Dan Patterson
Room B340A LA
Carleton University
1125 Colonel By Drive
Ottawa, Ontario
K1S 5B6 Canada

Dear Dan Patterson,

The enclosed document is a learning module entitled "Using ArcGIS to Aid Setup and Evaluation of Air Quality Models" submitted to satisfy the project component of the fall 2011 session of the GEOG 5804 - Introduction to GIS course offered by the department of Arts and Social Science at Carleton University.

The learning module enclosed is designed to help open up the ArcGIS community tools to individuals involved in air quality modelling. The crowning achievement of this document, in my opinion, is the detailed description of the development of a tool that allows modellers to import their IOAPI formatted output files into ArcGIS via a one step process. To my knowledge, no one has successfully done this before. And therefore, more than a school project, this document could qualify as a community contribution. I believe this statement is further reinforced by how another project was already effectively enabled by the work done on this project even before completion.

The style of the learning module is that of a ArcGIS tutorial. As such, a first person - sometimes casual - tone was adopted and a basic competence with ArcGIS was assumed.

Though this learning module is far from what the original project proposal would suggest, I believe it does met the objective of the project criteria, specifically, this learning module demonstrates my use and comprehension of:

- the multidimensional toolbox
- tools and techniques for point analysis
- working with and defining projections and spatial reference frames
- developing custom tools, complete with full tool descriptions
- using third-party community resources

Moreover, this project has been an exciting opportunity for me to learn Python and associated technologies (*mod_python*), become substantially more familiar with the NetCDF file format and increase my familiarity with ArcGIS.

The reason for the substantial change in purpose and scope of the project relative to the proposal was that when the proposal was written I was not familiar enough with ArcGIS to properly estimate the effort involved in what was being proposed. More than anything else, I was figuratively blindsided by my knowledge gap of how spatial data is defined in NetCDF files, how far off IOAPI files were from that, and the methods required to transform IOAPI files.

Along with that note, it is only fair to mention that this project was challenged by time lines, as such, comprehensive quality assurance techniques have not been completed on the tools developed for this learning module, and the code developed could use more complete comments. Therefore, it is possible (though unlikely!) that some use cases that the code below was developed for may fail for unexpected reasons.

For any questions concerning this learning module, please contact me at mrussel2@connect.carleton.ca.

Sincerely,

Matthew Russell (100298305)
Room 3346 ME
Carleton University
1125 Colonel By Drive
Ottawa, Ontario
K1S 5B6 Canada

Encl: Learning Module: "Using ArcGIS to Aid Setup and Evaluation of Air Quality Models"

Carleton University
Department of Geography and Environmental Studies
Faculty of Arts and Social Sciences

Learning Module
Using ArcGIS to Aid Setup and Evaluation of Air Quality Models

GEOG 5804 – Introduction to GIS

December 6th, 2011

Presented to:
Daniel Patterson
Room B340A LA
Carleton University
1125 Colonel By Drive
Ottawa, Ontario
K1S 5B6 Canada

520-2600 x2567
dpatters@connect.carleton.ca

Presented by:
Matthew Russell 100298305

Contents

1. Introduction

This learning module is aimed towards students and researchers who work with Air Quality Model (AQM). Specifically, this learning module introduces a tool that has been developed to import AQM output into ArcGIS.

Air quality modelling involves assimilating meteorological, emission, land use and solar data and running Chemical Transport Models (CTMs) to model the concentration of gas and particle species that can be harmful to human health. Though the techniques discussed in this learning module are applicable to almost any model, it is particularly aimed at those who use the US EPA's model Community Multiscale Air Quality (CMAQ). This is because the main challenge that this learning module addresses is specific to the file format output by CMAQ

The AQM community is one that is growing every year as air quality issues become more relevant to every day life. One factor that contributes to the growth of this - or any - community is the availability of tools to aid ones research. In this authors experience, a few large software tools are developed by large governmental organizations such as the US EPA which serve as the backbone to the modelling process, then small research group develop smaller innovative tools that suits their specific needs.

For example, BAMS produces IOAPI -a NetCDF wrapper layer - which is used for all inputs and outputs to CMAQ, and then an individual research group will develop automation scripts to manipulate the inputs and outputs in any given way, and send the data to some visualization package.

CMAQ - for better or for worse - has chosen the aforementioned file format, IOAPI, that cannot properly be imported into GIS programs. The result is that modellers using the IOAPI format are therefore cut off form the ArcGIS community tools - a vast library of tools to aid in anything from data manipulation to data analysis. This learning

module introduces a tool that can convert IOAPI files to a form that can be read by ArcGIS, thus bridging the gap between the two technologies and perhaps even the two communities.

This learning module reads as a tutorial where the overall goal is to use ArcGIS to aid in domain set up to model evaluation. To follow this learning module, some non-standard prerequisite software is however required (python libraries and ArcGIS upgrades), this is described in section 1.1

Section 2 walks the reader through creating the domain they wish to model. Notes on what makes good domain choices as well as instructions on defining your projection are presented.

Section 3 begins by assuming that the reader has used the domains set up in section 2 and now have IOAPI output files to import into ArcGIS. This section introduces the tool to import IOAPI data.

Section 4 briefly introduces basic techniques of model evaluation using ArcGIS. This section is by no means comprehensive, and is only intended to provide the reader with an idea of the possibilities.

