Caloric and Agricultural Suitability

May 15, 2015

0.1 # Caloric Suitability Indices (CSI)

The suitability of land for agriculture (Rammankutty, Foley, Norman, and McSweeney 2001) has become a standard control for the effect of geographical characteristics on comparative economic development. This measure, however, is rather crude and it does not capture the large variation in the potential caloric yield across equally suitable land. In particular, geographical regions that according to this measure are comparable in terms of their suitability for agriculture may differ significantly in their potential caloric output per hectare per year, reflecting the fact that land that is suitable for agriculture is not necessarily suitable for the most productive crops in terms of their caloric return.

In light of the importance of pre-industrial population density in the subsequent course of economic development, and the instrumental role played by caloric yield in sustaining and supporting population growth, it is rather apparent that this commonly used index is not well designed for properly capturing the effect of the suitability of land for agriculture on economic development.

Galor and Özak (2014) rectify this deficiency and introduce a novel index of land suitability: "The Caloric Suitability Indices" (CSI) that properly capture the variation in potential crop yield across the globe, as measured in calories per hectare per year. Moreover, in light of the expansion in the set of crops that are available for cultivation in the course of the Columbian Exchange, the CSI indices provide a distinct measure for caloric suitability for the pre-1500 and the post 1500 era.
The CSI indices provide fours estimates of caloric suitability for each cell of size $5' \times$ 5 in the world:

1. The maximum potential caloric yield attainable given the set of crops that are suitable for cultivation in the pre-1500 period.
2. The maximum potential caloric yield attainable, given the set of crops that are suitable for cultivation in the post-1500 period.
3. The average potential yields within each cell attainable given the set of crops that are suitable for cultivation in the pre-1500 period.
4. The average potential yields within each cell attainable given the set of crops that are suitable for cultivation in the post-1500 period.

0.2 The Caloric Suitability Indices Data

These historical measures are constructed based on data from the Global Agro-Ecological Zones (GAEZ) project of the Food and Agriculture Organization (FAO). The GAEZ project supplies global estimates of crop yield and crop growth cycle for 48 crops in grids with cells size of $5' \times 5'$ (i.e., approximately 100 km^2).

The crops available are alfalfa, banana, barley, buckwheat, cabbage, cacao, carrot, cassava, chickpea, citrus, coconut, coffee, cotton, cowpea, dry pea, flax, foxtail millet, greengram, groundnuts, indigo rice, maize, oat, oilpalm, olive, onion, palm heart, pearl millet, phaseolus bean, pigeon pea, rye, sorghum, soybean, sunflower, sweet potato, tea, tomato, wetland rice, wheat, spring wheat, winter wheat, white potato, yams, giant yams, subtropical sorghum, tropical highland sorghum, tropical lowland, sorghum, white yams.

For each crop, GAEZ provides estimates for crop yield based on three alternative levels of inputs – high, medium, and low - and two possible categories of sources of water supply – rain-fed and irrigation. Additionally, for each input-water source category, it provides two separate estimates for crop yield, based on agro-climatic conditions, that are arguably unaffected by human intervention, and agro-ecological constraints, that could potentially reflect human intervention.

In order to capture the conditions that were prevalent during the pre-industrial era, while mitigating potential endogeneity concerns, the indices use the estimates of potential crop yield under low level of inputs and rain-fed agriculture – cultivation methods that characterized early stages of development. Moreover, the estimates of potential crop yield are based on agro-climatic constraints that are largely orthogonal to human intervention. Thus, these restrictions remove the potential concern that the level of agricultural inputs, the irrigation method, and soil quality, reflect endogenous choices that could be potentially correlated with individual preferences or institutional settings. Additionally, the choice of rain-fed conditions is further justified by the fact that, although some societies had access to irrigation prior to the industrial revolution, GAEZ's data only provides estimates based on irrigation infrastructure available during the late twentieth century

The FAO dataset provides for each cell in the agro-climatic grid the potential yield for each crop (measured in tons, per hectare, per year). These estimates account for the effect of temperature and moisture on the growth of the crop, the impact of pests, diseases and weeds on the yield, as well as climatic related "workability constraints".

In order to better capture the nutritional differences across crops, and thus to ensure comparability in the measure of crop yield, the yield of each crop in the GAEZ data (measured in tons, per hectare, per year) is converted into caloric return (measured in millions of kilo calories, per hectare, per year). This conversion is based on the caloric content of crops, as provided by the United States Department of Agriculture Nutrient Database for Standard Reference. Using the estimates of the caloric content for each crop in the GAEZ data (measured in kilo calories per 1g), a comparable measure of crop yield (in millions of kilo calories, per hectare, per year) is constructed for each crop.

Based on these estimates Galor and Özak (2014) construct the maximum potential caloric yield estimate they use in their paper. Here varios additional indices of caloric suitability are constructed and presented. First, for each cell the average caloric yield across all available crops pre- and post-1500CE is computed. Second, for each cell the total caloric yield across all available crops pre- and post-1500CE is computed. Finally, the analysis assigns to each cell the highest potential yield among the available crops pre- and post-1500CE. Additionally, for each caloric index raster the same index is constructed including and excluding cells where no calories can be produced or for averages the crops without caloric output are excluded.

Thus, the research constructs for each type of index, namely *Average*, *Total* and *Maximal* Caloric Suitability, four sets of grids: 1. Caloric Suitability pre-1500CE (without zeros) 2. Caloric Suitability pre-1500CE (with zeros) 3. Caloric Suitability post-1500CE (without zeros) 4. Caloric Suitability post-1500CE (with zeros)

These grids can be used to assess the exogenous effect of agricultural potential on various economic and social outcomes. The next section shows how it can be done and compares with another measure of agricultural suitability.

---

1 Download Options for Caloric Suitability Indices

The **Caloric Suitability Indices** can be downloaded as a zip file, or individually. They come in GeoTiff format and WGS84 projection. Use the links below to download (or you can fork the associated Github repository which contains also this notebook).

- All files (zip)
- Pre-1500CE:
  - Average Calories
  - Average Calories (No Zeros)
  - Maximum Calories
  - Maximum Calories (No Zeros)

- Post-1500CE:
  - Average Calories
  - Average Calories (No Zeros)
  - Maximum Calories
  - Maximum Calories (No Zeros)

If you use the data, please cite:
Oded Galor and Ömer Özak, 2014. "The Agricultural Origins of Time Preference," NBER Working Papers 20438, National Bureau of Economic Research, Inc..

---

2 Caloric Crop Suitability and Agricultural Suitability

This section plots the various Caloric Suitability Indices constructed following Galor and Özak (2014) and introduced in the previous section. Additionally, it compares them to the agricultural suitability index of Rammankutty, Foley, Norman, and McSweeney (2001).

