**ATTACHMENT 1-3: Method for Establishing the Use Site Footprints**

Use site footprint layers represent the application sites for agricultural and non-agricultural label uses. The best available data to spatially characterize specific agricultural crops is the Cropland Data Layer (CDL), produced by the U.S. Department of Agriculture. Several methods have been employed to minimize data errors within the CDL. The CDL is a land cover dataset that has over 100 cultivated classes that were grouped into 11 general classes (see **ATTACHMENT 1-2**). Lumping classes reduces the likelihood of errors of omission and commission between similar crop categories. Additionally, as the CDL is annually produced, 6 years of CDL, from 2010-2015, were aggregated to account for crop rotations, so that anywhere a class occurs within those 6 years would be represented in the footprint layer.

The agricultural classes were further refined by comparing county level National Agricultural Statistics Service (NASS) 2012 Census of Agriculture (CoA) acreage reports to county level CDL acreages (further detail can be found in the [“Processing the Census of Agriculture Data”](#_Processing_the_Census_1) section). The CDL acreages represent the temporally aggregated and categorically grouped processing steps previously described, summarized at the county level. If a county’s CDL acreage for a given class was lower than the NASS acreage, the CDL class’s extent was expanded within cultivated areas until the CDL acreage matched the NASS CoA. Using the temporally and thematically aggregated CDL as an input, a script was developed that compares each CDL crop group in each county to the corresponding NASS CoA acreage report. If the CDL acreage was less than NASS, the raster was expanded in 1 pixel iterations until the NASS acreage value was reached, or the area within the cultivated mask was built out. Region growing was restricted using the most recent CDL Cultivated Layer as a mask (2015), so as to avoid buffering into any non-agricultural land cover types. This method reduced land cover mapping errors by adjusting the extent of each category to the most recent CoA values, in this case, 2012. A detailed version of the python scripts for this process can be found in the [“CDL Processing Python Scripts”](#_CDL_Processing_Python) section.

The CDL is not available for areas outside of the Contiguous United States (CONUS). The CoA is often unavailable outside of CONUS as well. The following list describes how agriculture was spatially modeled by regions outside of CONUS.

* **Alaska (AK)**
  + National Land Cover Dataset (NLCD) Cultivated Class (82)
  + NASS CoA crosswalked to the 11 general crop groups, summarized by county
* **Hawaii (HI)**
  + National Oceanic & Atmospheric Administration (NOAA) Coastal Change Analysis Program (CCAP), Cultivated Class (6)
  + Crosswalk the Hawaiian Agricultural Land Use Map (ALUM) commodity attribute into the general classes discussed for CONUS (**Table 1**)

| Table 1: HI ALUM Crop Crosswalk | |
| --- | --- |
| **ALUM Crop Category** | **General Class** |
| Grazing | Pasture/Hay/Silage |
| Vegetables/Melons | Vegetables & Ground Fruit |
| Forage and Grain | Pasture/Hay/Silage |
| Banana | Orchards & Vineyards |
| Papaya | Orchards & Vineyards |
| Macadamia Nuts | Orchards & Vineyards |
| Avocado | Orchards & Vineyards |
| Coffee | Orchards & Vineyards |
| Guava | Orchards & Vineyards |
| Other Orchard | Orchards & Vineyards |
| Pineapple | Vegetables & Ground Fruit |
| Aquaculture | Other Crops |
| Sugarcane | Other Grains |

* **Puerto Rico (PR)**
  + NLCD Cultivated Class (82)
  + Crosswalk the Puerto Rico Census of Agriculture crop attribute into the general classes discussed for CONUS (**Table 2**)

| Table 2: PR Agriculture census crosswalk | | |
| --- | --- | --- |
| **PR Agriculture Census** | **General Class** | **Double Crop Class** |
| Coconuts | Orchards & Vineyards |  |
| Coffee/Plantain/Fruit Trees | Orchards & Vineyards |  |
| Fruit Trees | Orchards & Vineyards |  |
| Grass | Other Crops |  |
| Pineapple | Vegetables & Ground Fruit |  |
| Plantains/ Tubers | Orchards & Vineyards | Vegetables & Ground Fruit |
| Plantains/Bananas | Orchards & Vineyards |  |
| Plantains/Fruit Trees | Orchards & Vineyards |  |
| Sugarcane | Other Grains |  |
| Tubers | Vegetables & Ground Fruit |  |
| Vegetables | Vegetables & Ground Fruit |  |
| Vegetables/Fruit | Vegetables & Ground Fruit |  |
| Vegetables/Fruit Trees | Vegetables & Ground Fruit | Orchards & Vineyards |

* **Guam (GU)**
  + CCAP Cultivated Class (6)
  + Current CoA is not available for GU
* **Marianas (CNMI)**
  + CCAP Cultivated Class (6)
  + Current CoA is not available for CNMI
* **American Samoa (AS)**
  + CCAP Cultivated Class (6)
  + Current CoA is not available for AS
* **Virgin Islands (VI)**
  + CCAP Cultivated Class (6)
  + Current CoA is not available for VI

Non-agricultural label uses include a wide range of land cover and land use categories. Each label use was carefully considered and cross-walked with the best available land cover data. The 2011 National Land Cover Dataset (NLCD) was used to represent many non-agricultural label uses (see below). These classes were represented using NLCD Developed or Open Space Developed land use categories.

* Grain/cereal/flour bins
* Grain/cereal/flour elevators
* Household/domestic dwellings (perimeter outdoor only)
* Non-agricultural outdoor building structures
* Ornamental and/or shade trees
* Ornamental herbaceous plants
* Ornamental non-flowering plants
* Ornamental woody shrubs and vines
* Refuse/solid waste containers (outdoors)
* Refuse/solid waste sites (outdoors)
* Commercial/Institution-Al/ Industrial Premises/ Equip. (Indoor and Outdoor) – Broadcast, Crack and Crevice/Void
* Domestic Dwellings Outdoor Premises;
* Food Processing Plant Premises (Nonfood Contact) – Crack and Crevice
* Nonagricultural Outdoor Buildings/Structures
* Poultry Litter
* Recreational Areas
* Sewer Manhole Covers and Walls
* Utilities – Broadcast
* Wood Protection Treatment to Buildings/Products Outdoor

When the NLCD was inadequate to represent a label use, other data sources were used in modeling as appropriate. The following list describes each label use and how it was spatially modeled by region.

* **Nurseries**

Nurseries represent a land use that is not exclusive to any nationwide land cover class. Nurseries are mapped by using geocoded Dun and Bradstreet (D&B) business database addresses.

* + **CONUS**
    - Using the Dun and Bradstreet business database, select all records with any SIC Codes starting with “018” (Horticultural Specialties) or “526” (Retail Nurseries, Lawn And Garden Supply Stores)
    - Selected points are then buffered by their facility size attribute. Where facility size is absent, substitute the Census of Agriculture’s average acreage by county, calculated using Nursery Totals. If a county’s nursery acreages are undisclosed, then an average of all county averages is used. A circular buffer is applied, where radius is solved for using the areas previously described. In an effort to map production facilities only and not business offices, use the ‘Location Type’ attribute to categorize locations.
  + **Alaska**
    - Dun and Bradstreet business database was used in the same method as applied to CONUS.
  + **Hawaii**
    - Dun and Bradstreet business database was used in the same method as applied to CONUS.
  + **Puerto Rico**
    - Dun and Bradstreet business database was used in the same method as applied to CONUS.
  + **Guam**
    - No Dun and Bradstreet business data were available for Guam.
  + **Marianas**
    - No Dun and Bradstreet business data were available for Marianas.
  + **American Samoa**
    - No Dun and Bradstreet business data were available for American Samoa.
  + **Virgin Islands**
    - Dun and Bradstreet business database was used in the same method as applied to CONUS.
* **Right-of-Ways**

NLCD developed classes are sufficient for most scenarios. NLCD developed classes are insufficient in cases of rural minor roads, rural transmission lines, and rural pipelines.

* + **CONUS**
    - All NLCD developed classes everywhere (21-24)
    - ESRI Railroads
    - United States Census Bureau’s Topologically Integrated Geographic Encoding and Referencing database (TIGER) transmission (MAF/TIGER Feature Class Code (MTFCC) code L4020) and pipeline (MTFCC code L4010) data
    - Bonneville Power Administration’s (BPA) Right-of-Way data
    - Navteq roads
  + **Alaska**
    - See CONUS method (without BPA data)
  + **Hawaii**
    - All National Oceanic & Atmospheric Administration (NOAA) Coastal Change Analysis Program (CCAP) developed classes everywhere (2-5)
    - ESRI Railroads
    - TIGER transmission (MTFCC code L4020) and pipeline (MTFCC code L4010) data
    - NAVTEQ roads
  + **Puerto Rico**
    - See CONUS method (without BPA data)
  + **Guam**
    - All CCAP developed classes everywhere (2-5)
    - No ESRI Railroads data available for Guam
    - TIGER transmission (MTFCC code L4020) and pipeline (MTFCC code L4010) data
    - No NAVTEQ roads data available for Guam
  + **Marianas**
    - All CCAP developed classes everywhere (2-5)
    - No ESRI Railroads data available for Marianas
    - TIGER transmission (MTFCC code L4020) and pipeline (MTFCC code L4010) data
    - No NAVTEQ roads data available for Marianas
  + **American Samoa**
    - All CCAP developed classes everywhere (2-5)
    - No ESRI Railroads data available for American Samoa
    - No TIGER data available for American Samoa
    - No NAVTEQ roads data available for American Samoa
  + **Virgin Islands**
    - All CCAP developed classes everywhere (2-5)
    - No ESRI Railroads data available for Virgin Islands
    - No TIGER data available for Virgin Islands
    - No NAVTEQ roads data available for Virgin Islands
* **Managed Forests**

