Agricultural Zone Climate Summary

Matthew Perry, Oct 21 2014

Ecotrust has developed a spatial dataset of agricultural zones for the western US. These areas form relatively contiguous regions of agricultural activity, including climatic conditions.

In the following analysis, we'll summarize the following climatic variables by ag zone:

- Minimum Temperature, December
- Maximum Temperature, August
- Growing Season
- Mean monthly winter precipitation
- Mean monthly summer precipitation

The climatic summaries will provide the min, mean, median, max, standard deviation and range of each of these five variables.

Load required python libraries

In [1]:
```
%matplotlib inline
%pylab inline
pylab.rcParams['figure.figsize'] = (10.0, 8.0)

import geopandas as gpd
import pandas as pd

import rasterio
import os
from rasterstats import zonal_stats

gpd.version.version
```

```
Populating the interactive namespace from numpy and matplotlib
```

Out[1]:
```
'0.1.0.dev-cb62e9f'
```

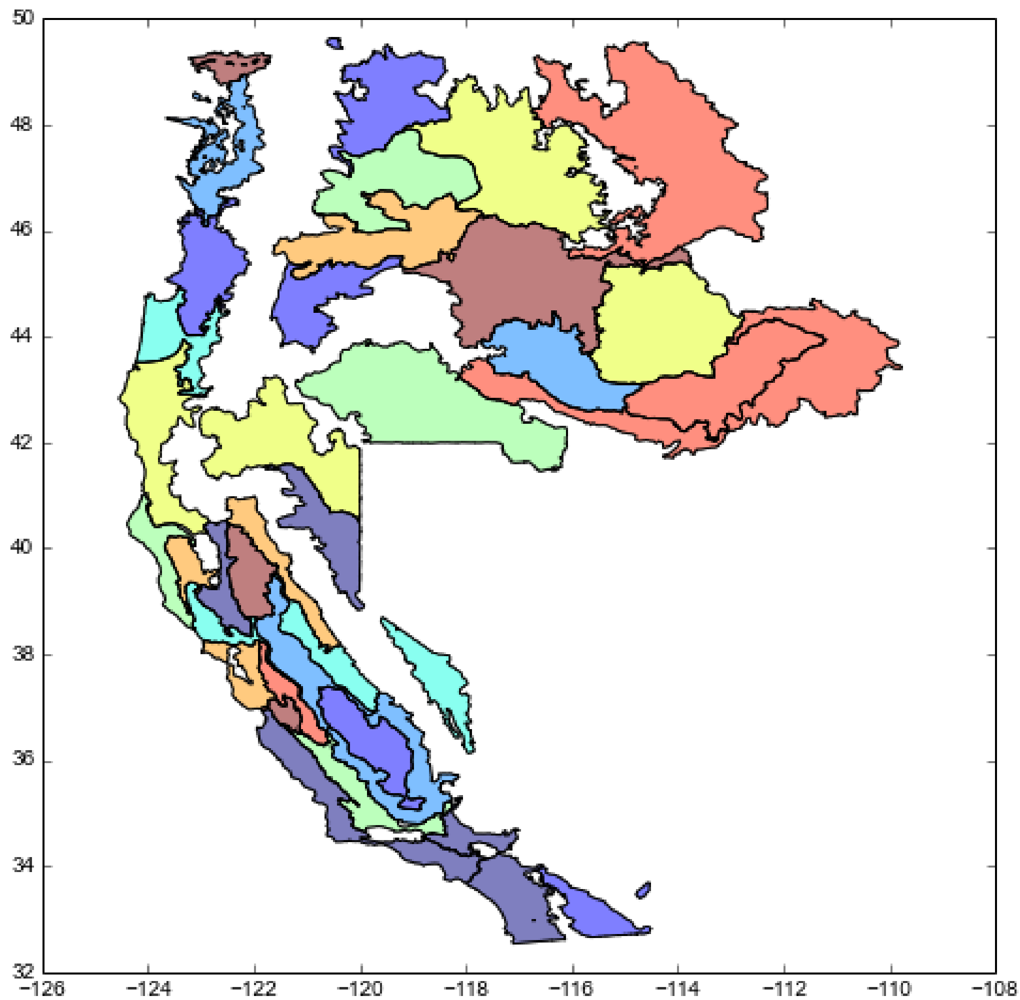
Prepare zone data

The source data is a geojson file with Polygon geometries in epsg:4326 (lat/long wgs84). We'll need to load the dataset into memory as a geopandas dataframe ....