1.1 Pre-Requisite Software

Many efforts were made for the toolbox and techniques used in this learning module to not rely on any non-default software, alas, some snuck in. This learning module requires some python modules, and ArcGIS to be upgraded to Service Pack 1.

The required python module is called the *MultipartPostHandler* and is used to transfer the IOAPI to and from the server (purpose described in the appendix B.) This module however is only available as a python *egg*, which requires their own library to be installed.

Therefore, it is recommended the following steps be followed. Remember throughout these instructions that ArcGIS only supports python 2.6, so if ever confronted with

a choice of package to download based on python version, choose 2.6.

1. Download *setup tools*

   This is described at http://packages.python.org/an_example_pypi_project/setuptools.html and available for download at http://pypi.python.org/pypi/setuptools

   You will require the file named *setuptools-0.6c11.win32-py2.6.exe*

2. Once downloaded, install it.

3. Ensure that your python is in your path. If you're using the python installed with ArcGIS (likely), this likely installed the setup tools (such as *easy_install*) into `C:/Python26/ArcGIS10.0/Scripts`. Therefore, this is likely the path to add to your path variable in **Control Panel** → **System** → **Advance** → **Environment Variables**

4. Download the *MultipartPostHandler* from http://pypi.python.org/pypi/MultipartPostHandler/

5. Open the command prompt, and install the *egg* file for this module with the *easy_install* command.

6. Download ArcGIS v10 Service Pack 1 from http://resources.arcgis.com/content/patches-and-service-packs?fa=viewPatch&PID=15&MetaID=1685#install-Windows

   Note, this takes considerably longer than you might expect to install.

1.2 Included Materials

This learning module is intended to be received with supplemental material.

The material is divided into folders,

prereqs Most software required by the tools in this learning module. These were discussed in section 1.1. Projection file used in this learning module can also be found here.

Example_Grid_Design All shape files used in this learning module, this includes but is not limited to:

- US State boundaries
- Random Points
- Extracted Points
- A sample IOAPI file
- The grid files set up

AQM_Toolbox The AQM Toolbox including the tool to import IOAPI files.

Webservice Python code for the webservice that actually writes the spatial information on the IOAPI file.

If an archive including these files was not provided with this document, one can be found at http://pontus.cee.carleton.ca/~matt/geog/geog5804.project.tgz

2. Defining Your Domain

The first step in running any environmental model is defining your domain. This section will bring us through the steps and rational behind those steps and how to use ArcGIS to define your grid.

Before we can run our CTM, CMAQ, we first have to run other models to prepare inputs for CMAQ. These models process meteorology, emissions and terrain topology; and these other models sometime require their own domains.

We want to set up a domain encompassing the north eastern United States with a resolution of of 36 km^2. To do this, we must:

- configure the projection we wish to use
- set up a domain to run our meteorology
- set up a domain for CMAQ

2.1 Domain Specifications

We will use a *Lambert Conic Conformal* grid projection in this exercise. This is the projection most commonly used for CMAQ, though other projections are supported.

The specific projection we would like to use is not provided by ArcGIS by default, so we will have to create it. See appendix A for instructions on how to implement the following projection. (A projection file is also available in the included materials, named *proj_lcc_cmaq.prj*)

The grid specifics are:

- False Easting: **0.0**
- False Northing: **0.0**

- Central Meridian: **-97.0**
- Standard_Parallel_1: **33**
- Standard_Parallel_2: **45**
- Latitude_Of_Origin: **40**
- Linear Unit: **Kilometre**

Or, if you prefer the information in the standard **proj** format.

```
+proj=lcc +datum=WGS84 +no_defs +lon_0=-97 +lat_1=33 +lat_2=40 +lat_0=40
    +units=km +x_0=0 +y_0=0 +title="\acs{cmaq} RPO-NA LCC"
```

Though ArcGIS does display projection information in this form, it is rather hidden. Other GIS packages however (such as QGIS) use this format more prominently.

Our CMAQ grid specification is:

- A 36 km^2 spanning the North Eastern US.
- Grid origin (lower left corner) at: $(936, -576)$ km
- 44 rows, 40 columns.

This grid was chosen because it is a standard grid used in the CMAQ test case. If you're choosing a new domain, you want to take the following into consideration:

- Is my domain large enough to properly model transport processes for the time period I intend to model?
- Do my boundaries exclude any major emission sources close to my boundaries? i.e., would expanding my domain by another cell include a major emission source that impacts my main area of interest?
- Is my domain an appropriate size and resolution to model the chemistry in my area of interest?

To accommodate this grid, our meteorology should be larger by 3 cells. Having a larger meteorological domain is is standard practice in air quality modelling, though whether it be 3 cells or some other number is the modellers choice. The reason for a larger meteorological domain is to reduce the effect of the meteorological boundaries on the CTM domain. These extra cells are later removed such that the meteorological domain matches the grid domain.

Therefore, our meteorological domain should be:

- A 36 km^2 encompassing our CMAQ domain with a padding of 3 cells
- Grid origin (lower left corner) at: $(828, -684)$ km
- 50 rows, 46 columns.

2.2 Drawing your domains in ArcGIS

Though not necessary, designing your grid in a GIS application - though slow - can be incredibly helpful to visualize your domain(s) and ensure they are exactly what you think they are. Without any visualization, the user may define their domain simply mathematically. e.g. Toronto is X km away from our grid centre in our projection, our cell size is Y, we want Z cells to be processed before Toronto, so we need K cells.