Forested areas managed for timber extraction

* + **CONUS**
    - Include all the following LandFire Existing Vegetation Type (EVT) classes; "Recently Logged-Herb and Grass Cover", "Recently Logged-Shrub Cover", "Recently Logged-Tree Cover", "Managed Tree Plantation-Northern and Central Hardwood and Conifer Plantation Group", or "Managed Tree Plantation-Southeast Conifer and Hardwood Plantation Group"
    - Include any of the following United States Geologic Survey (USGS) National Gap Analysis Program (GAP) Public Model Ready Events; "Thinning", "Other Mechanical", "Clearcut", "Harvest", or "Reforestation"
    - Include any of the following USGS GAP Land Cover classes; "Recently Logged Areas", "Harvested Forest - Grass/Forb Regeneration", "Harvested Forest-Shrub Regeneration", "Harvested Forest - Northwestern Conifer Regeneration", "Managed Tree Plantation", "Evergreen Plantation or Managed Pine", "Deciduous Plantations"
    - Include either of the following USGS GAP Protected Areas Database classes where NLCD indicates "Forest" (41-43); "3 - managed for multiple uses - subject to extractive (*e.g*., mining or logging) or Off Highway Vehicles (OHV) use" and "4 - no known mandate for protection"
  + **Alaska**
    - Include either of the following USGS GAP Protected Areas Database classes where NLCD indicates "Forest" (41-43); "3 - managed for multiple uses - subject to extractive (*e.g*., mining or logging) or OHV use" and "4 - no known mandate for protection"
    - Include any of the following USGS GAP Public Model Ready Events; "Thinning", "Other Mechanical", "Clearcut", "Harvest", or "Reforestation"
    - AK LandFire EVT and GAP land cover do not have classes indicative of forest management
  + **Hawaii**
    - Include the following LandFire EVT class; “Hawai'i Managed Tree Plantation”
    - Include either of the following USGS GAP Protected Areas Database classes where CCAP indicates "Forest" (9-11); "3 - managed for multiple uses - subject to extractive (*e.g*., mining or logging) or OHV use" and "4 - no known mandate for protection"
    - HI GAP land cover and USGS GAP Public Model Ready Events for HI do not have classes indicative of forest management
  + **Puerto Rico**
    - Include the following GAP land cover classes; “Abandoned dry forest plantation”, “Woody agriculture and plantations: Palm plantations”
    - Include either of the following USGS GAP Protected Areas Database classes where CCAP indicates "Forest" (9-11); "3 - managed for multiple uses - subject to extractive (*e.g.*, .mining or logging) or OHV use" and "4 - no known mandate for protection"
    - PR LandFire EVT is not available
  + **Guam**
    - Include either of the following USGS GAP Protected Areas Database classes where CCAP indicates "Forest" (9-11); "3 - managed for multiple uses - subject to extractive (*e.g*., mining or logging) or OHV use" and "4 - no known mandate for protection"
    - LandFire EVT, GAP land cover, and USGS GAP Public Model Ready Events are not available for Guam
  + **Marianas**
    - Include either of the following USGS GAP Protected Areas Database classes where CCAP indicates "Forest" (9-11); "3 - managed for multiple uses - subject to extractive (*e.g*., mining or logging) or OHV use" and "4 - no known mandate for protection"
    - LandFire EVT, GAP land cover, and USGS GAP Public Model Ready Events are not available for the Marianas
  + **American Samoa**
    - LandFire EVT, GAP land cover, and USGS GAP Public Model Ready Events are not available for the Marianas
    - USGS GAP Protected Areas Database does not indicate areas indicative of forest management
  + **Virgin Islands**
    - Include either of the following USGS GAP Protected Areas Database classes where CCAP indicates "Forest" (9-11); "3 - managed for multiple uses - subject to extractive (*e.g*., mining or logging) or OHV use" and "4 - no known mandate for protection"
    - LandFire EVT, GAP land cover, and USGS GAP Public Model Ready Events are not available for the Marianas
* **Open Space Developed**

Open Space Developed (OSD) is used to spatially represent certain non-agricultural label uses

* + **CONUS**
    - NLCD class 21
  + **Alaska**
    - NLCD class 21
  + **Hawaii**
    - CCAP class 5
  + **Puerto Rico**
    - NLCD class 21
  + **Guam**
    - CCAP class 5
  + **Marianas**
    - CCAP class 5
  + **American Samoa**
    - CCAP class 5
  + **Virgin Islands**
    - CCAP class 5
* **Developed**

Developed land cover is used to spatially represent certain non-agricultural label uses

* + **CONUS**
    - NLCD class 22-24
  + **Alaska**
    - NLCD class 22-24
  + **Hawaii**
    - CCAP class 2-4
  + **Puerto Rico**
    - NLCD class 22-24
  + **Guam**
    - CCAP class 2
  + **Marianas**
    - CCAP class 2
  + **American Samoa**
    - CCAP class 2
  + **Virgin Islands**
    - CCAP class 2
* **Cattle Ear Tags**

Cattle ear tag use is best characterized by spatially mapping rangeland. The CDL and NLCD map a pasture class, but this is primarily grassland pastures. The grazing cattle land use is added to additional land cover types, such as forests, shrublands, wetlands, etc.

* + **CONUS**
    - CDL and NLCD pasture classes everywhere
    - Undeveloped NLCD classes within Bureau of Land Management (BLM) and United States Forest Service (USFS) grazing allotment boundaries
      * *Exclude NLCD developed, water, and cultivated*
  + **Alaska**
    - NLCD pasture class everywhere
    - Undeveloped NLCD classes within BLM grazing allotment boundaries
    - No USFS grazing allotment boundaries available for AK
  + **Hawaii**
    - CCAP pasture class 7
    - No BLM or USFS grazing allotment boundaries available for HI
  + **Puerto Rico**
    - NLCD pasture class 81
    - No BLM or USFS grazing allotment boundaries available for PR
  + **Guam**
    - CCAP pasture class 7
    - No BLM or USFS grazing allotment boundaries available for GU
  + **Marianas**
    - CCAP pasture class 7
    - No BLM or USFS grazing allotment boundaries available for CNMI
  + **American Samoa**
    - CCAP pasture class 7
    - No BLM or USFS grazing allotment boundaries available for AS
  + **Virgin Islands**
    - CCAP pasture class 7
    - No BLM or USFS grazing allotment boundaries available for VI
* **Pine Seed Orchards**

Specifically for slash pine seed stock

* + **CONUS**
    - Include areas where USFS Slash Pine Presence co-occurs with “Managed Forests” (see above)
    - Exclude National Forests
* **Cull Piles**

Culled orchard fruit

* + **CONUS**
    - Orchards & Vineyards general class derived from CDL as outlined in the beginning of this appendix

# Processing the Census of Agriculture Data

NASS Census of Agriculture commodities on acres harvested were obtained for the 2012 census period and consolidated into normalized tables. The online QuickStats application was used at the county level for acres harvested (<https://quickstats.nass.usda.gov/>). A cross-walk of the relationship between 73 different crops identified in the 2012 NASS Census of Agriculture with the 11 General Class Cropland Data Layer (CDL) categories is provided in [Table 3](#_Table_3._Relationship). A summary table at the county level was then made which totaled the acreage of each individual NASS record for the crop within the CDL general class. From this, a cross-tab table was made showing the total acreage value at the intersection of a given Federal Information Processing Standard (FIPS)/county with the General Class. Counties where the acreage total was withheld to protect farmer confidentiality were assigned the value of -1 in the compiled data and this was folded into the total sum. Most of the data processing was performed via a series of Structured Query Language (SQL) queries in MS Access 2013. For details on the specific queries run see [Code A](#_Code_A:_Formatting).

# Code A: Formatting Census of Agriculture Data

Most of the data processing for formatting the Census of Agriculture Data was performed via a series of Structured Query Language (SQL) queries in MS Access 2013.

In MS Access 2013 perform the following queries:

This first query takes the compiled NASS table from QuickStats and aggregates it into the GenClass11 categories ([Table 3](#_Table_3._Relationship)). It filters on Acres >0. It is called qMkTbl\_FIPS\_GenClass\_Acres\_2012:

1. **SELECT** ztbl\_NASS2012\_compiled.FIPS, trel\_NASS\_GenClass11.GenClassID\_fk, Sum(ztbl\_NASS2012\_compiled.Value) **AS** SumOfValue **INTO** tbl\_FIPS\_GenClass\_Acres\_2012
2. **FROM** trel\_NASS\_GenClass11 **INNER** JOIN ztbl\_NASS2012\_compiled **ON** trel\_NASS\_GenClass11.NASS2012\_crop = ztbl\_NASS2012\_compiled.Commodity
4. **GROUP** **BY** ztbl\_NASS2012\_compiled.FIPS, trel\_NASS\_GenClass11.GenClassID\_fk
5. **HAVING** (((Sum(ztbl\_NASS2012\_compiled.Value))>0));

The next series takes the above table and pivots it by the classes which become headers. It is called qxTab\_FIPS\_CDL\_xTab:

1. TRANSFORM Sum(tbl\_FIPS\_GenClass\_Acres\_2012.SumOfValue) **AS** SumOfSumOfValue
2. **SELECT** tbl\_FIPS\_GenClass\_Acres\_2012.FIPS
3. **FROM** tbl\_FIPS\_GenClass\_Acres\_2012
4. **GROUP** **BY** tbl\_FIPS\_GenClass\_Acres\_2012.FIPS
5. PIVOT tbl\_FIPS\_GenClass\_Acres\_2012.GenClassID\_fk;

