExtent (#GRID_ENVTYPE_MAXOF, Rect.MakeNull)

    g=mergegridList.Get(0)
    mergegrid=g.MERGE(mergegridList)
    mergeFileName = "f:\steve\eea\results\rc\merge\m"+
                    (MergeCounter.AsString)
    Av.ShowMsg(mergefileName)
    mergeGRID.SaveDataset(MergeFileName.AsFileName)
    mergeGRIDList.Empty
    av.PurgeObjects
          rmpFileList = "C:\temp".AsFileName.ReadFiles("rmp*.txt")
          for each r in rmpFileList
```

```
            File.Delete(r)
          end 'for each r
      MergeCounter = MergeCounter + 1
      GridCounter = 1

          theCM = theCMVtab.ReturnValue(theCmfield, c)
          theGridFileName = (resultpath+theCM).AsFileName
          if (GRID.IsValidDataSetFileName(theGridFileName)) then
            theGridSrcName = GRID.MakeSrcName(theGridFileName.AsString)
            theGRID = Grid.Make(theGridSrcName)
            if (theGrid.IsInteger) then
              FailFn = ("f:\steve\eea\results\rc\fail\"+theCM).AsFileName
              GRID.copyDataSet(theGRIDFileName, failFn)
            else
              mergeGridList.Add(theGRID)
            end
            GridCounter = GridCounter + 1

            theCMVtab.SetValue(theAnalysedField, c, "Y")
      end

    end
    recordcounter = recordcounter + 1
  end 'for each c
```