The mathematical methods work of course, but become complicated when entering domains into the different models. For example, WRF, a common meteorological model, uses grid points rather than grid cells, and calculates the offsets from the grid origin. Meanwhile, CMAQ, our CTM, uses grid cells and requires inputs from the projection centre.

First, for context, add a shapefile for the continental US. Such a shape file can be downloaded at the Global Administrative Regions website (http://www.gadm.org/) I recommend adding the *USA_adm1* file as it contains all the state boundaries.

Figure 2.1: Third-party Fishnet tool

We will use the *Fishnet* tool to draw the grids onto your projection to reduce the amount of confusion. Though ArcGIS has a native Fishnet tool, it is - in my opinion - less than intuitive to use. Therefore, this lesson recommends downloading a third-party fishnet tool from http://arcscripts.esri.com/details.asp?dbid=12807. Once you download this tool, add it to your menu bar, it should appear as in figure 2.1.

Use the fishnet tool to create the CMAQ domain specified above, i.e.

- X (km): **936**, Y (km): **-576**
- Rows: **44**, Cols: **40**
- Size of each Cell: **36**, **36**
- Specify Output Shapefile: CMAQ_36
- Coordinate System (optional): Select the projection you created earlier

You should now see a new grid layer added. At this point, I normally change it's colour to "hallow" with a blue line colour.

Next, we want to create the meteorological domain.

- X (km): **828**, Y (km): **-684**
- Rows: **50**, Cols: **46**
- Size of each Cell: **36**, **36**
- Specify Output Shapefile: WRF_36
- Coordinate System (optional): Import the projection off the previous grid

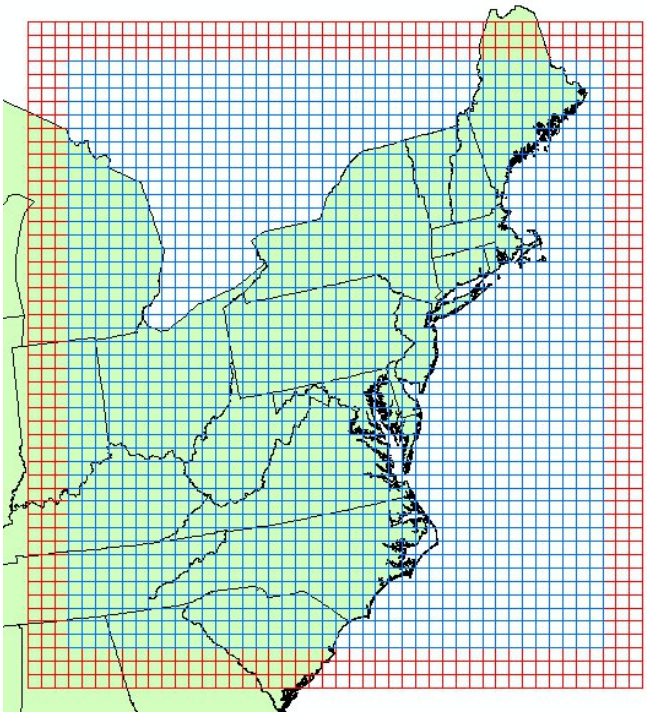

Figure 2.2: Meteorological (red) and CTM (blue) domain

The specifics of setting up the models to run this domain, and running the models is outside the scope of this tutorial.

3. Import IOAPI Data

Now that you've integrated your concentrations with CMAQ it is time to import them back into ArcGIS for evaluation. As mentioned in the introduction, IOAPI files - the output format of CMAQ - does not include standardized spatial georeferencing information, and therefore will not display properly in ArcGIS. Therefore, in order to get around this we have developed an AQM Toolbox for ArcGIS with a tool that can include IOAPI files.

In short, this tool reads the custom attributes IOAPI uses to record spatial information and builds the spatial information to the file in a more standard way[1]. The specific details of this tool are documented in Appendix B.

Add this tool to your toolbox. It has not yet been published to the Esri Resources site, but it is included in the included materials. The tool should look as shown in figure 3.1.

Remember to enable Tool Help for details on the parameters. For your input IOAPI file, choose an output file from your model that you would like to display. If you used CMAQ, you would likely want to choose the *ACONC* file. For the variable, type the

[1] Specifically, the spatial information is written using the NetCDF CF Conventions.