# Table 3. Relationship between NASS Census and CDL general class categories.

| **NASS Census of Ag 2012** | **CDL General Class** |
| --- | --- |
| ARTICHOKES | Vegetables/Ground Fruit |
| ASPARAGUS | Vegetables/Ground Fruit |
| BARLEY | Other Grains |
| BEANS | Vegetables/Ground Fruit |
| BEETS | Vegetables/Ground Fruit |
| BROCCOLI | Vegetables/Ground Fruit |
| BRUSSELS SPROUTS | Vegetables/Ground Fruit |
| BUCKWHEAT | Other Grains |
| CABBAGE | Vegetables/Ground Fruit |
| CANOLA | Other Grains |
| CARROTS | Vegetables/Ground Fruit |
| CAULIFLOWER | Vegetables/Ground Fruit |
| CELERY | Vegetables/Ground Fruit |
| CHICORY | Vegetables/Ground Fruit |
| CORN | Corn |
| COTTON | Cotton |
| CRAMBE | Vegetables/Ground Fruit |
| CUCUMBERS | Vegetables/Ground Fruit |
| DAIKON | Vegetables/Ground Fruit |
| DILL | Vegetables/Ground Fruit |
| EGGPLANT | Vegetables/Ground Fruit |
| EMMER & SPELT | Other Grains |
| ESCAROLE & ENDIVE | Vegetables/Ground Fruit |
| FIELD CROPS, OTHER | Other Row Crops |
| FLAXSEED | Other Grains |
| GARLIC | Vegetables/Ground Fruit |
| GINGER ROOT | Vegetables/Ground Fruit |
| GRASSES & LEGUMES TOTALS | Other Crops |
| GUAR | Other Crops |
| HAY & HAYLAGE | Hay/Forage/Silage |
| HERBS | Vegetables/Ground Fruit |
| HOPS | Vegetables/Ground Fruit |
| JOJOBA | Orchards/Vineyards |
| LEGUMES | Other Crops |
| LENTILS | Vegetables/Ground Fruit |
| MILLET | Other Grains |
| MINT | Vegetables/Ground Fruit |
| OATS | Other Grains |
| ORCHARDS | Orchards/Vineyards |
| PEANUTS | Vegetables/Ground Fruit |
| PEAS | Vegetables/Ground Fruit |
| PEPPERS | Vegetables/Ground Fruit |
| POPCORN | Vegetables/Ground Fruit |
| POTATOES | Vegetables/Ground Fruit |
| PUMPKINS | Vegetables/Ground Fruit |
| RADISHES | Vegetables/Ground Fruit |
| RAPESEED | Other Grains |
| RHUBARB | Vegetables/Ground Fruit |
| RICE | Rice |
| RYE | Other Grains |
| SAFFLOWER | Other Grains |
| SESAME | Other Grains |
| SORGHUM | Other Grains |
| SOYBEANS | Soybeans |
| SPINACH | Vegetables/Ground Fruit |
| SQUASH | Vegetables/Ground Fruit |
| SUGARBEETS | Vegetables/Ground Fruit |
| SUGARCANE | Other Grains |
| SUNFLOWER | Other Row Crops |
| SWEET CORN | Vegetables/Ground Fruit |
| SWEET POTATOES | Vegetables/Ground Fruit |
| SWEET RICE | Rice |
| SWITCHGRASS | Hay/Forage/Silage |
| TARO | Vegetables/Ground Fruit |
| TOBACCO | Vegetables/Ground Fruit |
| TOMATOES | Vegetables/Ground Fruit |
| TRITICALE | Other Grains |
| TURNIPS | Vegetables/Ground Fruit |
| VEGETABLES, MIXED | Vegetables/Ground Fruit |
| VEGETABLES, OTHER | Vegetables/Ground Fruit |
| WATERCRESS | Vegetables/Ground Fruit |
| WHEAT | Wheat |
| WILD RICE | Rice |

# CDL Processing Python Scripts

This section contains text copied from 8 sequential python scripts, plus two supplemental tables used in the code. All sections have been copy/pasted from Python Files into this word document, and as a result, the integrity of indentation, wrapping, and comment symbology (##) should be carefully reviewed when repasting into integrated development environments (IDE). All paths are relative and need updating. Suggested directory structure needs manual creation in most cases. These scripts demonstrate 6 years of CDL (2010-2015); the previously released draft was 5 years (2010-2014). The user should edit scripts as needed for timeframe desired. Please contact Steve Lennartz via email (lennartz.steven@epa.gov) to receive the \*.py files.

# Table of Scripts

[1\_GenClass11\_Recode\_Script1\_1015.py](#_Script_1:_“1_GenClass11_Recode_Scri_1)

[2\_GenClass11\_Recode\_Script2\_1015.py](#_Script_2:_“2_GenClass11_Recode_Scri)

[3\_CDL\_CountyClipping\_1015.py](#_Script_3:_“3_CDL_CountyClipping_101)

[4\_CDL\_UpdateAreaValues\_1015.py](#_Script_4:_“4_CDL_UpdateAreaValues_1)

[5\_Cultivated\_CountyClipping.py](#_Script_5:_“5_Cultivated_CountyClipp)

[6\_CDL\_Expand\_1015.py](#_Script_6:_“6_CDL_Expand_1015.py”)

[7\_CDL\_MosaicExpanded\_1015.py](#_Script_7:_“7_CDL_MosaicExpanded_101)

[8\_CDL\_FixHoles\_1015.py](#_Script_8:_“8_CDL_FixHoles_1015.py”)

[Table 4: cdl\_recode\_new.txt](#_Supplemental_Table:_\“cdl_recode_new)

[Table 5: EPA\_CDL\_Recode\_Key.txt](#_Supplemental_Table:_“EPA_CDL_Recode)

# Script 1: “1\_GenClass11\_Recode\_Script1\_1015.py”

1. ## Script 1. First Script to aggregate CDL classes. Requires each year of CDL to be recoded as describe in, "ATTACHMENT
2. ## 1-2: CDL Crosswalk (DOCX)", of Implementing NAS Report Recommendations on Ecological Risk Assessment for Endangered
3. ## and Threatened Species, https://www3.epa.gov/pesticides/nas/attachment-1-2.docx . This can be done using ArcToolbox
4. ## Spatial Analyst Tools/Reclass/Reclass by Table, using "cdl\_recode\_new.txt" loaded as a remap table. These scripts
5. ## demonstrate 6 years of CDL (2010-2015); the previously released draft was 5 years (2010-2014). Edit scripts as needed
6. ## for timeframe desired. All paths are relative and need updating. Suggested directory structure needs manual creation
7. ## in most cases.
9. \_\_author\_\_ = 'Steven Lennartz'
11. ## Import Site Packages
12. **import** arcpy
13. **from** arcpy **import** env
14. **from** arcpy.sa **import** \*
15. **import** datetime
17. ## Check out the ArcGIS Spatial Analyst extension license
18. arcpy.CheckOutExtension("Spatial")
20. ## Set variables. All paths are relative and need updating.
21. ## Create a File Geodatabase, populated with recoded CDL layers as described in the header. This will be your in-
22. ## workspace.
23. inws = r"../CDL\_Recodes/CDL\_Reclass\_1015.gdb/"
24. ## Create a File Geodatabase for an out-workspace
25. outws = r"../CDL\_Recodes/CDL\_GenAggregates\_1015.gdb/"
26. ## The cultivated layer from the CDL website
27. cult = Raster("../Cultivated\_Layer/2015\_Cultivated\_Layer/2015\_Cultivated\_Layer.img")
29. ## Environmental settings
30. arcpy.env.workspace = inws
31. arcpy.env.overwriteOutput = 1
32. arcpy.env.snapRaster = cult
33. arcpy.env.extent = cult
34. arcpy.env.scratchWorkspace = outws
35. arcpy.env.parallelProcessingFactor = "100%"
37. ## Create a list of the rasters in the in-workspace
38. ras = arcpy.ListRasters()
40. ## The Key is included as a txt (add file extension if using txt format)
41. key = "EPA\_CDL\_Recode\_Key"
42. valueList = {row[0] **for** row **in** arcpy.da.SearchCursor(key, "Value")}
44. start\_t1 = datetime.datetime.now()
46. ## Aggregate multiple years of CDL
47. ## For loop
48. **for** value **in** valueList:
49. **if** arcpy.Exists("{}CDL\_1015\_{}".format(outws, value)):
50. **print** "{} Exists".format("CDL\_1015\_{}".format(value))
51. **else**:
52. start\_t = datetime.datetime.now()
53. **print** "Temporally Aggregating Class {} at {}".format(value, start\_t)
54. ## Conditional statement aggregating 6 years of CDL
55. out = Con(((Raster(ras[0]) == value) | (Raster(ras[1]) == value) | (Raster(ras[2]) == value) | (Raster(ras[
56. 3]) == value) | (Raster(ras[4]) == value) | (Raster(ras[5]) == value)), 1, 0)
57. out.save("{}CDL\_1015\_{}".format(outws, value))
58. **print** "Building Pyramids....."
59. arcpy.BuildPyramids\_management("{}CDL\_1015\_{}".format(outws, value))
60. **print** "Completed Temporal Aggregation of Class {} in: {}".format(value, datetime.datetime.now() - start\_t)
62. **print** "Completed Temporal Aggregation of all Classes in: {}".format(datetime.datetime.now() - start\_t1)