Additionally, it shows how one can use open source tools, in particular Python, to extract data. In order to do the analysis a working installation of Scientific Python is required and the packages imported below have to be installed also. If needed, follow the installation instructions here or view this notebook.

Start by importing the modules to be used.

```
In [36]: from __future__ import division
         import sys, os, time
         # Math, data
         import numpy as np
         import pandas as pd
         pd.set_option('display.width', 140)
         # GIS packages
         from rasterstats import zonal_stats
         import geopandas as gp
         import georasters as gr
         from shapely.geometry import Polygon
         from fiona.crs import from_string
         import mplleaflet
         import folium
         import plotly.plotly as py
         from plotly.graph_objs import *
```

```
In [37]: %matplotlib inline
         folium.initialize_notebook()
         from IPython.display import display
         from IPython.display import DisplayObject
         from IPython.display import HTML
```

```
<IPython.core.display.HTML object>


<IPython.core.display.HTML object>
```

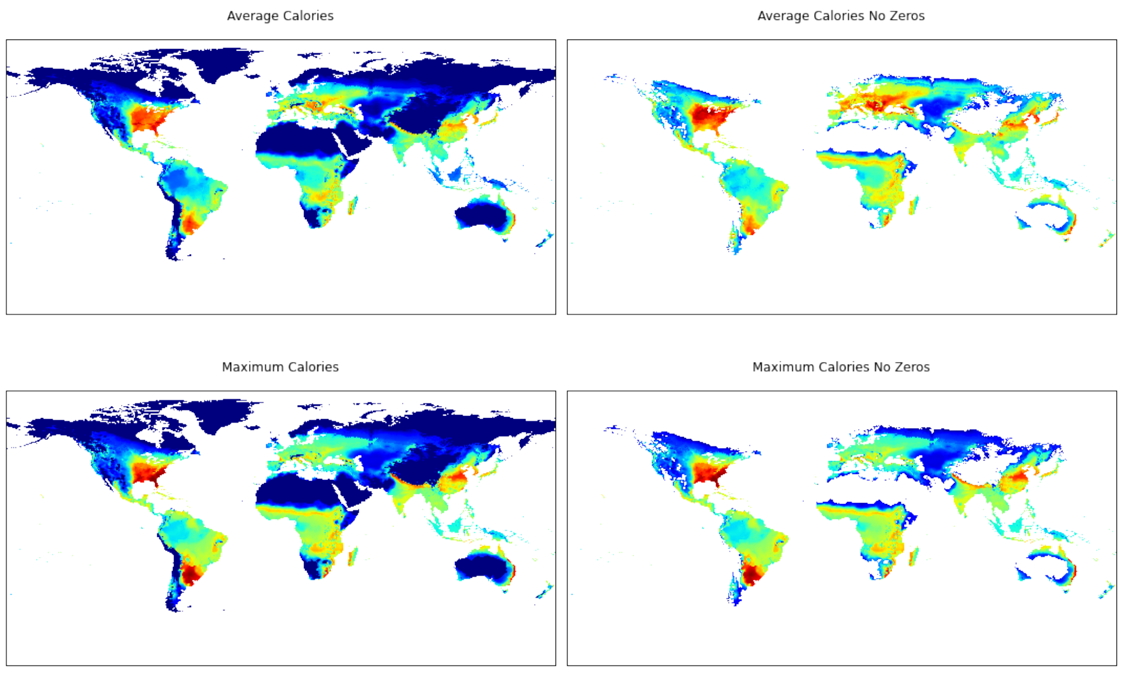
2.1 Import Post-1500CE Caloric Suitability Indices

```
In [5]: post1500mean = gr.from_file('../data/post1500AverageCalories.tif')
        post1500mean0 = gr.from_file('../data/post1500AverageCaloriesNo0.tif')
        post1500max = gr.from_file('../data/post1500OptCalories.tif')
        post1500max0 = gr.from_file('../data/post1500OptCaloriesNo0.tif')
```

2.1.1 Names ending with zero (0) exclude zeros in their construction

In particular, this means, that if for a cell c, n of the 48 crops in the FAO GAEZ data are not suitable for the production of calories, in that cell c only $48 - n$ crops will be used in the computations. Below are the plots of the 4 rasters for the post 1500CE period.

```
In [76]: f, ((ax1, ax2), (ax5, ax6)) = plt.subplots(2,2,figsize=(15,10))
         # Subplot 1
         ax1.matshow(post1500mean.raster)
         ax1.set_title('Average Calories')
         ax1.set_xticks([],[])
         ax1.set_yticks([],[])
         # Subplot 2
         ax2.matshow(post1500mean0.raster)
         ax2.set_title('Average Calories No Zeros')
         ax2.set_xticks([],[])
         ax2.set_yticks([],[])
         # Subplot 3
         ax5.matshow(post1500max.raster)
         ax5.set_title('Maximum Calories')
         ax5.set_xticks([],[])
         ax5.set_yticks([],[])
         # Subplot 4
         ax6.matshow(post1500max0.raster)
         ax6.set_title('Maximum Calories No Zeros')
         ax6.set_xticks([],[])
         ax6.set_yticks([],[])
         plt.tight_layout()
         plt.show()
```



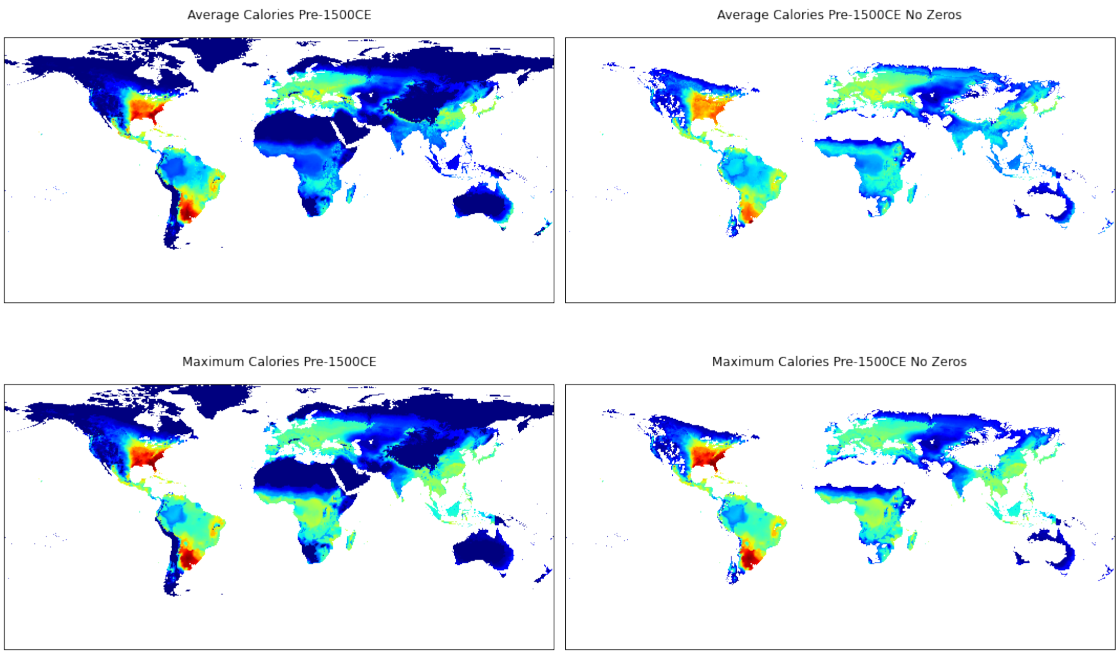
2.2 Import Pre-1500CE Caloric Suitability Indices