In [2]:
```
orig_zones = gpd.GeoDataFrame.from_file('data/food_zones.geojson')
orig_crs = orig_zones.crs
orig_zones.plot()
```

```
/usr/local/lib/python2.7/dist-packages/pkg_resources.py:1045: UserWarn
ing: /home/mperry/.python-eggs is writable by group/others and vulnera
ble to attack when used with get_resource_filename. Consider a more se
cure location (set with .set_extraction_path or the PYTHON_EGG_CACHE e
nvironment variable).
  warnings.warn(msg, UserWarning)
```

Out[2]: <matplotlib.axes.AxesSubplot at 0x7f68e93b6b10>



In [3]:
```
"Number of columns = {}".format(len(orig_zones.columns))
```

Out[3]: 'Number of columns = 617'

Let's prune the columns down a bit, retaining only the zone_id and the geometry

In [4]:
```
zones = orig_zones[['zone_id', 'geometry']]
zones.head()
```

Out[4]:

	zone_id	geometry
0	Central Coast	(POLYGON ((-117.9893266136572834 33.6663290530...
1	Central Oregon	(POLYGON ((-120.8778057195829803 43.6927970362...
2	Central Valley	(POLYGON ((-120.8050860911718729 36.7601758742...
3	Central Valley Foothills	(POLYGON ((-121.1485200055207514 37.2051306983...
4	Coastal Valleys and Foothills	(POLYGON ((-118.7981542807468713 34.4892457104...

Reproject to the coordinate reference system of the climate data

Now we'll load up some climate data and reproject the zones to the same coordinate reference system.

In [5]:
```
with rasterio.open('/home/mperry/projects/Moore_food/_aez_data/trainin
g/tmin12c/hdr.adf') as ds:
    band = ds.read_band(1)
    meta = ds.meta

raster_crs = meta['crs']
raster_crs
```

Out[5]:
```
{u'ellps': u'WGS84',
 u'lat_0': -100,
 u'lon_0': 6370997,
 u'no_defs': True,
 u'proj': u'laea',
 u'towgs84': u'0,0,0,0,0,0,0',
 u'units': u'm',
 u'x_0': 45,
 u'y_0': 0}
```

Here we see the first problem: **The crs from the raster data is malformed**. This is the ESRI projection named `Sphere ARC_INFO_Lamber_Azimuthal_Equal_Area`. Note that this version has `lon_0` as not actually a longitude and the false easting (`x_0`) is not in meters. I'm not sure where the fault lies here (either the ESRI ArcInfo Grid has bad projection info *or* the GDAL driver misreads it) but either way it needs to be addressed.

This is the true coordinate reference system according to arcmap:

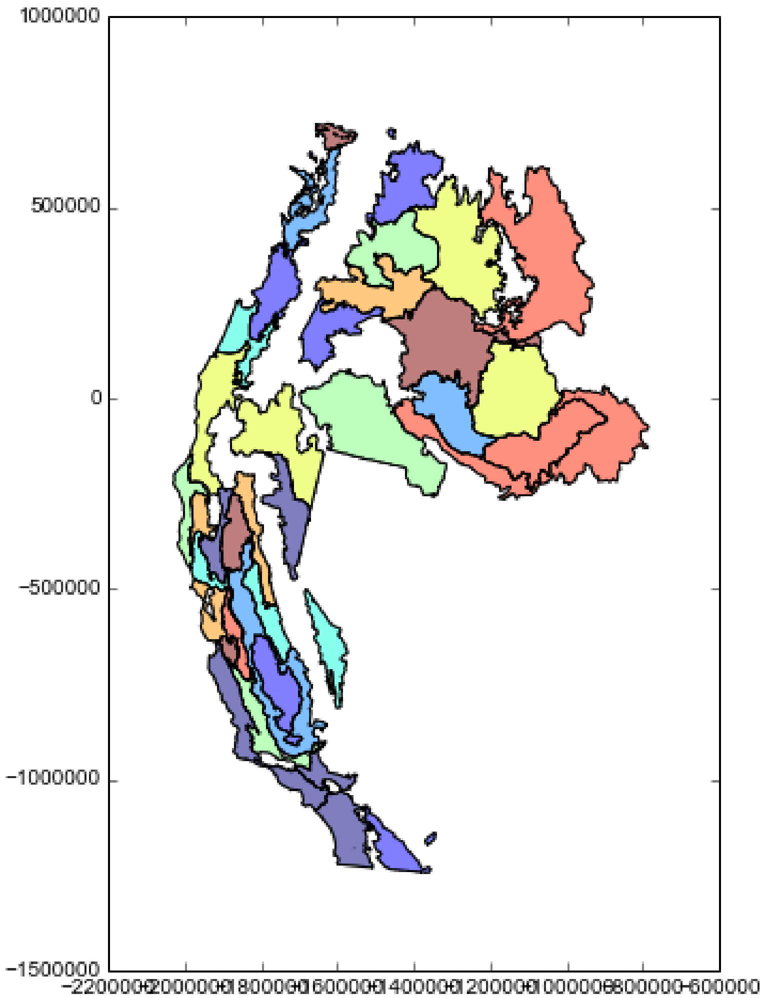
```
Projected Coordinate System:    Sphere_ARC_INFO_Lambert_Azimuthal_Equal_Area
Projection:    Lambert_Azimuthal_Equal_Area
False_Easting:    0.00000000
False_Northing:    0.00000000
Central_Meridian:    -100.00000000
Latitude_Of_Origin:    45.00000000
Linear Unit:     Meter

Geographic Coordinate System:    GCS_Sphere_ARC_INFO
Datum:     D_Sphere_ARC_INFO
Prime Meridian:     Greenwich
Angular Unit:     Degree
```

The `GCS_Sphere_ARC_INFO` spheroid uses a radius of `6370997 meters`; let's recreate that as a crs dictionary:

In [6]:
```
true_crs = {
    'a': 6370997,
    'b': 6370997,
    'lat_0': 45.0,
    'lon_0': -100.0,
    'no_defs': True,
    'proj': 'laea',
    'units': 'm'
}
```

In [7]:
```
zones_proj = zones.to_crs(crs=true_crs)
zones_proj.plot()
```

Out[7]: <matplotlib.axes.AxesSubplot at 0x7f68e928bd90>



Load climate rasters

In [8]:
```
climate_variables = "tmin12c tmax8c pmean_wntrc pmean_sumrc grwsnc".sp
lit()

dirs = {
  'rcpNA_2000s': '/home/mperry/projects/Moore_food/_aez_data/training/
',
  'rcp45_2030s': '/home/mperry/projects/Moore_food/_aez_data/RCP45/203
0s/',
  'rcp45_2050s': '/home/mperry/projects/Moore_food/_aez_data/RCP45/205
0s/',
  'rcp45_2070s': '/home/mperry/projects/Moore_food/_aez_data/RCP45/207
0s/',
  'rcp45_2080s': '/home/mperry/projects/Moore_food/_aez_data/RCP45/208
0s/',
  'rcp85_2030s': '/home/mperry/projects/Moore_food/_aez_data/RCP85/203
0s/',
  'rcp85_2050s': '/home/mperry/projects/Moore_food/_aez_data/RCP85/205
0s/',
  'rcp85_2070s': '/home/mperry/projects/Moore_food/_aez_data/RCP85/207
0s/',
  'rcp85_2080s': '/home/mperry/projects/Moore_food/_aez_data/RCP85/208
0s/',
}

rasters = {}
for n, d in dirs.items():
    for r in climate_variables:
        raster = os.path.join(d, r, "hdr.adf")
        name = n + "_" + r
        rasters[name] = raster
```

