

# Generating Scheduled Rasters using Python

Kevin B. Cary, GISP

Department of Geography and Geology  
Western Kentucky University  
1906 College Heights Blvd #31066  
Bowling Green, KY 42101-1066

July 2, 2015

## 2015 EdUC Abstract

Raster surfaces of precipitation variables from Ky. Mesonet weather stations are generated on a daily basis by a Python script using the ArcPy site package executed from Window's Task Scheduler. Point feature data called from an SDE geodatabase maintained by WKU's Center for GIS is interpolated for creating a continuous raster by using Spatial Analyst's Kriging tool. Raster surfaces are clipped to the commonwealth of Kentucky and uploaded as a service to ArcGIS for Server to be used in a web map Flexviewer application as an exploratory tool for drought in Kentucky examining precipitation totals and normals.

## Introduction

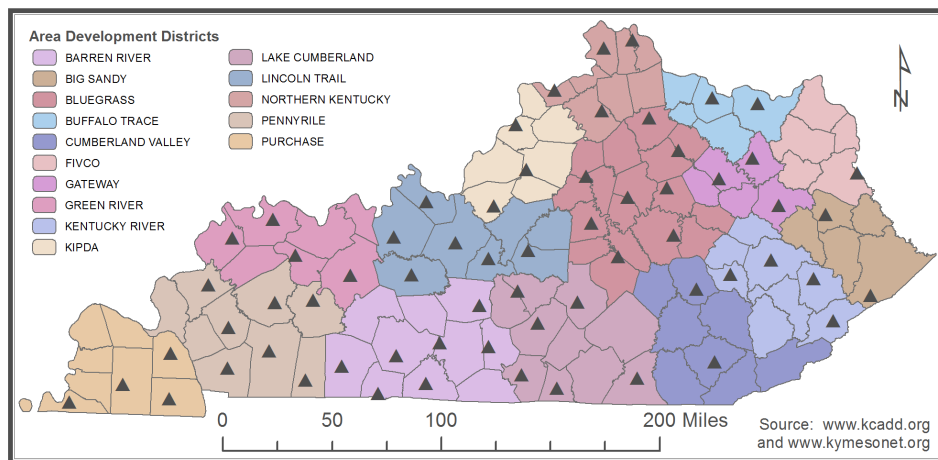


Figure 1: Map of Kentucky's Area Development Districts with Kentucky Mesonet Stations (▲).

The idea driving a need for generating precipitation surfaces are for researchers at Western Kentucky University (WKU). These precipitation surfaces are presented as raster layers in a web map for researchers investigating drought to visualize and explore quickly any areas of drought in

Kentucky from the current day in which it is viewed. This web map summarizes both precipitation amounts and deviation from normal for the past 30, 60, 90, 120, and 180 days from the day in which it is viewed. Also pushing the need are Kentucky's Area Development Districts (ADDs). An ADD can be thought of as a regional spatial unit in Kentucky with an average size of ten counties. The purpose of Ky. ADDs is to put an emphasis on progress at the state, regional and local scales (Kentucky Council, 2015). There are a total of 15 ADDs in Kentucky (Figure 1).

Data used in this project are available from the Kentucky Mesonet via Center for GIS at WKU as an SDE geodatabase. Simply put, The Kentucky Mesonet is a network of automated weather stations throughout Kentucky. The Kentucky Mesonet and Center for GIS at WKU are both housed in the Department of Geography and Geology at WKU. Since The Kentucky Mesonet's inception in 2007, it has grown to a network of over sixty weather stations across the Commonwealth (Figure 1).

The web map application used by researchers is in a FlexViewer for ArcGIS using GIS services provided by WKU's Center for GIS ArcGIS for Server (ArcGIS Viewer, 2015). Each precipitation layer for both precipitation amounts and deviation from normals are automatically updated daily at 2am in the morning on their server using Kentucky Mesonet data originally downloaded in comma delimited format and imported into an SDE database. Data is extracted from an SDE geodatabase from the Center for GIS at WKU and then interpolated to create a continuous raster surfaces of precipitation amounts and deviation from normals for the past 30, 60, 90, 120, and 180 days. This paper presents and discusses a script for automating and updating raster GIS services.

## The Script(s)

The script (or scripts) is executed as two separate Python scripts due to unexplained errors. It may be true that both scripts can be combined to run as one script, but the author's attempt was unsuccessful due to reasons unknown with running 32 bit and 64 bit versions of Python.exe. For this paper, two scripts will be discussed because that is how it is executed on the server and the reader may experience the same unexplained errors while running as one script because of the 32 and 64 bit issue with Python.exe.