Now, import and plot the pre-1500CE data.

```
In [38]: pre1500mean = gr.from_file('../data/pre1500AverageCalories.tif')
         pre1500mean0 = gr.from_file('../data/pre1500AverageCaloriesNo0.tif')
         pre1500max = gr.from_file('../data/pre1500OptCalories.tif')
         pre1500max0 = gr.from_file('../data/pre1500OptCaloriesNo0.tif')
```

```
In [75]: f, ((ax1, ax2), (ax5, ax6)) = plt.subplots(2,2,figsize=(15,10))
         # Subplot 1
         ax1.matshow(pre1500mean.raster)
         ax1.set_title('Average Calories Pre-1500CE')
         ax1.set_xticks([],[])
         ax1.set_yticks([],[])
         # Subplot 2
         ax2.matshow(pre1500mean0.raster)
         ax2.set_title('Average Calories Pre-1500CE No Zeros')
         ax2.set_xticks([],[])
         ax2.set_yticks([],[])
         # Subplot 3
         ax5.matshow(pre1500max.raster)
         ax5.set_title('Maximum Calories Pre-1500CE')
         ax5.set_xticks([],[])
         ax5.set_yticks([],[])
         # Subplot 4
         ax6.matshow(pre1500max0.raster)
         ax6.set_title('Maximum Calories Pre-1500CE No Zeros')
         ax6.set_xticks([],[])
         ax6.set_yticks([],[])
```

```
plt.tight_layout()
plt.show()
```


Average Calories Pre-1500CE
Average Calories Pre-1500CE No Zeros
Maximum Calories Pre-1500CE
Maximum Calories Pre-1500CE No Zeros

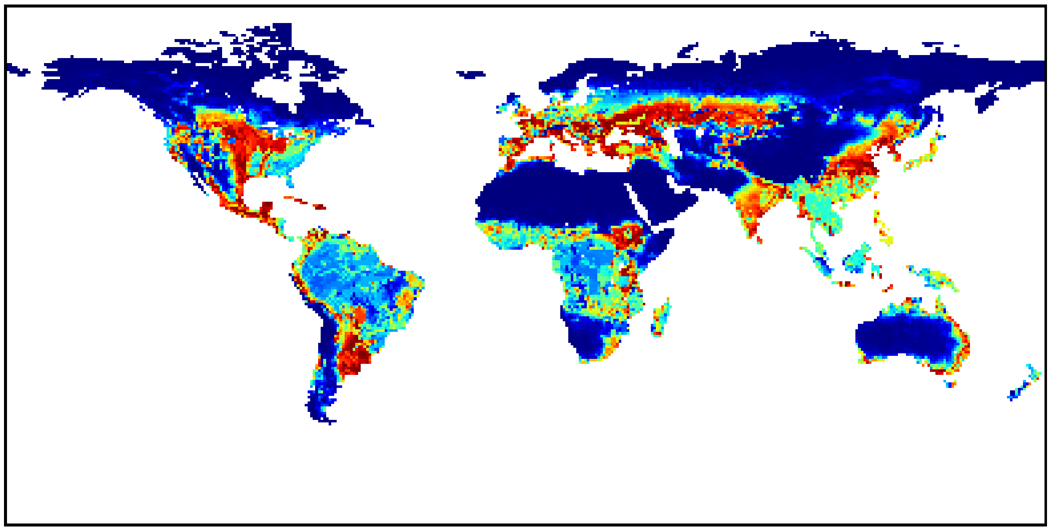
2.3 Import Rammankutty et al. data

Finally, import and plot the agricultural suitability data.

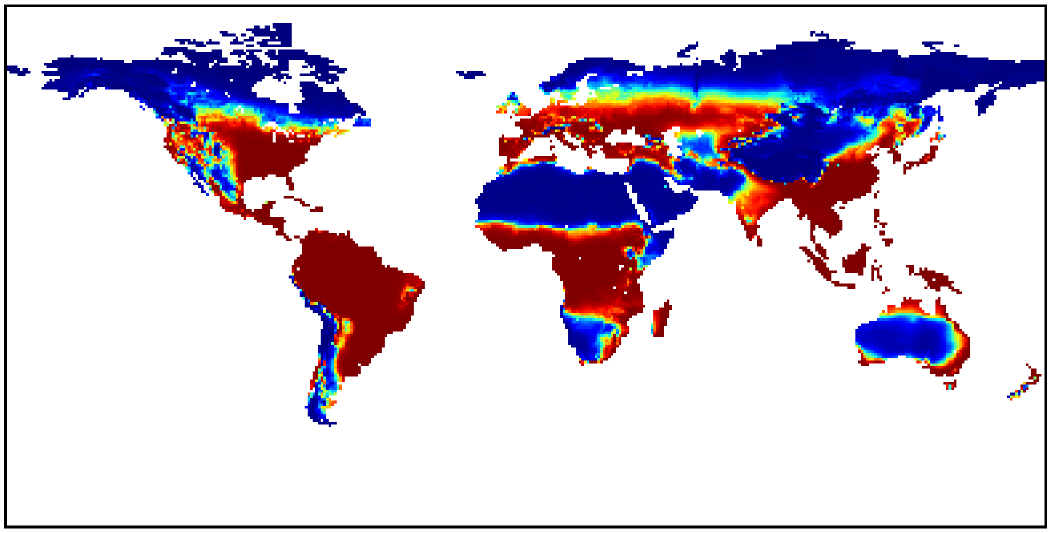
```
In [39]: rammankutty = gr.from_file('../data/suit.tif')
         climrammankutty = gr.from_file('../data/climfac.tif')
         soilrammankutty = gr.from_file('../data/soilfac.tif')

         f, (ax1, ax2, ax3) = plt.subplots(3,1,figsize=(15,10))
         ax1.matshow(rammankutty.raster)
         ax1.set_title('Agricultural Suitability')
         ax1.set_xticks([],[])
         ax1.set_yticks([],[])
         ax2.matshow(climrammankutty.raster)
         ax2.set_title('Climatic Agricultural Suitability')
         ax2.set_xticks([],[])
         ax2.set_yticks([],[])
         ax3.matshow(soilrammankutty.raster)
         ax3.set_title('Soil Agricultural Suitability')
         ax3.set_xticks([],[])
         ax3.set_yticks([],[])
         plt.tight_layout(h_pad=2.25)
         plt.show()
```