# Script 2: “2\_GenClass11\_Recode\_Script2\_1015.py”

1. ## Script 2. Second script to aggregate CDL classes over multiple years.
3. \_\_author\_\_ = 'Steven Lennartz'
5. ## Import Site Packages
6. **import** arcpy
7. **from** arcpy **import** env
8. **from** arcpy.sa **import** \*
9. **import** datetime
11. ## Check out the ArcGIS Spatial Analyst extension license
12. arcpy.CheckOutExtension("Spatial")
14. ## Set variables
15. ## In-workspace is the out-workspace from step 1
16. inws = r"../CDL\_Recodes/CDL\_GenAggregates\_1015.gdb/"
17. ## Create a File Geodatabase for an out-workspace
18. outws = r"../CDL\_Recodes/CDL\_GenClass11\_1015.gdb/"
19. ## The cultivated layer from the CDL website
20. cult = Raster("../Cultivated\_Layer/2015\_Cultivated\_Layer/2015\_Cultivated\_Layer.img")
21. ## From and to years of CDL. This example is 6 years, 2010-2015, "CDL\_1015\_". For use in formatting strings.
22. cdl = "CDL\_1015\_"
24. ## Environmental settings
25. arcpy.env.workspace = inws
26. arcpy.env.overwriteOutput = 1
27. arcpy.env.snapRaster = cult
28. arcpy.env.extent = cult
29. arcpy.env.scratchWorkspace = outws
30. arcpy.env.parallelProcessingFactor = "100%"
32. start\_t1 = datetime.datetime.now()
34. ## Conditional statements producing 11 general CDL layers
35. start\_t = datetime.datetime.now()
36. **print** "Recoding Corn at {}".format(start\_t)
37. out1 = Con(((Raster("{}10".format(cdl)) == 1) | (Raster("{}14".format(cdl)) == 1) | (Raster("{}15".format(cdl)) == 1)
38. | (Raster("{}18".format(cdl)) == 1)), 1, 0)
39. out1.save("{}{}10".format(outws, cdl))
40. **print** "Building pyramids........"
41. arcpy.BuildPyramids\_management("{}{}10".format(outws, cdl))
42. **print** "Completed Corn recode in: {}".format(datetime.datetime.now() - start\_t)
44. start\_t = datetime.datetime.now()
45. **print** "Recoding Cotton at {}".format(start\_t)
46. out2 = Con(((Raster("{}20".format(cdl)) == 1) | (Raster("{}25".format(cdl)) == 1) | (Raster("{}26".format(cdl)) == 1)
47. | (Raster("{}42".format(cdl)) == 1)), 1, 0)
48. out2.save("{}{}20".format(outws, cdl))
49. **print** "Building pyramids........"
50. arcpy.BuildPyramids\_management("{}{}20".format(outws, cdl))
51. **print** "Completed Cotton recode in: {}".format(datetime.datetime.now() - start\_t)
53. start\_t = datetime.datetime.now()
54. **print** "Copying Rice at {}".format(start\_t)
55. arcpy.Copy\_management("{}30".format(cdl), "{}{}30".format(outws, cdl))
56. **print** "Completed Rice copy in: {}".format(datetime.datetime.now() - start\_t)
58. start\_t = datetime.datetime.now()
59. **print** "Recoding Soybeans at {}".format(start\_t)
60. out4 = Con(((Raster("{}40".format(cdl)) == 1) | (Raster("{}42".format(cdl)) == 1) | (Raster("{}45".format(cdl)) == 1)
61. | (Raster("{}48".format(cdl)) == 1) | (Raster("{}14".format(cdl)) == 1)), 1, 0)
62. out4.save("{}{}40".format(outws, cdl))
63. **print** "Building pyramids........"
64. arcpy.BuildPyramids\_management("{}{}40".format(outws, cdl))
65. **print** "Completed Soybeans recode in: {}".format(datetime.datetime.now() - start\_t)
67. start\_t = datetime.datetime.now()
68. **print** "Recoding Wheat at {}".format(start\_t)
69. out5 = Con(((Raster("{}50".format(cdl)) == 1) | (Raster("{}56".format(cdl)) == 1) | (Raster("{}58".format(cdl)) == 1)
70. | (Raster("{}15".format(cdl)) == 1) | (Raster("{}25".format(cdl)) == 1) | (Raster("{}45".format(cdl)) ==
71. 1)), 1, 0)
72. out5.save("{}{}50".format(outws, cdl))
73. **print** "Building pyramids........"
74. arcpy.BuildPyramids\_management("{}{}50".format(outws, cdl))
75. **print** "Completed Wheat recode in: {}".format(datetime.datetime.now() - start\_t)
77. start\_t = datetime.datetime.now()
78. **print** "Recoding Veg & Ground Fruit at {}".format(start\_t)
79. out6 = Con(((Raster("{}60".format(cdl)) == 1) | (Raster("{}61".format(cdl)) == 1) | (Raster("{}68".format(cdl)) == 1)
80. | (Raster("{}26".format(cdl)) == 1) | (Raster("{}56".format(cdl)) == 1)), 1, 0)
81. out6.save("{}{}60".format(outws, cdl))
82. **print** "Building pyramids........"
83. arcpy.BuildPyramids\_management("{}{}60".format(outws, cdl))
84. **print** "Completed Veg & Ground Fruit recode in: {}".format(datetime.datetime.now() - start\_t)
86. start\_t = datetime.datetime.now()
87. **print** "Copying Orchards & Vineyards at {}".format(start\_t)
88. arcpy.Copy\_management("{}70".format(cdl), "{}{}70".format(outws, cdl))
89. **print** "Completed Orchards & Vineyards copy in: {}".format(datetime.datetime.now() - start\_t)
91. start\_t = datetime.datetime.now()
92. **print** "Recoding Other Grains at {}".format(start\_t)
93. out8 = Con(((Raster("{}80".format(cdl)) == 1) | (Raster("{}18".format(cdl)) == 1) | (Raster("{}48".format(cdl)) == 1)
94. | (Raster("{}58".format(cdl)) == 1)), 1, 0)
95. out8.save("{}{}80".format(outws, cdl))
96. **print** "Building pyramids........"
97. arcpy.BuildPyramids\_management("{}{}80".format(outws, cdl))
98. **print** "Completed Other Grains recode in: {}".format(datetime.datetime.now() - start\_t)
100. start\_t = datetime.datetime.now()
101. **print** "Copying Other Row Crops at {}".format(start\_t)
102. arcpy.Copy\_management("{}90".format(cdl), "{}{}90".format(outws, cdl))
103. **print** "Completed Other Row Crops copy in: {}".format(datetime.datetime.now() - start\_t)
105. start\_t = datetime.datetime.now()
106. **print** "Copying Other Crops at {}".format(start\_t)
107. arcpy.Copy\_management("{}100".format(cdl), "{}{}100".format(outws, cdl))
108. **print** "Completed Other Crops copy in: {}".format(datetime.datetime.now() - start\_t)
110. start\_t = datetime.datetime.now()
111. **print** "Copying Pasture, Hay, Silage at {}".format(start\_t)
112. arcpy.Copy\_management("{}110".format(cdl), "{}{}110".format(outws, cdl))
113. **print** "Completed Pasture, Hay, Silage copy in: {}".format(datetime.datetime.now() - start\_t)
115. **print** "Copying Key....."
116. arcpy.Copy\_management("../CDL\_Recodes/CDL\_GenClass11\_1015.gdb/CDL\_GenClass11\_key",
117. "{}CDL\_GenClass11\_key".format(outws))
119. **print** "Completed 11 Class Recode in: {}".format(datetime.datetime.now() - start\_t1)

# Script 3: “3\_CDL\_CountyClipping\_1015.py”

1. ## Script 3. This script clips each raster from script 2 by county. These will serve as an input for the expansion script.
3. \_\_author\_\_ = 'Steven Lennartz'
5. ## Import Site Packages
6. **import** arcpy
7. **import** datetime
8. **from** arcpy **import** env
9. **from** arcpy.sa **import** \*
11. ## Check out the ArcGIS Spatial Analyst extension license
12. arcpy.CheckOutExtension("Spatial")
14. ## Set variables
15. ## In-workspace is the out-workspace from step 2
16. inws = r"../CDL\_Recodes/CDL\_GenClass11\_1015.gdb/"
17. ## Output directory for
18. outdir = r"../Cnty\_Clips/1015/"
19. ## Tiger Line County boundaries, vector
20. cnty = r"../Boundaries/St\_Cnty\_Boundaries.gdb/tl\_2013\_cnty\_conus\_alb"
21. ## Unique ID for counties, "FIPS code"
22. fipscode = "GEOID"
23. ## The cultivated layer from the CDL website
24. cult = Raster("../Cultivated\_Layer/2015\_Cultivated\_Layer/2015\_Cultivated\_Layer.img")
26. start\_t = datetime.datetime.now()
28. ## Environmental Settings
29. arcpy.env.overwriteOutput = 1
30. arcpy.env.workspace = inws
31. arcpy.env.snapRaster = cult
32. arcpy.env.parallelProcessingFactor = "100%"
33. **print** "Starting process at: {}".format(start\_t)
35. ## List county FIPS codes
36. uniqueFips = {(row[0], row[1]) **for** row **in** arcpy.da.SearchCursor(cnty, (fipscode, "SHAPE@"))}
38. ## Create list of rasters in current workspace
39. rasList = arcpy.ListRasters()
41. ## Clip each raster by county FIPS code
42. ## Environmental setting, extent
43. arcpy.env.extent = "MAXOF"
44. clipfc = "clip\_lyr"
45. ## Make county feature layer
46. arcpy.MakeFeatureLayer\_management(cnty, clipfc)
48. ## Loop through raster list
49. **for** raster **in** rasList:
50. start\_t2 = datetime.datetime.now()
51. ## return string crop code from raster name
52. arcpy.env.workspace = inws
53. desc = arcpy.Describe(raster)
54. name = desc.name
55. val = name.split('\_')[2]
57. ## check for existing geodatabase
58. outws = r"{}CDL\_{}\_cnty.gdb".format(outdir, val)
59. **if** **not** arcpy.Exists(outws):
60. arcpy.CreateFileGDB\_management(outdir, "CDL\_{}\_cnty.gdb".format(val))
61. arcpy.env.workspace = outws
63. ## Loop through counties to clip raster layers
64. **for** fips, shape **in** uniqueFips:
65. start\_t1 = datetime.datetime.now()
66. extent = shape.extent
67. extent = "{} {} {} {}".format(extent.XMin, extent.YMin, extent.XMax, extent.YMax)
68. arcpy.SelectLayerByAttribute\_management(clipfc, "NEW\_SELECTION", "{} = '{}'".format(fipscode, fips))
69. in\_file = "{}{}".format(inws, raster)
70. out\_file = "{}/CDL\_{}\_{}".format(outws, val, fips)
71. **if** **not** arcpy.Exists(out\_file):
72. **print** "Clipping CDL\_{}\_{}...".format(val, fips)
73. arcpy.Clip\_management(in\_file, extent, out\_file, clipfc, "2147483647", "ClippingGeometry", "NO\_MAINTAIN\_EXTENT")
74. **print** "Finished clip in: {}".format(datetime.datetime.now() - start\_t1)
75. **print** "Completed Clipping {} in: {}".format(val, datetime.datetime.now() - start\_t2)
76. **print** "Completed All County Clipping in: {}".format(datetime.datetime.now() - start\_t)