The below script, one of three scripts, is actually a Windows batch file (.bat) executed daily from Windows Task Schedule at 2am CDT. This batch file will launch two Python scripts respectively: precipgen.py and then renewservice.py. Executing precipgen.py uses the default Python.exe, but renewservice.py requires the 32 bit version of Python.exe for reasons unknown. (It took many hours to figure this one out, but it was "the luck of the draw" type of troubleshooting scenarios, but it works.) The next script that is discussed is precipgen.py.

```
D:\Path1\precipgen.py  
C:\Python27\ArcGIS10.2\python.exe D:\Path1\renewservice.py
```

The script (precipgen.py) on the next page begins as any other stand-alone Python script that uses the ArcPy site package by first importing the system modules of ArcPy<sup>1</sup>. It then loads the Shutil modules for managing operating system files and folders, mostly used for cleaning up intermediate data and other files during the process of executing these scripts. This module is necessary

---

<sup>1</sup>Must have ArcGIS installed on computer executing script using the ArcPy site package.

since files (or raster data files) will be deleted for the script to work correctly as a result of running the script in the future.

```
# Name: PrecipGen.py
# Description: Generate raster surfaces of precipitation amounts
#               and deviation from normals.
# Requirements: Spatial Analyst Extension

# Import system modules
import arcpy
from arcpy import env
from arcpy.sa import *

# Import shutil modules
import shutil
from shutil import rmtree

import os

# Delete the old rasters in Rasters folder to avoid copy over errors (if any)
shutil.rmtree('Rasters')

# Create Rasters folder again for storing
os.makedirs('D:\Path1\Rasters')

env.workspace = r"Database_Connections\MesonetLive(AdminMode).sde"
inFeatures = "SDE.StationsLive"
outLocation = "D:\Path1"

# Execute (SDE) Feature Class To Shapefile
arcpy.FeatureClassToShapefile_conversion(inFeatures, outLocation)
env.workspace = "D:\Path1"
env.extent = Extent(-89.7, 36.4, -82, 39.2)

# Convert shapefile to Ky. State Plane South US Feet
arcpy.Project_management("StationsLive.shp", "SDE_StationsLiveKYSPS.shp",
"PCSKySPS.prj")
```

Continuing on with the script (precipgen.py) above, it deletes a directory labeled *Rasters* and all its contents in which are raster data from a previous execution of this script. Doing this avoids renaming future rasters a unique name as well as conserving memory since the interest is not having a historical catalog of raster layers residing on server. After deleting directory and all of its contents, this script will create a directory of the same name, *Rasters*. Next, it will connect to an SDE geodatabase (via ArcGIS for Desktop applications residing on same server) for retrieving Kentucky Mesonet weather stations with the appropriate precipitation attributes (Figure 2). The precipitation amount attributes are denoted with “p” followed by the number of days, e.g., p30. The precipitation deviation from normal attributes are denoted with “n” followed by the number of days, e.g., n30. For geoprocessing purposes, this script will save a feature class from the SDE database as a shapefile (StationsLive.shp) saving it in the correct projected coordinate system (SDE\_StationsLiveKYSPS.shp) since the SDE geodatabase is in a geographic coordinate system.

This script (precipgen.py) continues below by setting up the environment settings again but with respect to a projection coordinate systems. (In this case, it is with respect to Kentucky

NetSiteAbb	n30	n60	n90	n120	n180	p30	p60	p90	p120	p180	Div
FARM	4.16	9.56	13.9	18.23	26.07	3.54	6.75	14.7	20.66	27.53	Central
RSVL	4.24	9.63	14.2	18.41	26.48	3.97	9.21	17.5	24.3	32.14	Western
MRHD	4.31	9.28	13.1	17.13	23.86	5.56	8.35	18.5	25.99	30.85	Eastern
MRRY	4.55	9.94	14.8	19.3	27.82	4.52	9.24	15.1	22.3	29.69	Western
PCWN	4.42	9.9	14.0	18.32	26.33	5.2	7.15	16.0	22.8	29.98	Central
HTFD	3.86	9.16	13.6	17.86	25.43	3.41	6.66	13.4	20.44	26.43	Western
CMBA	4.33	9.73	13.9	18.28	26.44	4.94	7.17	15.5	21.76	28.43	Central
CRMT	4.29	9.53	13.7	18.23	25.5	5.93	8.08	14.5	21.81	26.27	Central
LXGN	4.3	9.39	13.2	17.33	24.28	5.17	6.56	15.6	22.83	27.71	Bluegrass
BLRK	3.97	9.43	13.8	18.09	25.81	2.49	9.69	18.5	26.01	31.81	Central
SCTV	4.32	9.89	14.2	18.57	27.08	6.93	9.61	16.1	21.03	28.29	Central
PRNC	4.27	9.46	14.1	18.38	26.32	2.58	6.38	13.2	20.15	26.47	Western
BMBL	4.48	9.47	13.7	17.86	25.94	4.37	6.81	13.7	19.58	27.89	Eastern
PGHL	4.3	9.64	14.3	18.55	26.57	2.44	8.47	14.2	21.17	28.03	Western
LSML	4.21	9.25	13.1	17.32	24.16	4.69	7.04	17.0	24.53	29.52	Bluegrass
ERLN	4.06	9.21	13.8	18.04	25.61	4.45	8.56	15.7	23.54	30.31	Western