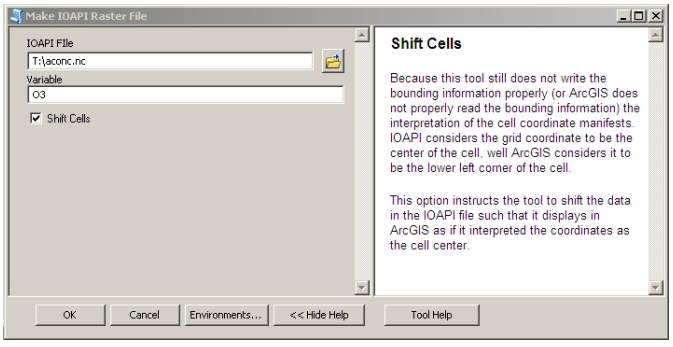

Figure 3.1: Example of *Make IOAPI Raster File* tool

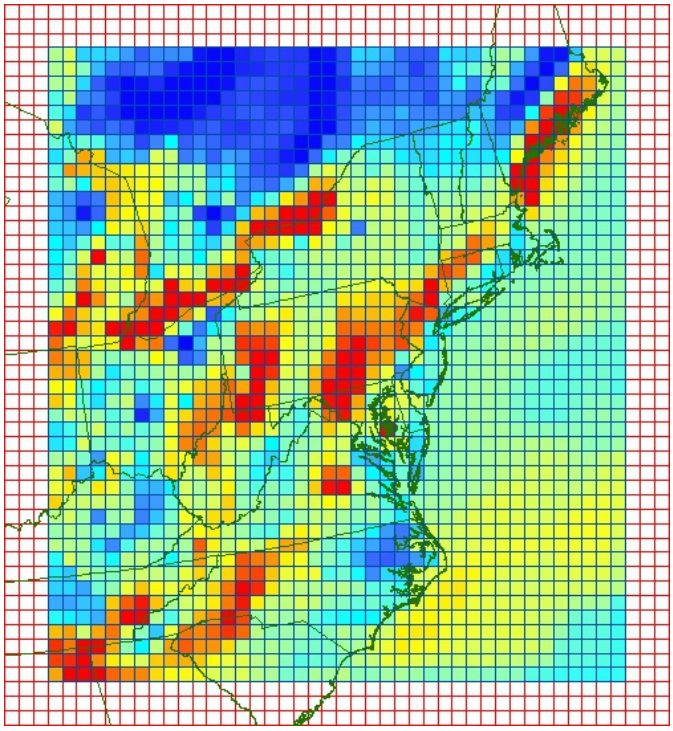

Figure 3.2: IOAPI as raster file in ArcGIS.

exact name of your variable into this text box[2]. Unlike the *Make NetCDF Raster Tool*, this tool is not able to read the variables from the file. Also, ensure that *Shift Cells* is enabled.

You might notice that this tool has substantially fewer inputs as the *Make NetCDF Raster Tool*, this is partly due to the tool being young, but also because the common use case this tool is expected to be used for does not require the other inputs.

Press *Ok*, your file is now having the spatial information corrected and will then be added into your data frame as a raster file.

If you still have the grids defined that were defined in section 2.2, your data frame should resemble figure 3.2.

Note that your time information has not been setup. If you wish to configure this,

[2]You may need to open your NetCDF file with an application such as *ncdump* to retrieve your variable names.

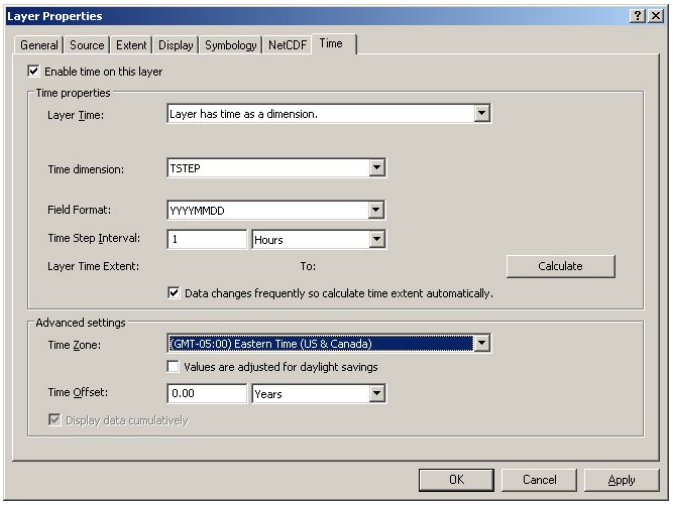

Figure 3.3: Enabling time information to a NetCDF raster file

right click on your raster layer and open the properties and choose the Time tab. This is shown in figure 3.3.

Now that time is added, you may now animate the layer. This is outside the scope of this learning module, however a tutorial for animation can be accessed here: http://www.crwr.utexas.edu/gis/gishydro06/SpaceAndTime/NetCDF/Animating%20netCDF%20Data%20in%20ArcMap.htm. This tutorial also provides a secondary source for adding NetCDF files.

4. Evaluating Results Against Observation

Now that your can import your modelled concentration into ArcGIS, you have access to all the available toolboxes. In this section, we are going to briefly show an example of how one might evaluate their modelled concentration to observations.

4.1 Generating Fake Observations

If you have observations to evaluate your model outputs, great! Otherwise, for the sake of this learning module we are going to generate fake observations. Because this section is outside the scope of this learning module, it will be briefer than other sections. If you wish, there is a random points shape file in the included material.

First, lets create a random point shape file. To do this, open the **Data Management Tools** → **Feature Class** → **Create Random Points** tool. Leave the optional *Constraining Feature Class (optional)* input empty, and instead select your *cmaq_36* grid for the *Constraining Extent (optional)*. Fill the remaining settings out as you see fit.

4.1.1 Faking Realistic Data

This will provide you with a feature of points at random locations within your area of interest. These points however have no data associated with them. For realistic data, we will extract concentration information from the raster layer at these points. Once extracted, will will use the field calculator to add a random signal to the concentrations thereby generating "realistic looking" data.

Open the **Spatial Analyst Tools** → **Extraction** → **Extract Values to Points** tool. Select your random points shape file as the *Input point features*, your raster file as the *Input Raster* and choose a place to output the shapefile.