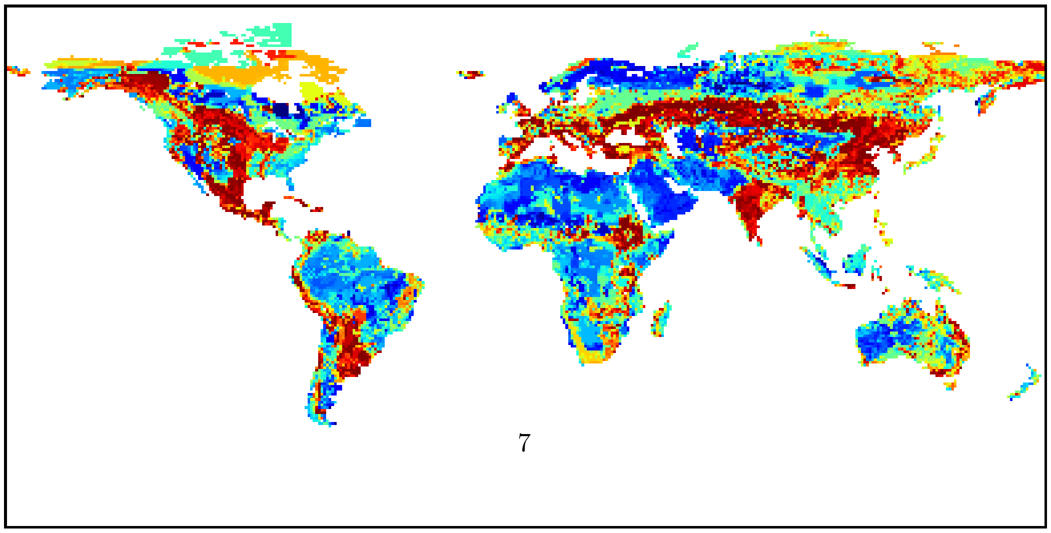
Agricultural Suitability



Climatic Agricultural Suitability



Soil Agricultural Suitability



As can be seen in the figures, most of the variation in *Agricultural Suitability* comes from the *Soil* component and not from the *Climatic* component. Thus, the use of the index in most economic research might be problematic, since for most research questions, especially in comparative development and long-run growth, the measure might be affected by human intervention.

2.4 Differences between both data sets

2.4.1 1. Caloric Suitability Indices of Galor and Özak have finer resolutions

Let's look at the sizes of the rasters

```
In [11]: print rammankutty.shape
         print post1500mean.shape
         print pre1500mean.shape
```

```
(360, 720)
(2160, 4320)
(2084, 4320)
```

So, each cell in the Rammankutty et al data is equivalent to 36 cells in the Galor and Özak dataset. This means one can work at much smaller scales. Additionally, less measurement error will be generated when extracting data for countries or smaller regions.

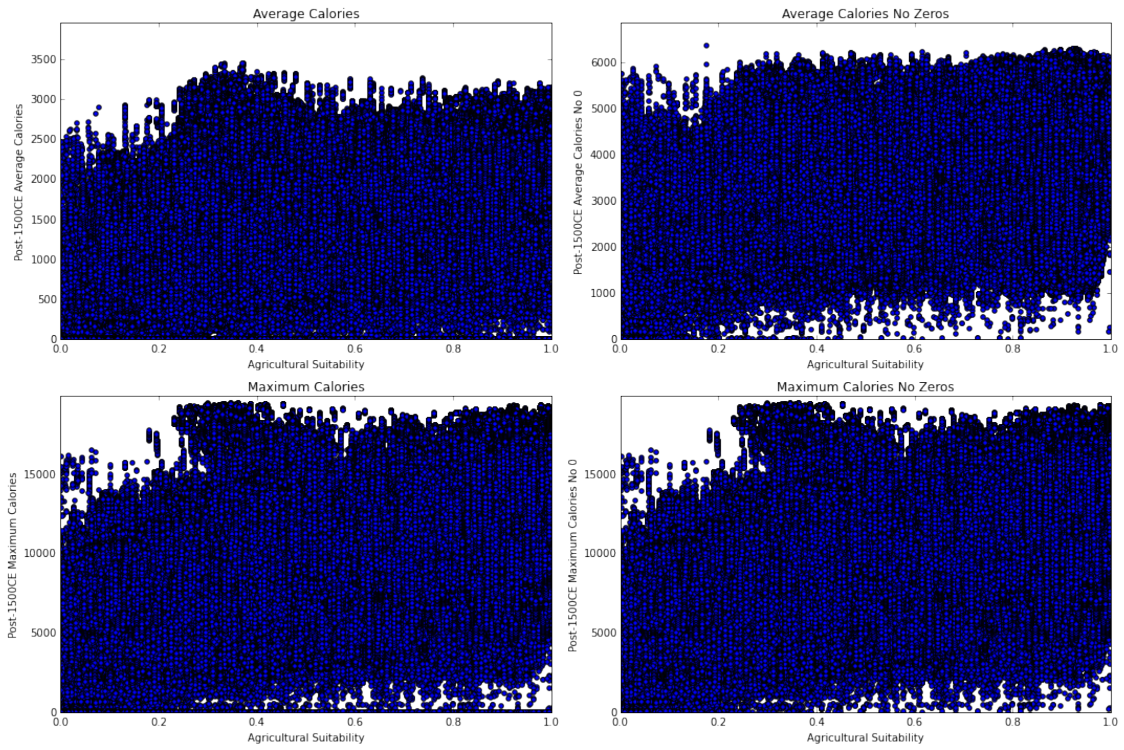
2.4.2 2. Caloric Suitability Indices are different and more essential than and Agricultural Suitability