# Script 4: “4\_CDL\_UpdateAreaValues\_1015.py”

1. ## Script 4. This script updates CDL acreage values for the table that compares CDL and Census county acreages. It requires
2. ## a geodatabase populated with tables generated using 'Spatial Analyst/Zonal/Tabulate Area' using each layer generated
3. ## in script 2 and the Tiger county boundaries.
5. \_\_author\_\_ = 'Steven Lennartz'
7. ## Import Site Packages
8. **import** arcpy
9. **import** datetime
10. **from** arcpy **import** env
11. **from** arcpy.sa **import** \*
13. ## Set variables
14. ## In-workspace
15. inws = r"../Analysis/NASS2012\_CDL1015.gdb/"
16. ## Master table. Table header should have FIPS code, 11 CDL columns, and 11 Census columns.
17. intbl = "NASS2012\_Genclass11"
19. ## Environmental Settings
20. arcpy.env.workspace = inws
21. arcpy.CheckOutExtension("Spatial")
22. arcpy.env.overwriteOutput = 1
23. arcpy.env.parallelProcessingFactor = "100%"
25. start\_t1 = datetime.datetime.now()
27. ## Create a table view from master table
28. **print** "Converting {} to Table View at: {}".format(intbl, start\_t1)
29. tblView = "TableView"
30. arcpy.MakeTableView\_management(intbl, tblView)
32. ## List tables generated from "tabulate area" command, described at the top of this script.
33. tblList = arcpy.ListTables("\*area")
35. ## Loop individual area tables to update master table
36. **for** tbl **in** tblList:
37. start\_t = datetime.datetime.now()
38. **print** "Processing table {} at: {}".format(tbl, start\_t)
39. ## return string crop code from tbl name
40. desc = arcpy.Describe(tbl)
41. name = desc.name
42. val = name.split('\_')[2]
44. ## addjoin intbl with tbl (FIPS, GEOID)
45. arcpy.AddJoin\_management(tblView, "FIPS", tbl, "GEOID", "KEEP\_COMMON")
47. ## calculate corresponding CDL field by converting VALUE\_1 from square meters to acres (1 sq meter is 0.000247 acres)
48. outval = "{}.CDL\_{}".format(intbl, val)
49. inval = "CDL\_1015\_{}\_cnty\_area.VALUE\_1".format(val)
50. arcpy.CalculateField\_management(tblView, outval, "(!{}! \* 0.000247)".format(inval), "PYTHON")
51. arcpy.RemoveJoin\_management(tblView)
52. **print** "Completed {} table update in: ".format(datetime.datetime.now() - start\_t)
54. **print** "Completed updating all tables in: {}".format(datetime.datetime.now() - start\_t1)

# Script 5: “5\_Cultivated\_CountyClipping.py”

1. ## Script 5. This script populates a geodatabase with CDL Cultivated Mask rasters by county. These serve as a mask for
2. ## the expansion script.
4. \_\_author\_\_ = 'Steven Lennartz'
6. ## Import Site Packages
7. **import** arcpy
8. **import** datetime
9. **from** arcpy **import** env
10. **from** arcpy.sa **import** \*
12. ## Check out the ArcGIS Spatial Analyst extension license
13. arcpy.CheckOutExtension("Spatial")
15. ## Set variables
16. ## Output directory same as cultivated layer
17. outdir = r"../Cultivated\_Layer/2015\_Cultivated\_Layer/"
18. ## Tiger Line county boundaries, vector
19. cnty = r"../Boundaries/St\_Cnty\_Boundaries.gdb/tl\_2013\_cnty\_conus\_alb"
20. ## Unique ID for counties, "FIPS code"
21. fipscode = "GEOID"
22. ## The cultivated layer from the CDL website
23. cult = Raster("{}2015\_Cultivated\_Layer.img".format(outdir))
25. start\_t = datetime.datetime.now()
27. ## Environmental Settings
28. arcpy.env.overwriteOutput = 1
29. arcpy.env.snapRaster = cult
30. arcpy.env.parallelProcessingFactor = "100%"
32. **print** "Starting process at: {}".format(start\_t)
34. ## List county FIPS codes
35. uniqueFips = {(row[0], row[1]) **for** row **in** arcpy.da.SearchCursor(cnty, (fipscode, "SHAPE@"))}
37. ## Check for output geodatabase
38. **if** **not** arcpy.Exists("{}2015\_Cultivated\_Layer.gdb".format(outdir)):
39. **print** "Creating 2015\_Cultivated\_Layer.gdb"
40. arcpy.CreateFileGDB\_management(outdir, "2015\_Cultivated\_Layer.gdb")
41. **if** **not** arcpy.Exists("{}2015\_Cultivated\_Layer.gdb/cultmask\_2015".format(outdir)):
42. ## Recode CDL Cultivated Layer
43. start\_t2 = datetime.datetime.now()
44. **print** "Recoding 2015\_Cultivated\_Layer.img"
45. arcpy.env.workspace = "{}2015\_Cultivated\_Layer.gdb".format(outdir)
46. arcpy.env.scratchWorkspace = "{}2015\_Cultivated\_Layer.gdb".format(outdir)
47. arcpy.gp.Reclassify\_sa(cult, "Value", "0 0;1 0;2 1;3 0", "{}2015\_Cultivated\_Layer.gdb/cultmask\_2015".format(
48. outdir), "DATA")
49. **print** "Completed Recode in: {}".format(datetime.datetime.now() - start\_t2)
50. ## Cast variable for agricultural mask
51. agmask = Raster("{}2015\_Cultivated\_Layer.gdb/cultmask\_2015".format(outdir))
52. **print** "Building Pyramids....."
53. arcpy.BuildPyramids\_management(agmask)
55. ## Clip CDL cultivated raster by county FIPS code
56. **if** **not** arcpy.Exists("{}2015\_Cult\_Cnty.gdb".format(outdir)):
57. arcpy.CreateFileGDB\_management(outdir, "2015\_Cult\_Cnty.gdb")
59. ## Set environments
60. arcpy.env.workspace = "{}2015\_Cult\_Cnty.gdb".format(outdir)
61. arcpy.env.scratchWorkspace = "{}2015\_Cult\_Cnty.gdb".format(outdir)
62. arcpy.env.extent = "MAXOF"
64. ## Make County Feature Layer
65. clipfc = "clip\_lyr"
66. arcpy.MakeFeatureLayer\_management(cnty, clipfc)
68. ## Loop through counties to clip CDL cultivated layer
69. **for** fips, shape **in** uniqueFips:
70. start\_t1 = datetime.datetime.now()
71. extent = shape.extent
72. extent = "{} {} {} {}".format(extent.XMin, extent.YMin, extent.XMax, extent.YMax)
73. arcpy.SelectLayerByAttribute\_management(clipfc, "NEW\_SELECTION", "{} = '{}'".format(fipscode, fips))
74. out\_file = "{}2015\_Cult\_Cnty.gdb/Cult\_2015\_{}".format(outdir, fips)
75. **print** "Clipping Cult\_2015\_{}....".format(fips)
76. arcpy.Clip\_management(agmask, extent, out\_file, clipfc, "2147483647", "ClippingGeometry", "NO\_MAINTAIN\_EXTENT")
77. **print** "Finished clip in: {}".format(datetime.datetime.now() - start\_t1)
78. **print** "Completed All County Clipping in: {}".format(datetime.datetime.now() - start\_t)

# Script 6: “6\_CDL\_Expand\_1015.py”

1. ## Script 6. This script uses the master table (updated in script 4) to expand the use footprint of counties where CDL
2. ## acreage is lower than Census acreage.
4. \_\_author\_\_ = 'Steven Lennartz'
6. ## Import Site Packages
7. **import** arcpy
8. **import** datetime
9. **from** arcpy **import** env
10. **from** arcpy.sa **import** \*
12. ## Set variables
13. ## Master table from script 4
14. acres = r"../Analysis/NASS2012\_CDL1015.gdb/NASS2012\_Genclass11"
15. ## Unique ID for counties, "FIPS code"
16. geoid = "FIPS"
17. ## Key table for general class crop names and codes
18. classcode = r"../Analysis/NASS2012\_CDL1015.gdb/NASS2012\_Genclass11\_codes"
19. ## Crop code field name. These codes are arbitrary, increments of 10, from 10 through 110.
20. classfield = "GenClassID"
22. ## In-workspace is the output directory from script 3
23. inws = r"../Cnty\_Clips/1015/"
24. ## Output directory
25. outws = r"../Expand/1015/"
26. ## The cultivated layer from the CDL website
27. maskws = r"../Cultivated\_Layer/2015\_Cultivated\_Layer/2015\_Cult\_Cnty.gdb/"
29. start\_t = datetime.datetime.now()
31. ## Check out the ArcGIS Spatial Analyst extension license
32. arcpy.CheckOutExtension("Spatial")
33. ## Environmental settings
34. arcpy.env.overwriteOutput = 1
35. arcpy.env.snapRaster = Raster("../Cultivated\_Layer/2015\_Cultivated\_Layer/2015\_Cultivated\_Layer.img")
36. arcpy.env.parallelProcessingFactor = "100%"
38. **print** "Begin processing at: {}".format(start\_t)
40. ## Get list of class codes
41. uniqueCrops = {row[0] **for** row **in** arcpy.da.SearchCursor(classcode, classfield)}
43. ## Define function to get a value's cell count
44. **def** get\_raster\_count(raster\_obj, value):
46. counts = [row.COUNT **for** row **in** arcpy.SearchCursor(raster\_obj) **if** row.VALUE == value]
47. **if** **not** counts:
48. **print** "No matching value"
49. **elif** len(counts) > 1:
50. **print** "Too many counts"
51. **else**:
52. **return** counts[0]
54. ## Expand looping
55. **for** crop **in** uniqueCrops:
56. crop = int(crop)
57. crop = str(crop)
59. ## check for existing geodatabase
60. **if** **not** arcpy.Exists("{}CDL{}x\_cnty.gdb".format(outws, crop)):
61. **print** "Creating CDL{}x\_cnty Geodatabase".format(crop)
62. arcpy.CreateFileGDB\_management(outws, "CDL{}x\_cnty.gdb".format(crop))
64. ## Return a unique set of values in the FIPS field for counties with higher NASS acreage than the CDL
65. gtacres = "gtlayer"
66. arcpy.MakeQueryTable\_management(acres, gtacres, "USE\_KEY\_FIELDS", "OBJECTID", "", "NASS\_{0} > CDL\_{0}".format(
67. crop))
68. uniqueValues = {(row[0], row[1]) **for** row **in** arcpy.da.SearchCursor(gtacres, (geoid, "NASS\_{}".format(crop)))}
70. ## Loop through counties needing expansion
71. **for** fips, nassacres **in** uniqueValues:
72. arcpy.env.workspace = "{}CDL{}x\_cnty.gdb/".format(outws, crop)
73. arcpy.env.scratchWorkspace = "{}CDL{}x\_cnty.gdb/".format(outws, crop)
74. fips = str(fips)
75. croptime1 = datetime.datetime.now()
76. **print** "expanding CDL{}\_{}".format(crop, fips)
77. ## Set mask from script 5
78. mask = Raster("{}Cult\_2015\_{}".format(maskws, fips))
79. infile = Raster("{0}CDL\_{1}\_cnty.gdb/CDL\_{1}\_{2}".format(inws, crop, fips))
80. ## Expand flagged county by 1 pixel
81. outfile = Expand(infile, 1, 1)
82. ## Using the environmental setting for mask does not work here. Use mask in map algebra instead.
83. outfile = outfile \* mask
84. arcpy.BuildRasterAttributeTable\_management(outfile, "Overwrite")
85. count = get\_raster\_count(outfile, 1)
86. ## Iteratively compare and expand county use footprint until area >= Census, or mask area buildout.
87. **if** count > 0:
88. cdlacres = float(count \* (900 / 4046.86))
89. **while** cdlacres < nassacres:
90. outfile = Expand(outfile, 1, 1)
91. outfile = outfile \* mask
92. arcpy.BuildRasterAttributeTable\_management(outfile, "Overwrite")
93. count = get\_raster\_count(outfile, 1)
94. lastacres = cdlacres
95. cdlacres = float(count \* (900 / 4046.86))
96. **if** **not** (lastacres - cdlacres):
97. **print** "No change in area"
98. **break**
99. outfile.save("{0}CDL{1}x\_cnty.gdb/CDL{1}x\_{2}".format(outws, crop, fips))
100. **print** "Building pyramids...."
101. arcpy.BuildPyramids\_management("{0}CDL{1}x\_cnty.gdb/CDL{1}x\_{2}".format(outws, crop, fips))
102. **print** "..{} expansion complete in {}".format(outfile, (datetime.datetime.now()-croptime1))
104. **print** "completed expansion in: {}".format(datetime.datetime.now() - start\_t)