In this new shapefile, open the *Attribute Table*, add a new field named *O3_obs* with

a precision of 10 and scale of 8. First, lets set the *ID* to something usable (perhaps for future joining). Open the field calculator on the *ID* column and enter the following formula,

```
!FID!
```

Now, open the field calculator on the new *O3_obs* column, in the *Pre-Logic Script Code* enter:

```
import random
def addnoise(val):
   val = val * (1 + random.randint(-10,10)/10)
   return val
```

Then in the *O3_obs* = text area, enter:

```
addnoise( !RASTERVALU! )
```

Delete the columns you do not need (*RASTERVALUE*, etc). And *voila*, you have realistic looking observations for all your points.

4.2 Evaluating Your Data

There are many tools available in ArcGIS to compare your modelled data to observation. These tools can be as simple as point by point comparison, or as complex as reading information from several sources at once (say population, and land type, and ...) In this section, we will generate a scatter plot to compare the performance of our modelled ozone concentration to observed concentration. Note, in this section we are assuming that time is not a factor, i.e. that the modelled concentration displaying and the observational data we have happen to be for the same time.

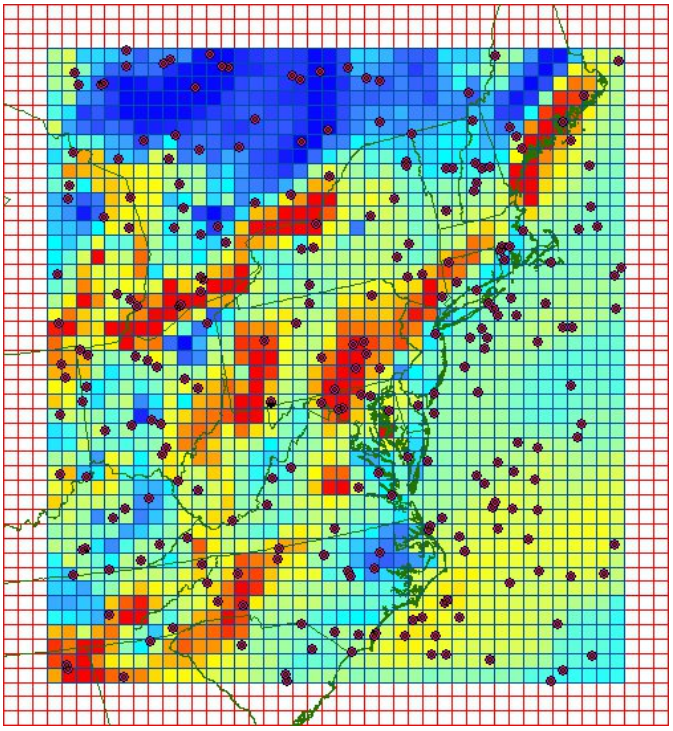

Figure 4.1: Example of observation site

Add your observations to the data frame. Your frame should appear similar to figure 4.1.

The first step is to extract your data from the raster file corresponding to your observation sites. We are going to repeat the procedure that was given in section 4.1.1.

Open the **Spatial Analyst Tools** → **Extraction** → **Extract Values to Points** tool. Select your observations point feature as the *Input point features*, your raster file as the *Input Raster* and choose a place to output the shapefile.

Open the attribute table for this tool, you will notice that it has copied the columns off your observation data. This feature saves us the effort of joining the tables.

On the *Table Options* button, select *Create Graph...*. From here, you can create a myriad of different graph styles. Figure 4.2 shows an example of a scatter plot of the observations versus the modelled data. If one were to interpret the data, they would see that there is little correlation (if any), but considering that this is faked data it really

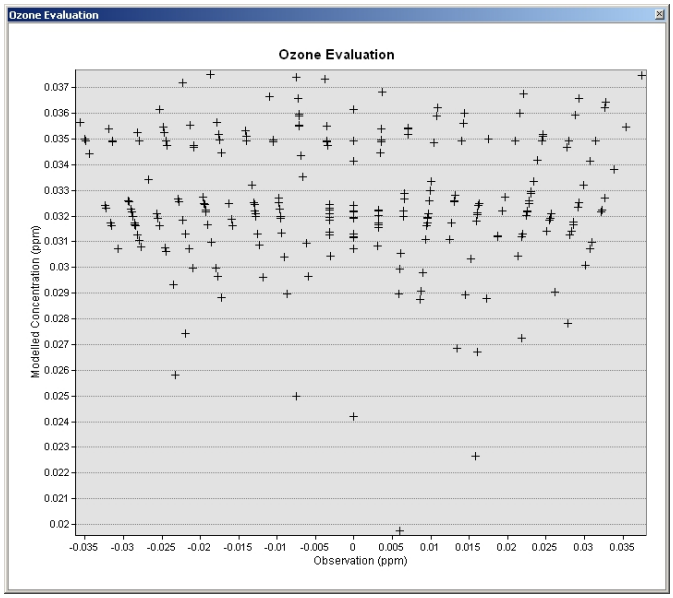

Figure 4.2: Example of a scatter plot comparing observed ozone to modelled ozone.

has no barring on the quality of our model.

To summarize, with the **Make IOAPI Raster File** tool, one can now import IOAPI output files from CMAQ and other AQMs into ArcGIS where they have a full and matured set of analysis tools.

Acronyms

AQM	Air Quality Model
BAMS	Baron Advanced Meteorological Systems
CMAQ	Community Multiscale Air Quality
CTM	Chemical Transport Model
LCC	Lambert Conic Conformal
WRF	Weather Research & Forecasting

A. Defining a Projection in ArcGIS

There are many ways to open the **Create Projection** tool. This section assumes that you have loaded multiple layers and wish to batch project them; therefore, we will use this as the starting point to creating a projection.