Figure 2: Attribute table of SDE geodatabase.

State Plane South US Survey Feet.) “env.extent” is set to an area surrounding Kentucky so that the output raster can be clipped to the Commonwealth of Kentucky since the default setting of output rasters are bound by the input point features in the most easterly, northerly, southerly and westerly position, ie., an area inside of Kentucky; therefore, the area is extended with the envelope predetermined in the script. Next a set up parameters are determined for the Inverse Distance Weighting (IDW) interpolation method. “inPointFeatures” holds the point feature shapefile with z-values to be interpolated. “cellSize” is simply the cell size of the output raster with respect to the environment coordinate system. “power” is the degree of influence at which point features farther away will have on the interpolated value. The default value is 2 and this line could be commented out of the script. The reason why it is left in this script is for experimenting with different numbers. “searchRadius” determines how many points will be used and puts a threshold on the distance it searches for points to be used in interpolation. The distance value for the “searchRadius” is with respect to the coordinate system of the input feature class. Just before the actual interpolation, a license of ArcGIS Spatial Analyst extension must be checked out by this stand-alone script.

```
#Set environment settings again
env.workspace = "D:\Path1"
env.extent = Extent(500000, 1650000, 2740000, 2700000)

# Set local variables
inPointFeatures = "SDE_StationsLiveKYSPS.shp"
cellSize = 2500.0
power = 2
searchRadius = RadiusVariable(8, 500000)

# Check out an ArcGIS Spatial Analyst extension license
```

```
arcpy.CheckOutExtension("Spatial")
```