For each *RCP, year, and climate variable*, we now have an item in the `rasters` dictionary. The naming convention for the key is:

```
<rcp>_<year>_<climate variable>
```

Loop through rasters and perform zonal stats

First demonstrate with a single raster (The current max summer temperature)

In [9]:
```
metrics = "min max mean median std range".split()
stats = zonal_stats(zones_proj, rasters['rcpNA_2000s_tmax8c'], stats=m
etrics)
stats[1]
```

Out[9]:
```
{'__fid__': 1,
 'max': 325.0,
 'mean': 279.3255426739023,
 'median': 279.0,
 'min': 206.0,
 'range': 119.0,
 'std': 13.583096912000343}
```

Now we can take this list of dicts and convert into a dataframe to later rejoin with the `zones` geodataframe

In [10]:
```
df = pd.DataFrame(stats)
df.head()
```

Out[10]:

	__fid__	max	mean	median	min	range	std
0	0	344	275.739149	279	189	155	31.759313
1	1	325	279.325543	279	206	119	13.583097
2	2	361	337.498115	339	316	45	8.630541
3	3	363	341.467954	341	306	57	8.565192
4	4	366	316.390621	321	223	143	25.934209

In [11]:
```
raster = 'rcpNA_2000s_tmax8c'
new_colnames = ["{}_{}".format(raster, metric) for metric in metrics]
df2 = df.rename(columns=dict(zip(metrics, new_colnames)))
df2.head()
```

Out[11]:

	__fid__	rcpNA_2000s_tmax8c_max	rcpNA_2000s_tmax8c_mean	rcpNA_2000s_tmax
0	0	344	275.739149	279
1	1	325	279.325543	279
2	2	361	337.498115	339
3	3	363	341.467954	341
4	4	366	316.390621	321

In [12]:
```
df3 = zones.join(df2)
df3.head()
```

Out[12]:

	zone_id	geometry	__fid__	rcpNA_2000s_tmax8c_max	rcpNA_2000s_
0	Central Coast	(POLYGON ((-117.9893266136572834 33.6663290530...	0	344	275.739149
1	Central Oregon	(POLYGON ((-120.8778057195829803 43.6927970362...	1	325	279.325543
2	Central Valley	(POLYGON ((-120.8050860911718729 36.7601758742...	2	361	337.498115
3	Central Valley Foothills	(POLYGON ((-121.1485200055207514 37.2051306983...	3	363	341.467954
4	Coastal Valleys and Foothills	(POLYGON ((-118.7981542807468713 34.4892457104...	4	366	316.390621

Now do it in a loop for all rasters

In [13]:
```
# make a copy since it's updated in place
working_zones = zones.copy()

for raster, path in rasters.items():
    print raster
    stats = zonal_stats(zones_proj, path, stats=metrics)
    new_colnames = ["{}_{}".format(raster, metric) for metric in metri
cs]

    df = pd.DataFrame(stats)
    df2 = df.rename(columns=dict(zip(metrics, new_colnames)))
    df3 = df2.drop('__fid__', axis=1)
    working_zones = working_zones.join(df3)  # append to working copy
```