1. Open the **Batch Project** tool from the Arc Toolbox. **Arc Toolbox → Data Management Tools → Projections and Transformations → Feature → Batch Project**
2. Add all the layers as input
3. Choose an *Output Workspace* (preferably a new directory with the projection in its name)
4. The *Output Coordinate System* input will open the window shown in figure A.1, choose New → Projected
   At this point you should be at the window shown in figure A.2
5. Choose the values shown in figure A.2, i.e.
   - False Easting: **0.0**
   - False Northing: **0.0**
   - Central Meridian: **-97.0**
   - Standard_Parallel_1: **33.0**
   - Standard_Parallel_2: **45.0**
   - Latitude_Of_Origin: **40.0**
   - Linear Unit: **Kilometre**
   - Geographic Coordinate System: **GCS_North_American_1983**

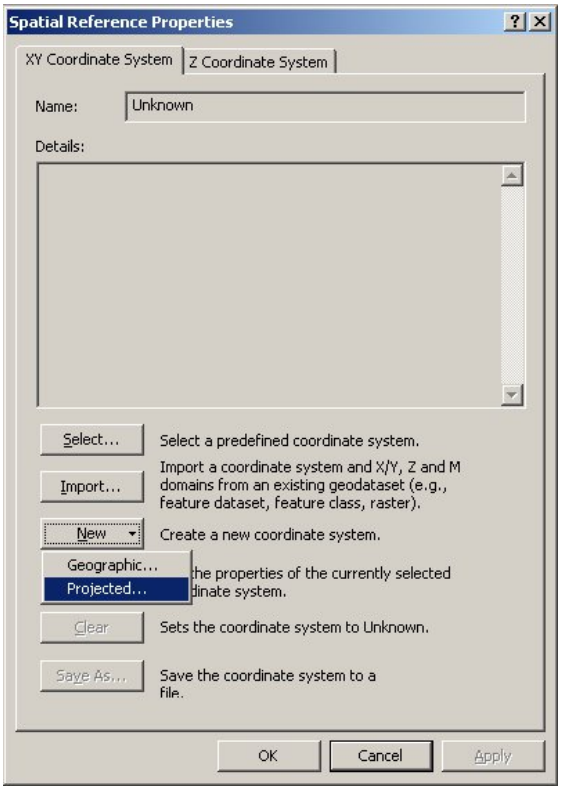

Figure A.1: Select Projection Dialog

Note: WGS84 is effectively identical to NAD83 for the purposes of this tutorial, but if WGS84 is chosen you will then be required to enter a transformation (NAD83_to_WGS_1984_1) when projecting. This is also uncertain, as other people have suggested that CMAQ uses a sphere projection[1].

6. Press **Save as...** to save the projection

Project the data, then load all the data into a new data frame.

[1] https://dawes.sph.unc.edu/groups/project_users/revisions/d7557/4/

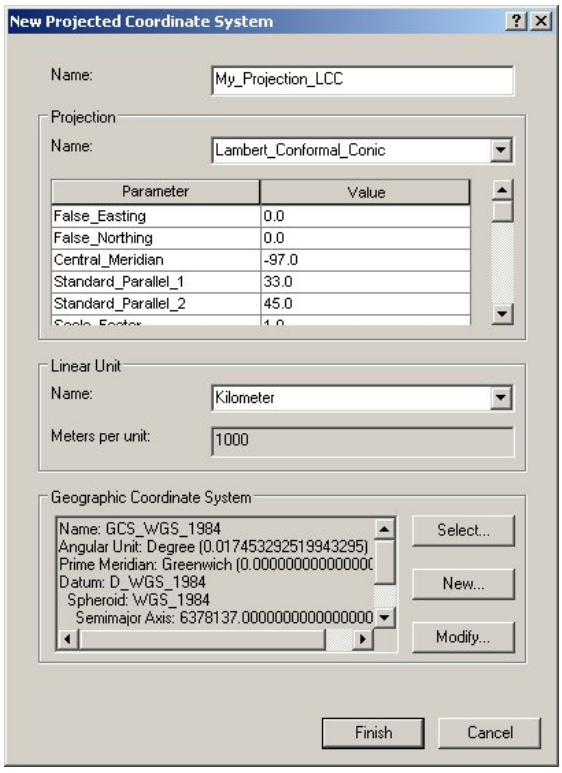

Figure A.2: Create Projection Dialog

B. Make IOAPI Raster File Tool

ArcGIS along with other GIS programs are capable of importing NetCDF files - a common format for saving multi-dimensional data. In this tutorial however, we are dealing with IOAPI files. IOAPI is a wrapper format for NetCDF files produced by the Baron Advanced Meteorological Systems (BAMS) organization and is used by CMAQ, the United States Environmental Protection Agency (US EPA)'s Chemical Transport Model (CTM).

This wrapper imposes stricter condition on how data is saved in the files and defines a custom header. Unfortunately however, the IOAPI headers are not compatible with NetCDF's CF Conventions - the convention on how data in NetCDF files is, among other things, spatially referenced. As a result, IOAPI files can still be imported using the Multidimensional Toolbox in ArcGIS, however that data can not easily be spatially referenced.

To compensate for this, I have developed a ArcGIS toolbox (**AQM → Make IOAPI Raster File**) that repairs the spatial information in the IOAPI file. This section describes in detail how this tool functions.