# Script 7: “7\_CDL\_MosaicExpanded\_1015.py”

1. ## Script 7. Mosaic the county based expansions from script 6.
3. \_\_author\_\_ = 'Steven Lennartz'
5. ## Import Site Packages
6. **import** arcpy
7. **import** datetime
8. **from** arcpy **import** env
9. **from** arcpy.sa **import** \*
11. ## Set Variables
12. ## Key table for general class crop names and codes
13. classcode = r"../Analysis/NASS2012\_CDL1015.gdb/NASS2012\_Genclass11\_codes"
14. ## Crop code field name. These codes are arbitrary, increments of 10, from 10 through 110.
15. classfield = "GenClassID"
16. ## Master table from script 4
17. acres = r"../Analysis/NASS2012\_CDL1015.gdb/NASS2012\_Genclass11"
18. ## Unique ID for counties, "FIPS code"
19. geoid = "FIPS"
21. ## In-workspace is the output directory from script 3
22. inws = r"../Cnty\_Clips/1015/"
23. ## Output directory
24. outws = r"../Expand/1015/Mosaic/"
26. start\_t = datetime.datetime.now()
28. ## Environmental settings
29. arcpy.env.overwriteOutput = 1
30. arcpy.env.snapRaster = Raster("../Cultivated\_Layer/2015\_Cultivated\_Layer/2015\_Cultivated\_Layer.img")
31. arcpy.env.parallelProcessingFactor = "100%"
33. **print** "Begun processing at: {}".format(start\_t)
35. ## Check out the ArcGIS Spatial Analyst extension license
36. arcpy.CheckOutExtension("Spatial")
38. ## Get list of class codes
39. uniqueCrops = {row[0] **for** row **in** arcpy.da.SearchCursor(classcode, classfield)}
41. ## Get FIPS codes
42. uniqueFips = {row[0] **for** row **in** arcpy.da.SearchCursor(acres, geoid)}
44. ## check for existing geodatabase
45. **if** **not** arcpy.Exists("{}CDL1015x\_mosaic.gdb".format(outws)):
46. **print** "Creating CDL1015x\_mosaic.gdb"
47. arcpy.CreateFileGDB\_management(outws, "CDL1015x\_mosaic.gdb")
49. ## Set output geodatabase
50. outgdb = "{}CDL1015x\_mosaic.gdb".format(outws)
52. ## Mosaic loop by crop
53. **for** crop **in** uniqueCrops:
54. crop = int(crop)
55. crop = str(crop)
57. croptime1 = datetime.datetime.now()
58. **print** "starting {} mosaic at: {}".format(crop, croptime1)
60. ## Expanded geodatabase from script 6
61. xgdb = "../Expand/1015/CDL{}x\_cnty.gdb".format(crop)
62. arcpy.env.workspace = xgdb
63. ## Create Raster Catalog
64. cat = "{}/CDL{}x\_cat".format(outgdb, crop)
65. **if** arcpy.Exists(cat):
66. **print** "{} Exists".format(cat)
67. **if** **not** arcpy.Exists(cat):
68. **print** "Creating Raster Catalog CDL{}x\_cat".format(crop)
69. arcpy.CreateRasterCatalog\_management(outgdb, "CDL{}x\_cat".format(crop))
70. **print** "Copying Expanded Rasters to Catalog CDL{}x\_cat".format(crop)
71. arcpy.WorkspaceToRasterCatalog\_management(xgdb, cat)
73. ## Load non-expanded rasters into Raster Catalog
74. **print** "Loading non-expanded rasters into CDL{}x\_cat".format(crop)
75. **for** fips **in** uniqueFips:
76. timex2 = datetime.datetime.now()
77. cdlx = "{}/CDL{}x\_{}".format(xgdb, crop, fips)
78. **if** arcpy.Exists(cdlx):
79. **print** "CDL{}x\_{} exists".format(crop, fips)
80. **if** **not** arcpy.Exists(cdlx):
81. **if** arcpy.Exists("{}/CDL\_{}\_{}".format(cat, crop, fips)):
82. **print** "{}/CDL\_{}\_{} Exists".format(cat, crop, fips)
83. **if** **not** arcpy.Exists("{}/CDL\_{}\_{}".format(cat, crop, fips)):
84. arcpy.CopyRaster\_management("{0}CDL\_{1}\_cnty.gdb/CDL\_{1}\_{2}".format(inws, crop, fips), "{}/CDL\_{}\_{}".format(cat, crop, fips))
85. **print** "Added CDL\_{}\_{} to Catalog in {}".format(crop, fips, datetime.datetime.now() - timex2)
87. ## Mosaic Raster Catalog to new dataset
88. **print** "Raster CDL{}x\_cat to Raster Dataset at: {}".format(crop, datetime.datetime.now())
89. arcpy.RasterCatalogToRasterDataset\_management(cat, "{}/CDL\_1015\_{}x".format(outgdb, crop))
90. **print** "......Building pyramids"
91. arcpy.BuildPyramids\_management("{}/CDL\_1015\_{}x".format(outgdb, crop))
92. **print** "......Building attribute table"
93. arcpy.BuildRasterAttributeTable\_management("{}/CDL\_1015\_{}x".format(outgdb, crop), "NONE")
94. **print** "..CDL\_1015\_{}x mosaic complete in {}".format(crop, (datetime.datetime.now()-croptime1))
96. **print** "completed mosaic in: {}".format(datetime.datetime.now() - start\_t)

# Script 8: “8\_CDL\_FixHoles\_1015.py”

1. ## Script 8. Some county FIPS codes from Tiger County Boundaries are not reported in the Ag Census. Those counties do
2. ## not join in the expansion script 6 and result in NoData values. This script corrects for NoData.
4. \_\_author\_\_ = ' Steven Lennartz'
6. ## Import Site Packages
7. **import** arcpy
8. **import** datetime
9. **from** arcpy **import** env
10. **from** arcpy.sa **import** \*
12. ## Set Variables
13. ## In-workspace is the mosaic geodatabase from script 7
14. inws = r"../Expand/1015/Mosaic/CDL1015x\_mosaic.gdb/"
15. ## Out-workspace from script 2 supplies replacement values for NoData
16. patch = r"../CDL\_Recodes/CDL\_GenClass11\_1015.gdb/"
18. ## Environmental settings
19. arcpy.env.workspace = inws
20. arcpy.env.overwriteOutput = 1
21. arcpy.env.snapRaster = Raster("../Cultivated\_Layer/2015\_Cultivated\_Layer/2015\_Cultivated\_Layer.img")
22. arcpy.env.parallelProcessingFactor = "100%"
23. arcpy.env.scratchWorkspace = patch
25. ## Check out the ArcGIS Spatial Analyst extension license
26. arcpy.CheckOutExtension("Spatial")
28. start\_t = datetime.datetime.now()
29. **print** "Begun processing at: {}".format(start\_t)
31. ## check for existing geodatabase
32. **if** **not** arcpy.Exists("../Expand/1015/Mosaic/Final/CDL1015x\_mosaic2.gdb"):
33. **print** "Creating CDL1015x\_mosaic2.gdb"
34. arcpy.CreateFileGDB\_management("../Expand/1015/Mosaic/Final/", "CDL1015x\_mosaic2.gdb")
35. outws = r"../Expand/1015/Mosaic/Final/CDL1015x\_mosaic2.gdb/"
37. ## List rasters in workspace
38. rasterList = arcpy.ListRasters()
39. ## Loop through rasters to correct NoData
40. **for** raster **in** rasterList:
41. start\_t1 = datetime.datetime.now()
42. **print** "Processing {} at {}".format(raster, datetime.datetime.now())
43. inras = Raster(raster)
44. desc = arcpy.Describe(inras)
45. name = desc.name
46. val = name.split('\_')[2]
47. val = val[:-1]
48. true = Raster("{}CDL\_1015\_{}".format(patch, val))
49. out = Con(IsNull(inras), true, inras)
50. out.save("{}CDL\_1015\_{}x2".format(outws, val))
51. **print** "Building Pyramids....."
52. arcpy.BuildPyramids\_management("{}CDL\_1015\_{}x2".format(outws, val))
53. **print** "Completed NoData holes for Class {} in: {}".format(val, datetime.datetime.now() - start\_t1)
54. **print** "Completed NoData holes for all classes in: {}".format(datetime.datetime.now() - start\_t)