Human existence requires consumption of sufficient calories. Thus, one can expect that mankind would evolve in regions that allow the effcient production of calories. While one can expect that agricultural and caloric suitability be (positively) correlated, they clearly are not the same concept, nor do they measure the same underlying process. In particular, as the following plots show, for any given probability of a cell being suitable for agriculture (as measured by Rammankutty et al.), the Caloric Suitability Indices vary over the full range of their possible values.

```
In [80]: rammankutty2=rammankutty.resize(post1500mean.shape)
```

```
In [81]: f, axs = plt.subplots(2,2,figsize=(15,10))
         plt.subplot(2, 2, 1)
         plt.scatter(rammankutty2.raster.ravel(),post1500mean.raster.ravel());
         plt.xlabel('Agricultural Suitability')
         plt.ylabel('Post-1500CE Average Calories')
         plt.title('Average Calories')
         plt.xlim(0,1)
         plt.ylim(0, post1500mean.max()+500)
         plt.subplot(2, 2, 2)
         plt.scatter(rammankutty2.raster.ravel(),post1500mean0.raster.ravel());
         plt.xlabel('Agricultural Suitability')
         plt.ylabel('Post-1500CE Average Calories No 0')
         plt.title('Average Calories No Zeros')
         plt.xlim(0,1)
         plt.ylim(0, post1500mean0.max()+500)
         plt.subplot(2, 2, 3)
         plt.scatter(rammankutty2.raster.ravel(),post1500max.raster.ravel());
         plt.xlabel('Agricultural Suitability')
         plt.ylabel('Post-1500CE Maximum Calories')
```

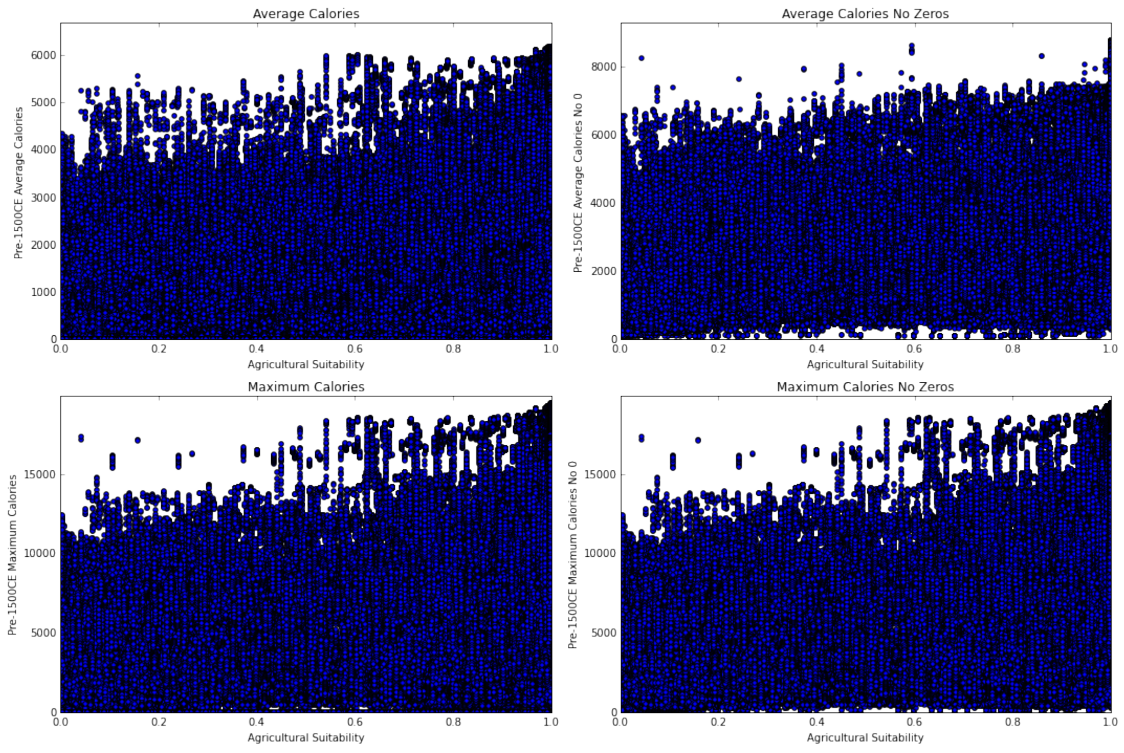
```
        plt.title('Maximum Calories')
        plt.xlim(0,1)
        plt.ylim(0, post1500max.max()+500)
        plt.subplot(2, 2, 4)
        plt.scatter(rammankutty2.raster.ravel(),post1500max0.raster.ravel());
        plt.xlabel('Agricultural Suitability')
        plt.ylabel('Post-1500CE Maximum Calories No 0')
        plt.title('Maximum Calories No Zeros')
        plt.xlim(0,1)
        plt.ylim(0, post1500max0.max()+500)
        plt.tight_layout(h_pad=1.25)
        plt.show()
```



Similar results if instead one uses the Pre-1500CE Caloric Suitability Indices

```
In [14]: rammankutty2=rammankutty.resize(pre1500mean.shape)
```