Accessing IOAPI files typically requires installing a long list of open source software that has to be built from source on the host computer. For the purposes of a tool that can be shared, this was deemed extremely impractical[1]. Therefore the IOAPI operations are performed remotely via a webservice.

How this works? Once the IOAPI file is uploaded and then re-downloaded with the spatial information properly encoded in the file. The local python script adds the layer to the current data frame.

[1] Building this software typically takes members in our research group months on Linux/Unix systems - the OS's that these systems are best supported. Installing these on Windows with Visual C++ is not feasible to require in a shared ArcGIS toolbox.

B.1 Local Script Code

B.1.1 Importing Parameters

First, the code imports arguments from ArcGIS. This is done with the *arcpy* module.

```
89          Input_netCDF_File = arcpy.GetParameterAsText(0).encode('ascii')
90          Input_netCDF_File = Input_netCDF_File.replace("\\","/")
91          Variable = arcpy.GetParameterAsText(1).encode('ascii')
92          shift_cells = arcpy.GetParameterAsText(2).encode('ascii')
93          if shift_cells == "true":
94                  shift_cells=True
```

The inputs for this script are:

1. Input IOAPI file

2. The data variable in the file to extract

3. Whether to shift the data in the raster file.

   This is here because the raster file at the end seems to be offset by half a grid cell. This is likely due to ArcGIS interpreting the how grid cells are represented, i.e. CMAQ considers the grid cell coordinate to be the centre of the cell, while ArcGIS considers it to be the lower left corner. To visualize this, in figure B.1, picture the blue grid as a grid made of vertices and connecting lines and the red grid as actual grid cells. The grid points in the blue grid must be shifted to correspond with the other grid.

Note that *.encode('ascii')* on many of the inputs. This is because by default ArcGIS returns text in a UTF encoding. When file names are encoded in a UTF format, access those files on the file system can then be tricky.

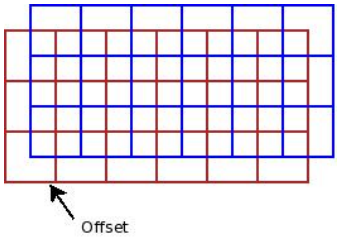

Figure B.1: Example of shifted grids due to one grid being defined by the grid centre, and the other by the lower left corner.

B.1.2 Uploading the IOAPI file

```
115 arcpy.AddMessage("Sending IOAPI file to webservice to repair spatial information")
116 arcpy.AddMessage("Uploading %s..."%Input_netCDF_File)
117 params = {'shift_cells':('1' if shift_cells else '0'), \
118         'file':open(Input_netCDF_File, 'rb')}
119
120 fileName = download(UPLOAD_URL, params, REPAIRED_IOAPI_FILE)
121 if os.access(fileName, os.R_OK):
122         arcpy.AddMessage("Downloaded file: %s"%fileName)
123 else:
124         arcpy.AddMessage("Repaired IOAPI file was not downloaded!")
125         raise Exception("Temporary file %s wasn't downloaded!"%fileName)
```

This code above calls the custom function *download*. The function is pass the current IOAPI file, all required parameters and downloads a NetCDF file with the spatial information intact.

B.1.3 Generating the Raster File

```
146 arcpy.MakeNetCDFRasterLayer_md(REPAIRED_IOAPI_FILE, Variable, X_Dimension, \
147         Y_Dimension, raster_name, "", "", "BY_VALUE")
```

Once the NetCDF file is downloaded from the server, the script calls the *MakeNetCDFRasterLayer_md* tool to generate the raster file.

B.1.4 Adding the Raster Layer to the Data Frame

```
157 arcpy.AddMessage("Loading raster file to data frame")
158
159 # get the map document
160 mxd = arcpy.mapping.MapDocument("CURRENT")
161
162 # get the data frame
163 df = arcpy.mapping.ListDataFrames(mxd,"*")[0]
164
165 # create a new layer
166 newlayer = arcpy.mapping.Layer(raster_name)
167
168 # add the layer to the map at the bottom of the TOC in data frame
169 arcpy.mapping.AddLayer(df, newlayer, "BOTTOM")
```

Once the raster file is generated, the code above is used to append it to the first data frame[2].

B.2 Webservice Code

The webservice code is written in Python and is run via *mod_python* in Apache. The host server is a Darwin OS X system, thus all the required software to work with IOAPI files can be installed with one command (the same is especially true for Ubuntu Linux systems.) This makes it a perfect candidate for a server to perform these specialized operations compared to running them on the MS Windows system running ArcGIS.

The server script operates by first parsing the HTTP request made by the client script, reading the uploaded IOAPI file, writing spatial information to it, and returning the new file. The IOAPI file is accessed using the open source NetCDF4-Python module[3].

[2] In future versions, this will be changed to add the layer to the active Data Frame
[3] Available at http://code.google.com/p/netcdf4-python/

The following code sections are shown to illustrate how the information was written. The information is written as per the NetCDF CF Conventions. Note that the variable *nc* is the original file, while the variable *nnc* is the new file that is being repaired.

B.2.1 Writing the Projection Information

Data in NetCDF files is not restricted to a single projection, every variable could have a separate projection. This is done by declaring a NetCDF variable per projection. In our file, we declare a Lambert Conic Conformal (LCC) variable.