rcp45_2030s_pmean_sumrc
rcp45_2050s_pmean_sumrc
rcp85_2030s_tmin12c
rcp45_2030s_grwsnc
rcp85_2080s_pmean_wntrc
rcp45_2080s_pmean_wntrc
rcp45_2070s_tmax8c
rcp45_2050s_tmin12c
rcp85_2030s_tmax8c
rcp85_2030s_grwsnc
rcpNA_2000s_pmean_wntrc
rcp85_2050s_pmean_wntrc
rcp85_2030s_pmean_sumrc
rcpNA_2000s_tmin12c
rcp85_2070s_pmean_wntrc
rcp85_2050s_grwsnc
rcp85_2050s_tmax8c
rcp85_2080s_pmean_sumrc
rcp45_2050s_grwsnc
rcp85_2080s_grwsnc
rcp45_2080s_grwsnc
rcp45_2080s_tmax8c
rcp85_2050s_pmean_sumrc
rcp85_2080s_tmin12c
rcp85_2080s_tmax8c
rcp85_2070s_pmean_sumrc
rcp85_2070s_grwsnc
rcp45_2070s_pmean_wntrc
rcp85_2070s_tmax8c
rcp45_2070s_grwsnc
rcp45_2070s_tmin12c
rcp45_2080s_tmin12c
rcpNA_2000s_pmean_sumrc
rcp85_2050s_tmin12c
rcpNA_2000s_tmax8c
rcp45_2030s_tmax8c
rcp45_2030s_pmean_wntrc
rcp85_2070s_tmin12c
rcp45_2080s_pmean_sumrc
rcpNA_2000s_grwsnc
rcp45_2030s_tmin12c
rcp45_2050s_tmax8c
rcp45_2050s_pmean_wntrc
rcp45_2070s_pmean_sumrc
rcp85_2030s_pmean_wntrc

In [14]:
```
working_zones.head()
```

Out[14]:

	zone_id	geometry	rcp45_2030s_pmean_sumrc_max	rcp45_2030s_pn
0	Central Coast	(POLYGON ((-117.9893266136572834 33.6663290530...	3	0.802643
1	Central Oregon	(POLYGON ((-120.8778057195829803 43.6927970362...	41	18.439689
2	Central Valley	(POLYGON ((-120.8050860911718729 36.7601758742...	4	0.901704
3	Central Valley Foothills	(POLYGON ((-121.1485200055207514 37.2051306983...	8	1.926873
4	Coastal Valleys and Foothills	(POLYGON ((-118.7981542807468713 34.4892457104...	6	1.205120

5 rows × 272 columns

Finally (because the geodf.join method returns as standard DataFrame, not a GeoDataFrame due to a geopandas bug) we have to explictly convert back to a geodataframe.