# Table 4: “cdl\_recode\_new.txt”

| **FROM** | **TO** | **OUT** | **MAPPING** |
| --- | --- | --- | --- |
| 0 | 0 | 0 | ValueToValue |
| 1 | 1 | 10 | ValueToValue |
| 2 | 2 | 20 | ValueToValue |
| 3 | 3 | 30 | ValueToValue |
| 4 | 4 | 80 | ValueToValue |
| 5 | 5 | 40 | ValueToValue |
| 6 | 6 | 90 | ValueToValue |
| 7 | 7 | 7 | ValueToValue |
| 8 | 8 | 8 | ValueToValue |
| 9 | 9 | 9 | ValueToValue |
| 10 | 10 | 90 | ValueToValue |
| 11 | 11 | 90 | ValueToValue |
| 12 | 12 | 60 | ValueToValue |
| 13 | 13 | 60 | ValueToValue |
| 14 | 14 | 60 | ValueToValue |
| 15 | 15 | 15 | ValueToValue |
| 16 | 16 | 16 | ValueToValue |
| 17 | 17 | 17 | ValueToValue |
| 18 | 18 | 18 | ValueToValue |
| 19 | 19 | 19 | ValueToValue |
| 20 | 20 | 20 | ValueToValue |
| 21 | 21 | 80 | ValueToValue |
| 22 | 22 | 50 | ValueToValue |
| 23 | 23 | 50 | ValueToValue |
| 24 | 24 | 50 | ValueToValue |
| 25 | 25 | 80 | ValueToValue |
| 26 | 26 | 45 | ValueToValue |
| 27 | 27 | 80 | ValueToValue |
| 28 | 28 | 80 | ValueToValue |
| 29 | 29 | 80 | ValueToValue |
| 30 | 30 | 80 | ValueToValue |
| 31 | 31 | 80 | ValueToValue |
| 32 | 32 | 80 | ValueToValue |
| 33 | 33 | 80 | ValueToValue |
| 34 | 34 | 80 | ValueToValue |
| 35 | 35 | 60 | ValueToValue |
| 36 | 36 | 110 | ValueToValue |
| 37 | 37 | 110 | ValueToValue |
| 38 | 38 | 80 | ValueToValue |
| 39 | 39 | 80 | ValueToValue |
| 40 | 40 | 40 | ValueToValue |
| 41 | 41 | 90 | ValueToValue |
| 42 | 42 | 60 | ValueToValue |
| 43 | 43 | 60 | ValueToValue |
| 44 | 44 | 100 | ValueToValue |
| 45 | 45 | 80 | ValueToValue |
| 46 | 46 | 60 | ValueToValue |
| 47 | 47 | 60 | ValueToValue |
| 48 | 48 | 60 | ValueToValue |
| 49 | 49 | 60 | ValueToValue |
| 50 | 50 | 60 | ValueToValue |
| 51 | 51 | 60 | ValueToValue |
| 52 | 52 | 60 | ValueToValue |
| 53 | 53 | 60 | ValueToValue |
| 54 | 54 | 60 | ValueToValue |
| 55 | 55 | 61 | ValueToValue |
| 56 | 56 | 90 | ValueToValue |
| 57 | 57 | 60 | ValueToValue |
| 58 | 58 | 100 | ValueToValue |
| 59 | 59 | 100 | ValueToValue |
| 60 | 60 | 110 | ValueToValue |
| 61 | 61 | 100 | ValueToValue |
| 62 | 62 | 110 | ValueToValue |
| 63 | 63 | 140 | ValueToValue |
| 64 | 64 | 160 | ValueToValue |
| 65 | 65 | 200 | ValueToValue |
| 66 | 66 | 70 | ValueToValue |
| 67 | 67 | 70 | ValueToValue |
| 68 | 68 | 70 | ValueToValue |
| 69 | 69 | 70 | ValueToValue |
| 70 | 70 | 75 | ValueToValue |
| 71 | 71 | 70 | ValueToValue |
| 72 | 72 | 70 | ValueToValue |
| 73 | 73 | 73 | ValueToValue |
| 74 | 74 | 70 | ValueToValue |
| 75 | 75 | 70 | ValueToValue |
| 76 | 76 | 70 | ValueToValue |
| 77 | 77 | 70 | ValueToValue |
| 78 | 78 | 78 | ValueToValue |
| 79 | 79 | 79 | ValueToValue |
| 80 | 80 | 80 | ValueToValue |
| 81 | 81 | 200 | ValueToValue |
| 82 | 82 | 123 | ValueToValue |
| 83 | 83 | 180 | ValueToValue |
| 84 | 84 | 84 | ValueToValue |
| 85 | 85 | 85 | ValueToValue |
| 86 | 86 | 86 | ValueToValue |
| 87 | 87 | 195 | ValueToValue |
| 88 | 88 | 200 | ValueToValue |
| 89 | 89 | 89 | ValueToValue |
| 90 | 90 | 90 | ValueToValue |
| 91 | 91 | 91 | ValueToValue |
| 92 | 92 | 100 | ValueToValue |
| 93 | 93 | 93 | ValueToValue |
| 94 | 94 | 94 | ValueToValue |
| 95 | 95 | 95 | ValueToValue |
| 96 | 96 | 96 | ValueToValue |
| 97 | 97 | 97 | ValueToValue |
| 98 | 98 | 98 | ValueToValue |
| 99 | 99 | 99 | ValueToValue |
| 100 | 100 | 100 | ValueToValue |
| 101 | 101 | 101 | ValueToValue |
| 102 | 102 | 102 | ValueToValue |
| 103 | 103 | 103 | ValueToValue |
| 104 | 104 | 104 | ValueToValue |
| 105 | 105 | 105 | ValueToValue |
| 106 | 106 | 106 | ValueToValue |
| 107 | 107 | 107 | ValueToValue |
| 108 | 108 | 108 | ValueToValue |
| 109 | 109 | 109 | ValueToValue |
| 110 | 110 | 110 | ValueToValue |
| 111 | 111 | 180 | ValueToValue |
| 112 | 112 | 200 | ValueToValue |
| 113 | 113 | 113 | ValueToValue |
| 114 | 114 | 114 | ValueToValue |
| 115 | 115 | 115 | ValueToValue |
| 116 | 116 | 116 | ValueToValue |
| 117 | 117 | 117 | ValueToValue |
| 118 | 118 | 118 | ValueToValue |
| 119 | 119 | 119 | ValueToValue |
| 120 | 120 | 120 | ValueToValue |
| 121 | 121 | 121 | ValueToValue |
| 122 | 122 | 122 | ValueToValue |
| 123 | 123 | 123 | ValueToValue |
| 124 | 124 | 124 | ValueToValue |
| 125 | 125 | 125 | ValueToValue |
| 126 | 126 | 126 | ValueToValue |
| 127 | 127 | 127 | ValueToValue |
| 128 | 128 | 128 | ValueToValue |
| 129 | 129 | 129 | ValueToValue |
| 130 | 130 | 130 | ValueToValue |
| 131 | 131 | 200 | ValueToValue |
| 132 | 132 | 132 | ValueToValue |
| 133 | 133 | 133 | ValueToValue |
| 134 | 134 | 134 | ValueToValue |
| 135 | 135 | 135 | ValueToValue |
| 136 | 136 | 136 | ValueToValue |
| 137 | 137 | 137 | ValueToValue |
| 138 | 138 | 138 | ValueToValue |
| 139 | 139 | 139 | ValueToValue |
| 140 | 140 | 140 | ValueToValue |
| 141 | 141 | 140 | ValueToValue |
| 142 | 142 | 140 | ValueToValue |
| 143 | 143 | 140 | ValueToValue |
| 144 | 144 | 144 | ValueToValue |
| 145 | 145 | 145 | ValueToValue |
| 146 | 146 | 146 | ValueToValue |
| 147 | 147 | 147 | ValueToValue |
| 148 | 148 | 148 | ValueToValue |
| 149 | 149 | 149 | ValueToValue |
| 150 | 150 | 110 | ValueToValue |
| 151 | 151 | 151 | ValueToValue |
| 152 | 152 | 160 | ValueToValue |
| 153 | 153 | 153 | ValueToValue |
| 154 | 154 | 154 | ValueToValue |
| 155 | 155 | 155 | ValueToValue |
| 156 | 156 | 156 | ValueToValue |
| 157 | 157 | 157 | ValueToValue |
| 158 | 158 | 158 | ValueToValue |
| 159 | 159 | 159 | ValueToValue |
| 160 | 160 | 160 | ValueToValue |
| 161 | 161 | 161 | ValueToValue |
| 162 | 162 | 162 | ValueToValue |
| 163 | 163 | 163 | ValueToValue |
| 164 | 164 | 164 | ValueToValue |
| 165 | 165 | 165 | ValueToValue |
| 166 | 166 | 166 | ValueToValue |
| 167 | 167 | 167 | ValueToValue |
| 168 | 168 | 168 | ValueToValue |
| 169 | 169 | 169 | ValueToValue |
| 170 | 170 | 170 | ValueToValue |
| 171 | 171 | 110 | ValueToValue |
| 172 | 172 | 172 | ValueToValue |
| 173 | 173 | 173 | ValueToValue |
| 174 | 174 | 174 | ValueToValue |
| 175 | 175 | 175 | ValueToValue |
| 176 | 176 | 110 | ValueToValue |
| 177 | 177 | 177 | ValueToValue |
| 178 | 178 | 178 | ValueToValue |
| 179 | 179 | 179 | ValueToValue |
| 180 | 180 | 180 | ValueToValue |
| 181 | 181 | 110 | ValueToValue |
| 182 | 182 | 182 | ValueToValue |
| 183 | 183 | 183 | ValueToValue |
| 184 | 184 | 184 | ValueToValue |
| 185 | 185 | 185 | ValueToValue |
| 186 | 186 | 186 | ValueToValue |
| 187 | 187 | 187 | ValueToValue |
| 188 | 188 | 188 | ValueToValue |
| 189 | 189 | 189 | ValueToValue |
| 190 | 190 | 190 | ValueToValue |
| 191 | 191 | 191 | ValueToValue |
| 192 | 192 | 192 | ValueToValue |
| 193 | 193 | 193 | ValueToValue |
| 194 | 194 | 194 | ValueToValue |
| 195 | 195 | 195 | ValueToValue |
| 196 | 196 | 196 | ValueToValue |
| 197 | 197 | 197 | ValueToValue |
| 198 | 198 | 198 | ValueToValue |
| 199 | 199 | 199 | ValueToValue |
| 200 | 200 | 200 | ValueToValue |
| 201 | 201 | 201 | ValueToValue |
| 202 | 202 | 202 | ValueToValue |
| 203 | 203 | 203 | ValueToValue |
| 204 | 204 | 70 | ValueToValue |
| 205 | 205 | 80 | ValueToValue |
| 206 | 206 | 60 | ValueToValue |
| 207 | 207 | 60 | ValueToValue |
| 208 | 208 | 60 | ValueToValue |
| 209 | 209 | 60 | ValueToValue |
| 210 | 210 | 70 | ValueToValue |
| 211 | 211 | 70 | ValueToValue |
| 212 | 212 | 70 | ValueToValue |
| 213 | 213 | 60 | ValueToValue |
| 214 | 214 | 60 | ValueToValue |
| 215 | 215 | 215 | ValueToValue |
| 216 | 216 | 60 | ValueToValue |
| 217 | 217 | 70 | ValueToValue |
| 218 | 218 | 70 | ValueToValue |
| 219 | 219 | 60 | ValueToValue |
| 220 | 220 | 70 | ValueToValue |
| 221 | 221 | 61 | ValueToValue |
| 222 | 222 | 60 | ValueToValue |
| 223 | 223 | 70 | ValueToValue |
| 224 | 224 | 110 | ValueToValue |
| 225 | 225 | 15 | ValueToValue |
| 226 | 226 | 18 | ValueToValue |
| 227 | 227 | 60 | ValueToValue |
| 228 | 228 | 228 | ValueToValue |
| 229 | 229 | 60 | ValueToValue |
| 230 | 230 | 56 | ValueToValue |
| 231 | 231 | 60 | ValueToValue |
| 232 | 232 | 26 | ValueToValue |
| 233 | 233 | 68 | ValueToValue |
| 234 | 234 | 58 | ValueToValue |
| 235 | 235 | 80 | ValueToValue |
| 236 | 236 | 58 | ValueToValue |
| 237 | 237 | 18 | ValueToValue |
| 238 | 238 | 25 | ValueToValue |
| 239 | 239 | 42 | ValueToValue |
| 240 | 240 | 48 | ValueToValue |
| 241 | 241 | 14 | ValueToValue |
| 242 | 242 | 61 | ValueToValue |
| 243 | 243 | 60 | ValueToValue |
| 244 | 244 | 60 | ValueToValue |
| 245 | 245 | 60 | ValueToValue |
| 246 | 246 | 60 | ValueToValue |
| 247 | 247 | 60 | ValueToValue |
| 248 | 248 | 60 | ValueToValue |
| 249 | 249 | 60 | ValueToValue |
| 250 | 250 | 61 | ValueToValue |
| 251 | 251 | 251 | ValueToValue |
| 252 | 252 | 252 | ValueToValue |
| 253 | 253 | 253 | ValueToValue |
| 254 | 254 | 48 | ValueToValue |
| 255 | 255 | 255 | ValueToValue |