Currently this projection is the only projection this script is capable of authoring projection information for. However, the script can be expanded to read the global attribute *GRIDTYPE* in IOAPI files to determine the projection in use. Because LCC is the most common projection used by CMAQ, and is used for all the examples in this tutorial, for now it is hard coded into the script.

```
134 try:
135         # Lets create the projection variable
136         lcc=nnc.createVariable('Lambert_Conformal', 'i4')
137         lcc.grid_mapping_name="lambert_conformal_conic"
138         lcc.standard_parallel=[nc.P_ALP, nc.P_BET]
139         lcc.longitude_of_central_meridian=nc.P_GAM
140         lcc.latitude_of_projection_origin=nc.YCENT
141         lcc.false_easting = 0.0
142         lcc.false_northing = 0.0
143 except ValueError as (errno, strerror):
144         # ...
```

In the code above, a dimensionless integer variable is being created with the NetCDF4-Python module. The values such as *nc.P_ALP* are IOAPI attributes on the original IOAPI file.

B.2.2 Adding Spatial Information to Variables

Now that a projection is defined, each variable to be projected with it must be set as such,

```
134 try:
135         for var in nc.variables:
136                 varname = str(var)
137                 nnc.variables[varname].grid_mapping = 'Lambert_Conformal'
138 except RuntimeError, err:
139         stderr += ('Ack!  RuntimeError: %s\n' % str(err))
```

This code loops through all the variables in the file assigning the projection information that we defined above to the variable attribute *grid_mapping*. Note that this is over all the variables, this is because the script *assumes* that all variables will use the same projection. This assumption is justified as IOAPI files are constrained to only containing data for one projection.

B.2.3 Defining Columns and Rows

ArcGIS requires X and Y dimensions to generate the raster file, our script writes that information to the one-dimensional float variables *COL* and *ROW* respectively.

```
# Add col/row variables
try:
        col_var=nnc.createVariable('COL', 'f4', ('COL',))
        col_var.units = "km"
        col_var.long_name = "x coordinate of projection"
        col_var.standard_name = "projection_x_coordinate"
        if verbose:
                stdout += "Added COL variable"
except ValueError as (errno, strerror):
        # ...


try:
```

```
        row_var=nnc.createVariable('ROW', 'f4', ('ROW',))
        row_var.units = "km"
        row_var.long_name = "y coordinate of projection"
        row_var.standard_name = "projection_y_coordinate"
        if verbose:
                stdout += "Added ROW variable"
except ValueError as (errno, strerror):
        # ...
```

These variables define the offset of every cell. This means that - with this implementation at least - there is no attribute that defines the sizes of each cell[4].

The data for these variables is again based on the attributes in the IOAPI file.

```
# Populate row/col vars
ncols = len(nc.dimensions['COL'])
nrows = len(nc.dimensions['ROW'])

yorig=nc.YORIG
xorig=nc.XORIG
cellsize=nc.XCELL

cols = numpy.arange(xorig, xorig + (cellsize*ncols), cellsize)
rows = numpy.arange(yorig, yorig + (cellsize*nrows), cellsize)

try:
        col_var[:] = cols;
        row_var[:] = rows;
except IndexError, err:
        # ...
```

B.3 Reviewing Spatial Information in ArcGIS

At this point, the file should include full spatial information. To confirm this, once the file is loaded into ArcGIS as a raster layer, on the layer **Properties** → **Source** tab, the **Spatial Reference** data, as well as all other data (cell size, extent) should now be filled

[4] A newer version of this script experimented with added the *cell_measures* data defined in the CF Conventions, however these attempts have to date been unsuccessful

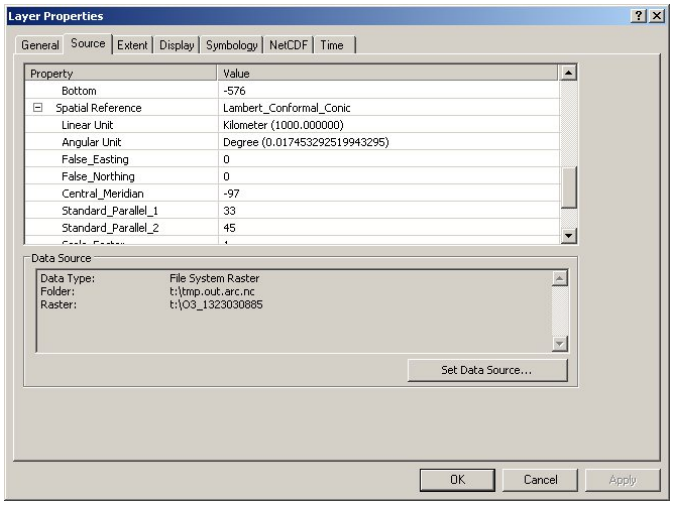

Figure B.2: Layer source information dialog

out. An example is shown in figure B.2.

B.4 Adding Tool Help

Adding Tool Help is a special challenge in ArcGIS v10. Tool Help and other metadata can be added by right clicking on the tool in Arc Toolbox and selecting *Item Description*. You will however notice that there is no place to enter parameter description (help for each input.) To do this, you must find the toolbox through Arc Catalogue through the folder connections tree, not the Toolboxes tree.

Once here, right-click and select *Item Description* again, now your description editor will have a place to enter parameter help.

Note, if you have already modified the tool description through the direct way, it will now have been erased. Also note that now that this has been done, parameter help can be edited in the *Item Description* on the tool in the Toolbox.

I suspect the roundabout approach here is due to bugs in ArcGIS v10.