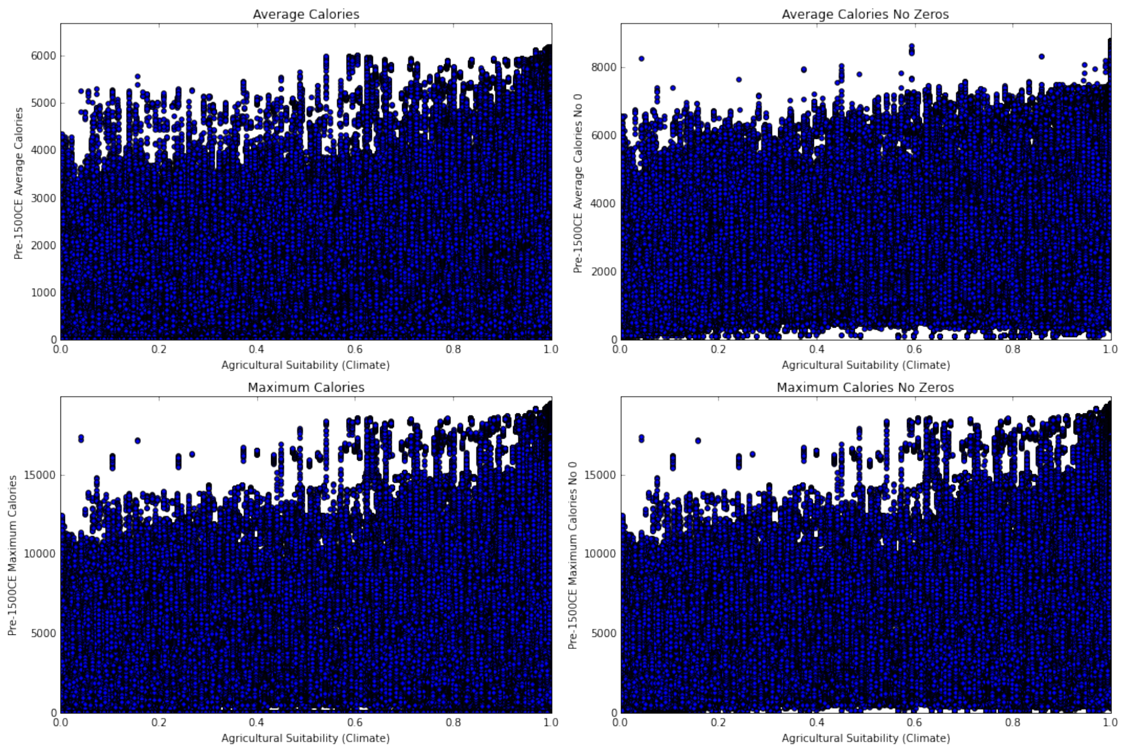
```
In [18]: f, axs = plt.subplots(2,2,figsize=(15,10))
        plt.subplot(2, 2, 1)
        plt.scatter(rammankutty2.raster.ravel(),pre1500mean.raster.ravel());
        plt.xlabel('Agricultural Suitability')
        plt.ylabel('Pre-1500CE Average Calories')
        plt.title('Average Calories')
        plt.xlim(0,1)
        plt.ylim(0, pre1500mean.max()+500)
        plt.subplot(2, 2, 2)
        plt.scatter(rammankutty2.raster.ravel(),pre1500mean0.raster.ravel());
        plt.xlabel('Agricultural Suitability')
        plt.ylabel('Pre-1500CE Average Calories No 0')
        plt.title('Average Calories No Zeros')
        plt.xlim(0,1)
        plt.ylim(0, pre1500mean0.max()+500)
        plt.subplot(2, 2, 3)
        plt.scatter(rammankutty2.raster.ravel(),pre1500max.raster.ravel());
        plt.xlabel('Agricultural Suitability')
        plt.ylabel('Pre-1500CE Maximum Calories')
        plt.title('Maximum Calories')
        plt.xlim(0,1)
        plt.ylim(0, pre1500max.max()+500)
        plt.subplot(2, 2, 4)
        plt.scatter(rammankutty2.raster.ravel(),pre1500max0.raster.ravel());
        plt.xlabel('Agricultural Suitability')
        plt.ylabel('Pre-1500CE Maximum Calories No 0')
        plt.title('Maximum Calories No Zeros')
        plt.xlim(0,1)
        plt.ylim(0, pre1500max0.max()+500)
        plt.tight_layout(h_pad=1.25)
        plt.show()
```



Similar results if instead of *Agricultural Suitability* one uses the *Climatic* component

```
In [19]: rammankutty2=climrammankutty.resize(pre1500mean.shape)

In [20]: f, axs = plt.subplots(2,2,figsize=(15,10))
         plt.subplot(2, 2, 1)
         plt.scatter(rammankutty2.raster.ravel(),pre1500mean.raster.ravel());
         plt.xlabel('Agricultural Suitability (Climate)')
         plt.ylabel('Pre-1500CE Average Calories')
         plt.title('Average Calories')
         plt.xlim(0,1)
         plt.ylim(0, pre1500mean.max()+500)
         plt.subplot(2, 2, 2)
         plt.scatter(rammankutty2.raster.ravel(),pre1500mean0.raster.ravel());
         plt.xlabel('Agricultural Suitability (Climate)')
         plt.ylabel('Pre-1500CE Average Calories No 0')
         plt.title('Average Calories No Zeros')
         plt.xlim(0,1)
         plt.ylim(0, pre1500mean0.max()+500)
         plt.subplot(2, 2, 3)
         plt.scatter(rammankutty2.raster.ravel(),pre1500max.raster.ravel());
         plt.xlabel('Agricultural Suitability (Climate)')
         plt.ylabel('Pre-1500CE Maximum Calories')
         plt.title('Maximum Calories')
         plt.xlim(0,1)
         plt.ylim(0, pre1500max.max()+500)
         plt.subplot(2, 2, 4)
         plt.scatter(rammankutty2.raster.ravel(),pre1500max0.raster.ravel());
         plt.xlabel('Agricultural Suitability (Climate)')
         plt.ylabel('Pre-1500CE Maximum Calories No 0')
         plt.title('Maximum Calories No Zeros')
         plt.xlim(0,1)
         plt.ylim(0, pre1500max0.max()+500)
         plt.tight_layout(h_pad=1.25)
         plt.show()
```