The following code below executes the IDW portion of the script (precipgen.py). It will first run the IDW method and save each as an IMG file. Next it will create rasters with the difference from normal. Finally, it will clip each raster using the clip management tool to the shape of Kentucky in which will be used in the service for the web map application.

```
# Execute IDW
outIDWp30 = Idw(inPointFeatures, "p30", cellSize, power, searchRadius)
outIDWp60 = Idw(inPointFeatures, "p60", cellSize, power, searchRadius)
outIDWp90 = Idw(inPointFeatures, "p90", cellSize, power, searchRadius)
outIDWp120 = Idw(inPointFeatures, "p120", cellSize, power, searchRadius)
outIDWp180 = Idw(inPointFeatures, "p180", cellSize, power, searchRadius)
outIDWpn30 = Idw(inPointFeatures, "n30", cellSize, power, searchRadius)
outIDWpn60 = Idw(inPointFeatures, "n60", cellSize, power, searchRadius)
outIDWpn90 = Idw(inPointFeatures, "n90", cellSize, power, searchRadius)
outIDWpn120 = Idw(inPointFeatures, "n120", cellSize, power, searchRadius)
outIDWpn180 = Idw(inPointFeatures, "n180", cellSize, power, searchRadius)

# Save the output
outIDWp30.save("D:\Path1\Rasters\pt30.img")
outIDWp60.save("D:\Path1\Rasters\pt60.img")
outIDWp90.save("D:\Path1\Rasters\pt90.img")
outIDWp120.save("D:\Path1\Rasters\pt120.img")
outIDWp180.save("D:\Path1\Rasters\pt180.img")
outIDWpn30.save("D:\Path1\Rasters\ptn30.img")
outIDWpn60.save("D:\Path1\Rasters\ptn60.img")
outIDWpn90.save("D:\Path1\Rasters\ptn90.img")
outIDWpn120.save("D:\Path1\Rasters\ptn120.img")
outIDWpn180.save("D:\Path1\Rasters\ptn180.img")

# Total minus Normal
outRasDiff30 = outIDWp30 - outIDWpn30
outRasDiff30.save("D:\Path1\Rasters\ptn30diff.img")
outRasDiff60 = outIDWp60 - outIDWpn60
outRasDiff60.save("D:\Path1\Rasters\ptn60diff.img")
outRasDiff90 = outIDWp90 - outIDWpn90
outRasDiff90.save("D:\Path1\Rasters\ptn90diff.img")
outRasDiff120 = outIDWp120 - outIDWpn120
outRasDiff120.save("D:\Path1\Rasters\ptn120diff.img")
outRasDiff180 = outIDWp180 - outIDWpn180
outRasDiff180.save("D:\Path1\Rasters\ptn180diff.img")

# Clip Raster
RasterToBeClippedp30 = "D:\Path1\Rasters\pt30.img"
RasterToBeClippedp60 = "D:\Path1\Rasters\pt60.img"
RasterToBeClippedp90 = "D:\Path1\Rasters\pt90.img"
RasterToBeClippedp120 = "D:\Path1\Rasters\pt120.img"
RasterToBeClippedp180 = "D:\Path1\Rasters\pt180.img"
RasterToBeClippedpn30 = "D:\Path1\Rasters\ptn30diff.img"
RasterToBeClippedpn60 = "D:\Path1\Rasters\ptn60diff.img"
RasterToBeClippedpn90 = "D:\Path1\Rasters\ptn90diff.img"
RasterToBeClippedpn120 = "D:\Path1\Rasters\ptn120diff.img"
RasterToBeClippedpn180 = "D:\Path1\Rasters\ptn180diff.img"

ClipLayer = "ky.shp"
```

```

ClippedRasterLayerp30 = "D:\Path1\Rasters\p30.img"
ClippedRasterLayerp60 = "D:\Path1\Rasters\p60.img"
ClippedRasterLayerp90 = "D:\Path1\Rasters\p90.img"
ClippedRasterLayerp120 = "D:\Path1\Rasters\p120.img"
ClippedRasterLayerp180 = "D:\Path1\Rasters\p180.img"
ClippedRasterLayerpn30 = "D:\Path1\Rasters\pn30.img"
ClippedRasterLayerpn60 = "D:\Path1\Rasters\pn60.img"
ClippedRasterLayerpn90 = "D:\Path1\Rasters\pn90.img"
ClippedRasterLayerpn120 = "D:\Path1\Rasters\pn120.img"
ClippedRasterLayerpn180 = "D:\Path1\Rasters\pn180.img"

arcpy.Clip_management(RasterToBeClippedp30, "#", ClippedRasterLayerp30, ClipLayer,
    "-2147483648", "ClippingGeometry", "MAINTAIN_EXTENT")
arcpy.Clip_management(RasterToBeClippedp60, "#", ClippedRasterLayerp60, ClipLayer,
    "-2147483648", "ClippingGeometry", "MAINTAIN_EXTENT")
arcpy.Clip_management(RasterToBeClippedp90, "#", ClippedRasterLayerp90, ClipLayer,
    "-2147483648", "ClippingGeometry", "MAINTAIN_EXTENT")
arcpy.Clip_management(RasterToBeClippedp120, "#", ClippedRasterLayerp120, ClipLayer,
    "-2147483648", "ClippingGeometry", "MAINTAIN_EXTENT")
arcpy.Clip_management(RasterToBeClippedp180, "#", ClippedRasterLayerp180, ClipLayer,
    "-2147483648", "ClippingGeometry", "MAINTAIN_EXTENT")

```

The following code is the remainder of the script (precipgen.py) before executing the next script (renewservice.py) for renewing map services on ArcGIS for Server. Using the ArcPy module, shapefiles and raster data may be deleted all at once without having to delete each individual file making up a shapefile and raster data. An exception to deleting raster data is that the *info* directory is to be deleted by the Shutil module since the *info* directory is shared by many raster ArcInfo datasets (as a result of intermediate data), which is the default storage raster data type.

```

# Delete existing SDE_StationsLive.shp as a result of running this
# script previously in an effort to keep attributes updated
arcpy.Delete_management("SDE_StationsLiveKYSPTS.shp")
arcpy.Delete_management("StationsLive.shp")

# Delete rasters that begin with the letters "pt" (this is intermediate data)
datasets = arcpy.ListRasters("pt*")
for dataset in datasets:
    arcpy.Delete_management(dataset)

# Delete the info folder as a result from intermediate data processing
shutil.rmtree('info')