# Table 5: “EPA\_CDL\_Recode\_Key.txt”

|  |  |
| --- | --- |
| **Value** | **CDL\_General\_Class** |
| 10 | Corn |
| 14 | Corn/soybeans |
| 15 | Corn/wheat |
| 18 | Corn/grains |
| 20 | Cotton |
| 25 | Cotton/wheat |
| 26 | Cotton/vegetables |
| 30 | Rice |
| 40 | Soybeans |
| 42 | Soybeans/cotton |
| 45 | Soybeans/wheat |
| 48 | Soybeans/grains |
| 50 | Wheat |
| 56 | Wheat/vegetables |
| 58 | Wheat/grains |
| 60 | Vegetables and ground fruit |
| 61 | (ground fruit) |
| 68 | Vegetables/grains |
| 70 | Orchards and grapes |
| 75 | Other trees |
| 80 | Other grains |
| 90 | Other row crops |
| 100 | Other crops |
| 110 | Pasture/hay/forage |
| 121 | Developed - open |
| 122 | Developed - low |
| 123 | Developed - med |
| 124 | Developed - high |
| 140 | Forest |
| 160 | Shrubland |
| 180 | Water |
| 190 | Wetlands - woods |
| 195 | Wetlands - herbaceous |
| 200 | Miscellaneous land |

**References**

* **Bonneville Power Administration Right of Way (BPA ROW)**
  + Bonneville Power Administration GIS, 2015, <https://bpagis.maps.arcgis.com/home/>
* **Bureau of Land Management (BLM) Grazing Allotments**
  + BLM GIS, Grazing Allotment Boundaries, 20140112 <https://catalog.data.gov/dataset/grazing-allotment-boundaries>
* **Dun & Bradstreet (D&B)**
  + Dun & Bradstreet, Agriculture, US, 2012, Dun & Bradstreet, SEGS, Short Hills, NJ, 2013/04/08, <http://igeo.epa.gov/data/Restricted/OEI/Agriculture/DunAndBradstreet_Agriculture.zip>
* **ESRI StreetMap North America Railroads**
  + ESRI, StreetMap North America, Redlands, CA 20100531
* **Hawaii State Department of Agriculture and the Hawaii Statewide GIS Program**
  + Agricultural Land Use Maps (ALUM), <http://planning.hawaii.gov/gis/download-gis-data/> , 20151104
* **NAVTEQ Street Data**
  + NAVTEQ 2013 Streets, Chicago, IL, 20131001
* **National Land Cover Dataset (NLCD)**
  + Homer, C.G., Dewitz, J.A., Yang, L., Jin, S., Danielson, P., Xian, G., Coulston, J., Herold, N.D., Wickham, J.D., and Megown, K., 2015, Completion of the 2011 National Land Cover Database for the conterminous United States-Representing a decade of land cover change information. Photogrammetric Engineering and Remote Sensing, v. 81, no. 5, p. 345-354
* **National Oceanic and Atmospheric Administration (NOAA) Coastal Change Analysis Program (CCAP)** 
  + National Oceanic and Atmospheric Administration, Coastal Services Center. 1995-present. The Coastal Change Analysis Program (C-CAP) Regional Land Cover. Charleston, SC: NOAA Coastal Services Center. Accessed at [www.csc.noaa.gov/digitalcoast/data/ccapregional/index.html](http://www.csc.noaa.gov/digitalcoast/data/ccapregional/index.html).
* **Puerto Rico Census of Agriculture**
  + Junta de Planificación, Censo Agricola 2002 <http://gis.jp.pr.gov/mipr/> , 20151001
* **United States Census Bureau’s Topologically Integrated Geographic Encoding and Referencing database (TIGER)**
  + 2015 TIGER/Line Shapefiles (machine readable data files) / prepared by the U.S. Census Bureau, 2015, <http://www.census.gov/geo/maps-data/data/tiger-geodatabases.html>
* **United States Department of Agriculture Cropland Data Layer (CDL)**
  + United States Department of Agriculture (USDA), National Agricultural Statistics Service (NASS), Research and Development Division (RDD), Geospatial Information Branch (GIB), Spatial Analysis Research Section (SARS), Cropland Data Layer for the United States, <https://www.nass.usda.gov/Research_and_Science/Cropland/SARS1a.php>
* **United States Forest Service Administrative Boundaries**
  + USDA Forest Service, Administrative Forest Boundaries, “S\_USA.AdministrativeForest”, 20151027, <http://data.fs.usda.gov/geodata/edw/datasets.php>
* **United States Forest Service Grazing Allotments**
  + USFS Range Allotment Boundaries, NationalAllotmentFeatureClassAlbers, Rangeland Management Unit, 20140916
* **United States Forest Service Slash Pine Presence**
  + Ellenwood\_ James R.; Krist\_ Frank J.\_ Jr.; Romero\_ Sheryl A. 2015. National Individual Tree Species Atlas. FHTET-15–01. Fort Collins\_ Colorado: U.S. Department of Agriculture\_ Forest Service\_ Forest Health Technology Enterprise Team.
* **United States Geological Survey GAP Land Cover Data (USGS GAP)**
  + US Geological Survey, Gap Analysis Program (GAP). May 2011. National Land Cover, Version 2
* **United States Geological Survey GAP Protected Areas Database (USGS GAP PAD-US)**
  + US Geological Survey, Gap Analysis Program (GAP). November 2012. Protected Areas Database of the United States (PADUS), version 1.3 Combined Feature Class.
* **United States Geological Survey LandFire Existing Vegetation Type (USGS LandFire EVT)**
  + LANDFIRE, 2012, Existing Vegetation Type Layer, LANDFIRE 1.3.0, U.S. Department of the Interior, Geological Survey. Accessed 15 July 2015 at <http://www.landfire.gov/lf_mosaics.php>
* **United States Geological Survey LandFire Public Events GeoDatabase (USGS LandFire Events)**
  + LANDFIRE, 2012, Public Events GeoDatabase, LANDFIRE 1.3.0, U.S. Department of the Interior, Geological Survey. Accessed 15 July 2015 at <http://www.landfire.gov/lf_mosaics.php>