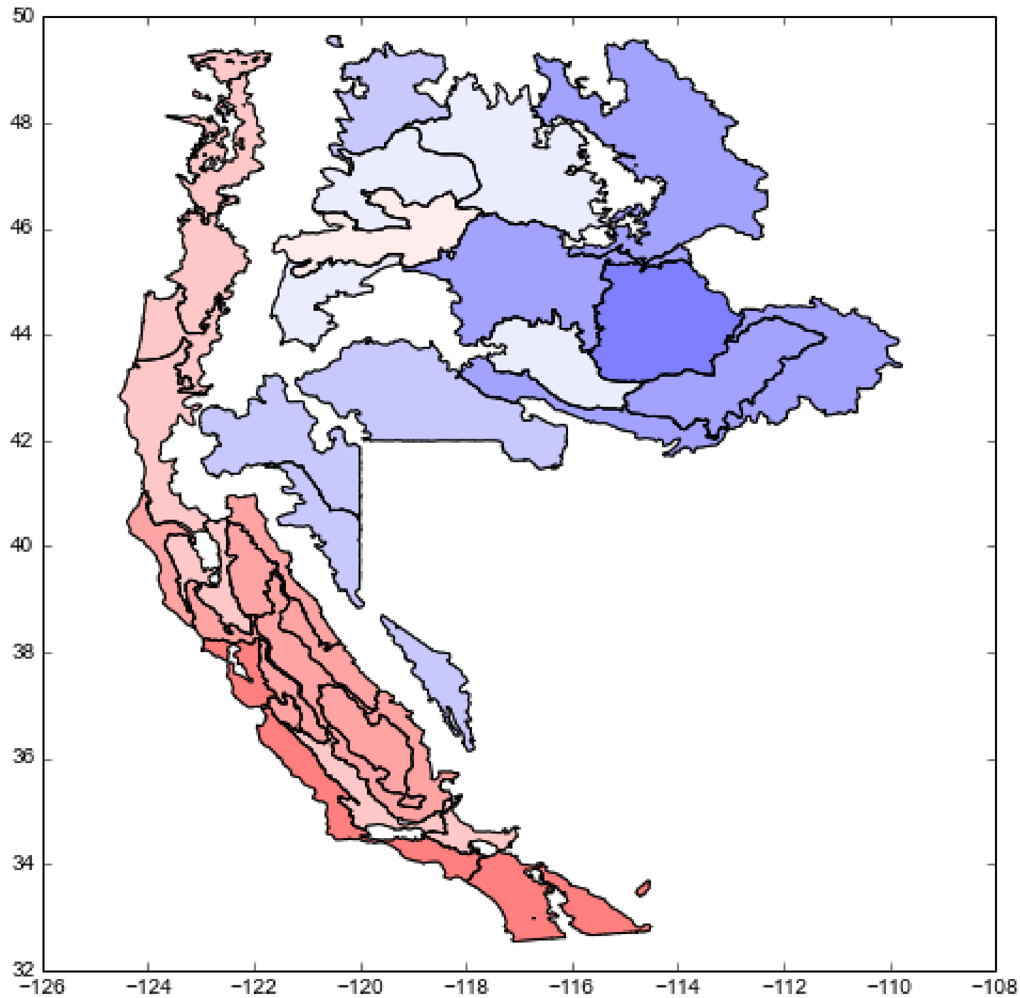
In [15]:
```
working_zones.__class__ = gpd.GeoDataFrame
working_zones.crs = orig_crs
working_zones.set_geometry('geometry')
print
```

Explore results

For example, let's create a choropleth map showing the present-day median max summer temperature by zone.

In [16]:
```
working_zones.plot(column='rcpNA_2000s_tmin12c_median', scheme='fisher_jenks', k=8, colormap='bwr')
```