```

The next script (renewservice.py) renews the map service on ArcGIS for Server. To run this script, it requires an existing service already residing on ArcGIS for Server. Furthermore, during the process of sharing as a services from ArcMap to set up an initial service as already mention, save a service definition draft file (.sddraft) to be used later for this script. Be warn, the .sddraft file will be buried deep in directories and so you may have to do a global search. The below script is almost taken verbatim from the sample code provided by ESRI (Upload Service, 2014).

```

# Description: Creates a service definition that can be used to overwrite an
#             existing service. When this service definition is published it
#             will overwrite the existing service.
# Requirements: Connection to an ArcGIS Server or My Hosted Services

```

```

# Import system modules
import arcpy
from arcpy import env
import xml.dom.minidom as DOM

import os
import shutil

shutil.copyfile( 'D:\Path1\ExperimentalDrought_Original.sddraft',
                'D:\Path1\ExperimentalDrought.sddraft' )

# Set environment settings
env.workspace = r"D:\Path1"

# Set local variables
inServiceDefinitionDraft = r"ExperimentalDrought.sddraft"
# .sddraft is created by exiting out of Service Editor right before the option
# to actually publish.
outServiceDefinition = r"D:\Path1\SDFFile\ExperimentalDrought.sd"

newType = 'esriServiceDefinitionType_Replacement'
xml = env.workspace + "/" + inServiceDefinitionDraft
doc = DOM.parse(xml)
descriptions = doc.getElementsByTagName('Type')
for desc in descriptions:
    if desc.parentNode.tagName == 'SVCManifest':
        if desc.hasChildNodes():
            desc.firstChild.data = newType
outXml = xml
f = open(outXml, 'w')
doc.writexml( f )
f.close()

# Execute StageService. Note: you may get Error 001269 Compressing the service
# definition failed if accessing other files in same folder as .sd file.
arcpy.StageService_server(inServiceDefinitionDraft, outServiceDefinition)

# Set local variables
inSdFile = outServiceDefinition
inServer = r"C:\Users\Administrator\AppData\Roaming\ESRI\Desktop10.2
\ArcCatalog\AGS_Machine_(publisher).ags"

# Execute UploadServiceDefinition
arcpy.UploadServiceDefinition_server(inSdFile, inServer)

os.remove( 'D:\Path1\SDFFile\ExperimentalDrought.sd' )

```

The trick to getting this script to work is saving the .sddraft file as another name so that it is always there and available for copying as a .sddraft file in which is used by the script. For some reason during the process of renewing the service with this script, the .sddraft file that is used is deleted after implementation, but the same .sddraft file is needed when running this script again. There may be a way to prevent this from happening, but no leads and so, as mentioned already, a work around is provided. To run this code, the 32 bit Python.exe is needed to execute the Python script above (renewservice.py). On some computers the 64 bit Python.exe (default) works well and sometimes not. The work around, run the 32 bit Python.exe in this case.

All rasters created from the above script(s) are available as one map service via ArcGIS for