While these figures show that there is not a strong relation between both sets of indices, it does not show the density or joint probability distibution. The following set of figures show histograms of the joint density of agricultural suitability and CSI. Feel free to move the graphs.

```
In [40]: # Functions to help producing the histograms
         def my3dhistinput(raster1,raster2, bins=30, normed=True):
             """
             Given two raster names constructs data for a 3D Histogram
             where:
                 raster1 and raster2 are strings with the name of the rasters
             """
             (x,y)=eval(raster1+'.align('+raster2+')')
             x=x.to_pandas()
             y=y.to_pandas()
             x.rename(columns={'value' : raster1}, inplace=True)
             y.rename(columns={'value' : raster2}, inplace=True)
             xy=x.merge(y)
             x_data=xy[raster1]
             y_data=xy[raster2]
             hist, xedges, yedges = np.histogram2d(x_data, y_data, bins=bins, normed=normed)

             elements = (len(xedges) - 1) * (len(yedges) - 1)
             xpos, ypos = np.meshgrid(xedges[:-1]+0.25, yedges[:-1]+0.25)

             xpos = xpos.flatten()
             ypos = ypos.flatten()
             zpos = np.zeros(elements)
             dx = 0.5 * np.ones_like(zpos)
             dy = dx.copy()
             dz = hist.flatten()
             return (xy, xpos, ypos, zpos, dx, dy, dz)

         def my3dhistinput2(xy,raster1,raster2, bins=30, normed=True):
             """
             Given a Pandas data frame and the name of two columns constructs data for a 3D Histogram
             of the two columns
             """
             x_data=xy[raster1]
             y_data=xy[raster2]
             hist, xedges, yedges = np.histogram2d(x_data, y_data, bins=bins, normed=normed)

             elements = (len(xedges) - 1) * (len(yedges) - 1)
             xpos, ypos = np.meshgrid(xedges[:-1]+0.25, yedges[:-1]+0.25)

             xpos = xpos.flatten()
             ypos = ypos.flatten()
             zpos = np.zeros(elements)
             dx = 0.5 * np.ones_like(zpos)
             dy = dx.copy()
             dz = hist.flatten()
             return (xpos, ypos, zpos, dx, dy, dz)
```

```
In [22]: # Construct joint density information for plots
         rammankutty2=rammankutty.resize(post1500mean.shape)
         (xypost1500mean, xpospost1500mean, ypospost1500mean, zpospost1500mean, dxpost1500mean, dypost15
         (xypost1500mean0, xpospost1500mean0, ypospost1500mean0, zpospost1500mean0, dxpost1500mean0, dyp
         (xypost1500max, xpospost1500max, ypospost1500max, zpospost1500max, dxpost1500max, dypost1500ma
         (xypost1500max0, xpospost1500max0, ypospost1500max0, zpospost1500max0, dxpost1500max0, dypost15

         rammankutty2=rammankutty.resize(pre1500mean.shape)
         (xypre1500mean, xpospre1500mean, ypospre1500mean, zpospre1500mean, dxpre1500mean, dypre1500mean
         (xypre1500mean0, xpospre1500mean0, ypospre1500mean0, zpospre1500mean0, dxpre1500mean0, dypre150
         (xypre1500max, xpospre1500max, ypospre1500max, zpospre1500max, dxpre1500max, dypre1500max, dzp
         (xypre1500max0, xpospre1500max0, ypospre1500max0, zpospre1500max0, dxpre1500max0, dypre1500max0

In [42]: # Format info for surface graph
         def HistoInfo(dz=None,xpos=None,ypos=None, xtitle=None, ytitle=None, title=None, bins=30):
             """docstring for SurfaceInfo"""
             data = Data([
                 Surface(
                     z=dz.reshape(bins,bins),
                     x=xpos.reshape(bins,bins)[0],
                     y=ypos.reshape(bins,bins).T[0],
                 )
             ])

             # Dictionary of style options for all axes
             axis = dict(
                 showbackground=True, # (!) show axis background
                 backgroundcolor="rgb(204, 204, 204)", # set background color to grey
                 gridcolor="rgb(255, 255, 255)",       # set grid line color
                 zerolinecolor="rgb(255, 255, 255)",   # set zero grid line color
                 #tickangle=0,
                 tickfont={'size':10},
                 titlefont={'size':14},
             )

             layout = Layout(
                 title=title,
                 scene=Scene(  # (!) axes are part of a 'scene' in 3d plots
                 xaxis=XAxis(axis, title=xtitle, ), # set x-axis style
                 yaxis=YAxis(axis, title=ytitle), # set y-axis style
                 zaxis=ZAxis(axis, title='Density'),  # set z-axis style
                 cameraposition=[[0.3, 0.5, 0.65, -0.25], [.3, 0, 0], 3],
                 ),
                 autosize=False,
                 width=650,
                 height=650,
                 margin=Margin(
                     l=65,
                     r=50,
                     b=65,
                     t=90
                 )
             )
             return data, layout
```

```
In [64]: # Generate formatting for Histograms
         datapost1500mean, layoutpost1500mean = HistoInfo(dz=dzpost1500mean,xpos=xpospost1500mean,ypos=y
         datapost1500mean0, layoutpost1500mean0 = HistoInfo(dz=dzpost1500mean0,xpos=xpospost1500mean0,yp
         datapost1500max, layoutpost1500max, = HistoInfo(dz=dzpost1500max,xpos=xpospost1500max,ypos=ypos
         datapost1500max0, layoutpost1500max0 = HistoInfo(dz=dzpost1500max0,xpos=xpospost1500max0,ypos=y
         datapre1500mean, layoutpre1500mean = HistoInfo(dz=dzpre1500mean,xpos=xpospre1500mean,ypos=yposp
         datapre1500mean0, layoutpre1500mean0 = HistoInfo(dz=dzpre1500mean0,xpos=xpospre1500mean0,ypos=y
         datapre1500max, layoutpre1500max, = HistoInfo(dz=dzpre1500max,xpos=xpospre1500max,ypos=ypospre1
         datapre1500max0, layoutpre1500max0 = HistoInfo(dz=dzpre1500max0,xpos=xpospre1500max0,ypos=yposp
```

```
In [65]: # Generate commands for plots
         for i in ['post1500mean','post1500mean0','post1500max','post1500max0',
                   'pre1500mean','pre1500mean0','pre1500max','pre1500max0',]:
             print('fig'+i+' = Figure(data=data'+i+', layout=layout'+i+', )')
             print('webaddress'+i+'=py.plot(fig'+i+', auto_open=False, filename="'+str(i)+'")')
             print('print(webaddress'+i+')')
             print('py.iplot(fig'+i+', filename="'+str(i)+'")')
             print('' )
```

```
figpost1500mean = Figure(data=datapost1500mean, layout=layoutpost1500mean, )
webaddresspost1500mean=py.plot(figpost1500mean, auto_open=False, filename="post1500mean")
print(webaddresspost1500mean)
py.iplot(figpost1500mean, filename="post1500mean")

figpost1500mean0 = Figure(data=datapost1500mean0, layout=layoutpost1500mean0, )