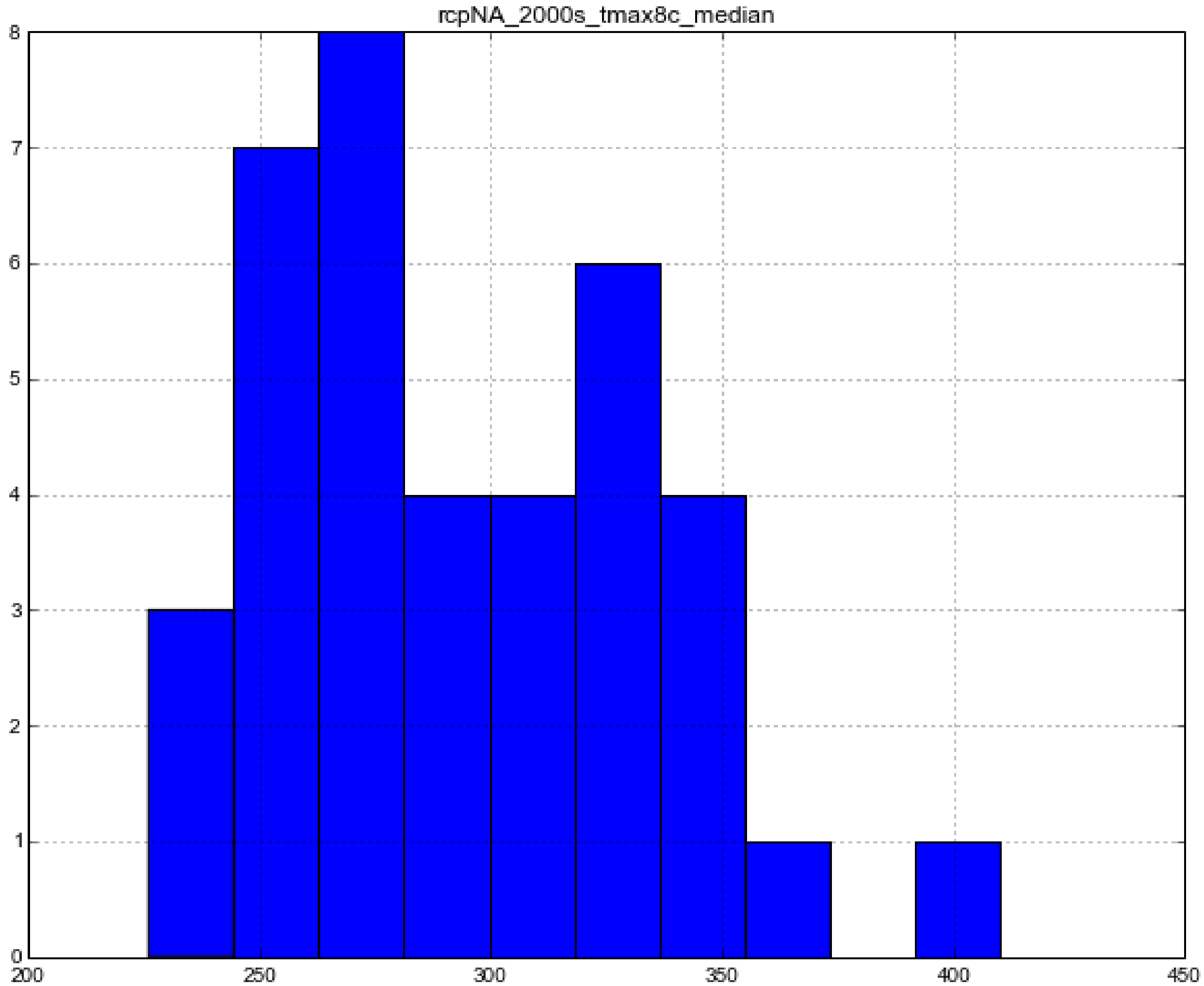
Out[16]: <matplotlib.axes.AxesSubplot at 0x7f68d4a89290>



And plot a histogram of the present-day median max summer temperatures

In [17]:
```
working_zones.hist('rcpNA_2000s_tmax8c_median')
```

Out[17]:
```
array([[<matplotlib.axes.AxesSubplot object at 0x7f68e5a03050>]], dtyp
e=object)
```



Save table to disk

Finally we'll save the table to csv and geojson formats

In [18]:
```
! rm zones_climate_summary.*
```

In [19]:
```
working_zones.to_file('zones_climate_summary.geojson', driver="GeoJSON
")

zones_tabular_only = working_zones.drop('geometry', axis=1)
zones_tabular_only.to_csv('zones_climate_summary.csv')
```

In [20]:
```
"9 climate scenarios/years X 5 variables X 6 statistical metrics = Num
ber of columns = {}".format(len(zones_tabular_only.columns)-1)
```

Out[20]:
```
'9 climate scenarios/years X 5 variables X 6 statistical metrics = Num
ber of columns = 270'
```

Metadata/Notes on data source

The source data is the "Community Earth System Model" (CESM) developed primarily at the National Center for Atmospheric Research (NCAR).

Note: Check with Jon B. for links to original metadata source.

RCP45 and RCP85 refer to two distinct "relative concentration pathways" which we commonly refer to as "low emissions" and "high emissions" respectively.

Definitions and Units for the five climate variables are as follows:

- **grwsnc**: growing season, *number of days*
- **pmean_sumrc**: mean precipitation for the summer months (June, July, August), *mm/month? check with JB on units*
- **pmean_wntrc**: mean precipitation for the winter months (December,January, February), *mm/month? check with JB on units*
- **tmax8c**: August max temp, *1/10th degree C*
- **tmin12c**: December min temp, *1/10th degree C*