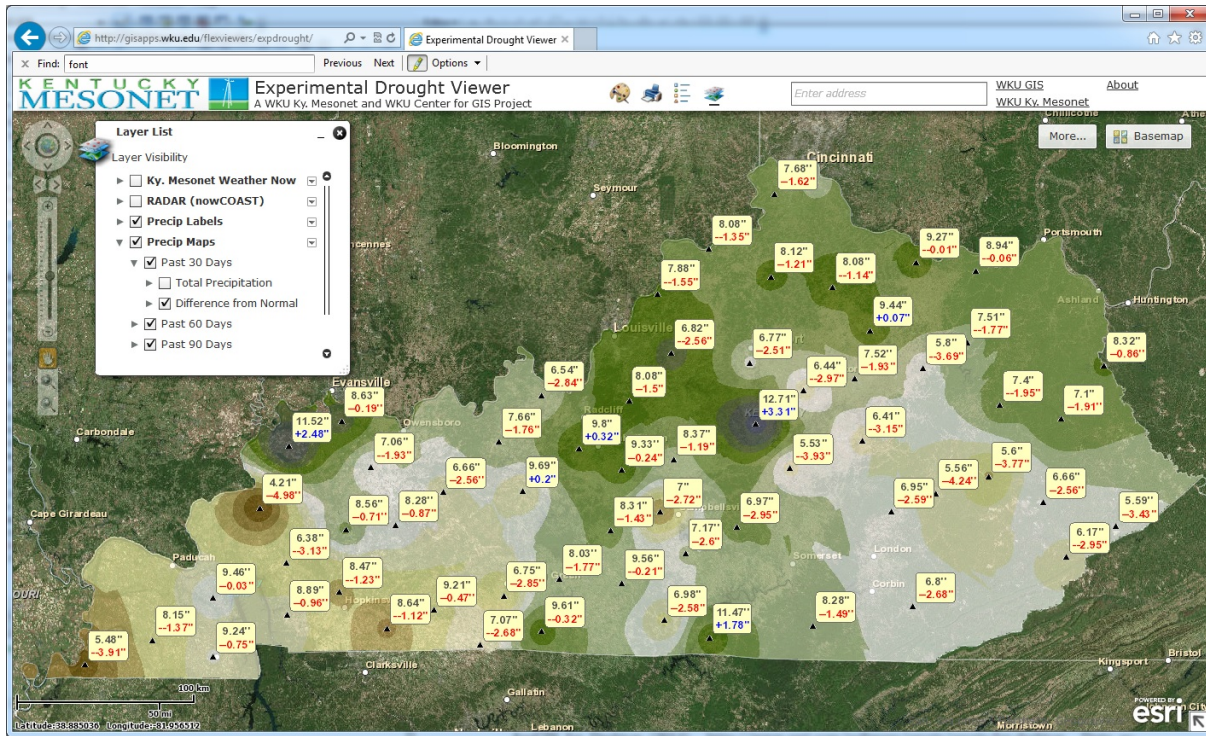


Figure 3: Kentucky Mesonet Experimental Drought Viewer.

Server. In addition, ‘Precipitation Labels’ layers are available as a map service as well to accompany the raster layers in the web map application. The precipitation labels are automatically updated every day too from an existing SDE service. The top value of a label reflects a precipitation amount for that station and the bottom value reflects how much an amount of precipitation deviates from the norm for that station. Map services are meshed into an ArcGIS Viewer for Flex as seen in Figure 3 above.

## Conclusions

This can’t be the best way of updating raster services, but it’s at least a solution that works and it’s dependable. This web map is mostly used by end users, researchers in this case, but the map services are available for retrieving into ArcMap or any other GIS applications accepting WMS and KML. Using the IDW is a start, but the Kriging method is the next investigation for interpolating precipitation values. The script above is expected to work for using the Kriging method and so it’s just a matter of simply copying and replacing the IDW code portion.

## References

- ArcGIS for Server (2014). <http://www.esri.com/software/arcgis/arcgisserver/features/>. Date accessed: June 23, 2014.
- ArcGIS Viewer for Flex (2015). <http://resources.arcgis.com/en/communities/flex-viewer/> Date accessed: June 30, 2015.



- Burke, D. and Dodd, P. (2009). Easy Access to Near Real Time Data. *ArcUser*, 9(2):36 – 38. Spring.
- Clip Data Management (2014). *ArcGIS Help 10.2*. <http://resources.arcgis.com/en/help/main/10.2/>. Date accessed: June 30, 2015.
- Cooper, C. (2012). Real-Time Updating of ArcSDE through SQL. *ArcUser*, 15(1):40 – 45. Winter.
- Delete Data Management (2014). *ArcGIS Help 10.2*. <http://resources.arcgis.com/en/help/main/10.2/>. Date accessed: June 30, 2015.
- Feature Class To Feature Class Conversion (2014). *ArcGIS Help 10.2*. <http://resources.arcgis.com/en/help/main/10.2/>. Date accessed: June 30, 2015.
- IDW Spatial Analyst (2014). *ArcGIS Help 10.2*. <http://resources.arcgis.com/en/help/main/10.2/>. Date accessed: June 30, 2015.
- Kentucky Council of Area Development Districts (2014). <http://www.kcadd.org>. Date accessed: June 30, 2015
- Kentucky Mesonet (2014). <http://www.kymesonet.org/>. Date accessed: June 30, 2015.
- Project Data Management (2014). *ArcGIS Help 10.2*. <http://resources.arcgis.com/en/help/main/10.2/>. Date accessed: June 30, 2015.
- Python Software Foundation (2014). <https://www.python.org/>. Date accessed: June 23, 2014.
- Upload Service Definition Server (2014). *ArcGIS Help 10.2*. <http://resources.arcgis.com/en/help/main/10.2/>. Date accessed: June 30, 2015.
- Western Kentucky University GIScience (2015). <http://www.wku.edu/gis>. Date accessed: June 30, 2015