webaddresspost1500mean0=py.plot(figpost1500mean0, auto_open=False, filename="post1500mean0")
print(webaddresspost1500mean0)
py.iplot(figpost1500mean0, filename="post1500mean0")

figpost1500max = Figure(data=datapost1500max, layout=layoutpost1500max, )
webaddresspost1500max=py.plot(figpost1500max, auto_open=False, filename="post1500max")
print(webaddresspost1500max)
py.iplot(figpost1500max, filename="post1500max")

figpost1500max0 = Figure(data=datapost1500max0, layout=layoutpost1500max0, )
webaddresspost1500max0=py.plot(figpost1500max0, auto_open=False, filename="post1500max0")
print(webaddresspost1500max0)
py.iplot(figpost1500max0, filename="post1500max0")

figpre1500mean = Figure(data=datapre1500mean, layout=layoutpre1500mean, )
webaddresspre1500mean=py.plot(figpre1500mean, auto_open=False, filename="pre1500mean")
print(webaddresspre1500mean)
py.iplot(figpre1500mean, filename="pre1500mean")

figpre1500mean0 = Figure(data=datapre1500mean0, layout=layoutpre1500mean0, )
webaddresspre1500mean0=py.plot(figpre1500mean0, auto_open=False, filename="pre1500mean0")
print(webaddresspre1500mean0)
py.iplot(figpre1500mean0, filename="pre1500mean0")

figpre1500max = Figure(data=datapre1500max, layout=layoutpre1500max, )
webaddresspre1500max=py.plot(figpre1500max, auto_open=False, filename="pre1500max")
print(webaddresspre1500max)
py.iplot(figpre1500max, filename="pre1500max")

figpre1500max0 = Figure(data=datapre1500max0, layout=layoutpre1500max0, )
```

```
webaddresspre1500max0=py.plot(figpre1500max0, auto_open=False, filename="pre1500max0")
print(webaddresspre1500max0)
py.iplot(figpre1500max0, filename="pre1500max0")
```

```
In [66]: figpost1500mean = Figure(data=datapost1500mean, layout=layoutpost1500mean, )
         webaddresspost1500mean=py.plot(figpost1500mean, auto_open=False, filename="post1500mean")
         print(webaddresspost1500mean)
         py.iplot(figpost1500mean, filename="post1500mean")
```

```
https://plot.ly/~ozak/25
```

```
Out[66]: <plotly.tools.PlotlyDisplay object>
```

```
In [67]: figpost1500mean0 = Figure(data=datapost1500mean0, layout=layoutpost1500mean0, )
         webaddresspost1500mean0=py.plot(figpost1500mean0, auto_open=False, filename="post1500mean0")
         print(webaddresspost1500mean0)
         py.iplot(figpost1500mean0, filename="post1500mean0")
```

```
https://plot.ly/~ozak/26
```

```
Out[67]: <plotly.tools.PlotlyDisplay object>
```

```
In [68]: figpost1500max = Figure(data=datapost1500max, layout=layoutpost1500max, )
         webaddresspost1500max=py.plot(figpost1500max, auto_open=False, filename="post1500max")
         print(webaddresspost1500max)
         py.iplot(figpost1500max, filename="post1500max")
```

```
https://plot.ly/~ozak/27
```

```
Out[68]: <plotly.tools.PlotlyDisplay object>
```

```
In [69]: figpost1500max0 = Figure(data=datapost1500max0, layout=layoutpost1500max0, )
         webaddresspost1500max0=py.plot(figpost1500max0, auto_open=False, filename="post1500max0")
         print(webaddresspost1500max0)
         py.iplot(figpost1500max0, filename="post1500max0")
```

```
https://plot.ly/~ozak/28
```

```
Out[69]: <plotly.tools.PlotlyDisplay object>
```

```
In [70]: figpre1500mean = Figure(data=datapre1500mean, layout=layoutpre1500mean, )
         webaddresspre1500mean=py.plot(figpre1500mean, auto_open=False, filename="pre1500mean")
         print(webaddresspre1500mean)
         py.iplot(figpre1500mean, filename="pre1500mean")
```

```
https://plot.ly/~ozak/29
```

```
Out[70]: <plotly.tools.PlotlyDisplay object>
```

```
In [71]: figpre1500mean0 = Figure(data=datapre1500mean0, layout=layoutpre1500mean0, )
         webaddresspre1500mean0=py.plot(figpre1500mean0, auto_open=False, filename="pre1500mean0")
         print(webaddresspre1500mean0)
         py.iplot(figpre1500mean0, filename="pre1500mean0")
```

```
https://plot.ly/~ozak/30
```

```
Out[71]: <plotly.tools.PlotlyDisplay object>
```

```
In [72]: figpre1500max = Figure(data=datapre1500max, layout=layoutpre1500max, )
         webaddresspre1500max=py.plot(figpre1500max, auto_open=False, filename="pre1500max")
         print(webaddresspre1500max)
         py.iplot(figpre1500max, filename="pre1500max")
```

```
https://plot.ly/~ozak/31
```

```
Out[72]: <plotly.tools.PlotlyDisplay object>
```

```
In [45]: figpre1500max0 = Figure(data=datapre1500max0, layout=layoutpre1500max0, )
         webaddresspre1500max0=py.plot(figpre1500max0, auto_open=False, filename="pre1500max0")
         print(webaddresspre1500max0)
         py.iplot(figpre1500max0, filename="pre1500max0")
```

```
https://plot.ly/~ozak/49
```

```
Out[45]: <plotly.tools.PlotlyDisplay object>
```

```
In [26]: # Print iframe to be used above plots
         for i in ['post1500mean','post1500mean0','post1500max','post1500max0',
                   'pre1500mean','pre1500mean0','pre1500max','pre1500max0',]:
             webaddress=eval('webaddress'+i)
             print("HTML('<iframe src="+webaddress+" width=1000 height=600></iframe>')")
```

```
HTML('<iframe src=https://plot.ly/~ozak/25 width=1000 height=600></iframe>')
HTML('<iframe src=https://plot.ly/~ozak/26 width=1000 height=600></iframe>')
HTML('<iframe src=https://plot.ly/~ozak/27 width=1000 height=600></iframe>')
HTML('<iframe src=https://plot.ly/~ozak/28 width=1000 height=600></iframe>')
HTML('<iframe src=https://plot.ly/~ozak/29 width=1000 height=600></iframe>')
HTML('<iframe src=https://plot.ly/~ozak/30 width=1000 height=600></iframe>')
HTML('<iframe src=https://plot.ly/~ozak/31 width=1000 height=600></iframe>')
HTML('<iframe src=https://plot.ly/~ozak/32 width=1000 height=600></iframe>')
```

```
In [27]: HTML('<iframe src=https://plot.ly/~ozak/25 width=1000 height=600></iframe>')

Out[27]: <IPython.core.display.HTML object>

In [28]: HTML('<iframe src=https://plot.ly/~ozak/26 width=1000 height=600></iframe>')

Out[28]: <IPython.core.display.HTML object>

In [30]: HTML('<iframe src=https://plot.ly/~ozak/27 width=1000 height=600></iframe>')

Out[30]: <IPython.core.display.HTML object>

In [31]: HTML('<iframe src=https://plot.ly/~ozak/28 width=1000 height=600></iframe>')

Out[31]: <IPython.core.display.HTML object>

In [32]: HTML('<iframe src=https://plot.ly/~ozak/29 width=1000 height=600></iframe>')

Out[32]: <IPython.core.display.HTML object>

In [33]: HTML('<iframe src=https://plot.ly/~ozak/30 width=1000 height=600></iframe>')

Out[33]: <IPython.core.display.HTML object>

In [34]: HTML('<iframe src=https://plot.ly/~ozak/31 width=1000 height=600></iframe>')

Out[34]: <IPython.core.display.HTML object>

In [46]: HTML('<iframe src=https://plot.ly/~ozak/49 width=1000 height=600></iframe>')

Out[46]: <IPython.core.display.HTML object>
```