Computer Lab 3b: Plotting GISTEMP Data

Climate Data Analysis, ATS 301, Fall 2018

Primary objectives for today:

- Read GISTEMP data from files into arrays
- Plot three temperature time series on one plot

Secondary objectives:

- Learn about plotting options

The Data Set

I have already downloaded the global-mean monthly, seasonal, and annual means filedata from the GISTEMP website (http://data.giss.nasa.gov/gistemp/ (http://data.giss.nasa.gov/gistemp/)). The files are in the directory: `/data/ATS_301/Data/` .

[To download this data youself, scroll to the section titled "Table Data: Global and Hemispheric Monthly Means and Zonal Annual Means." We are using the Combined Land-Surface Air and Sea-Surface Water Temperature Anomalies (Land-Ocean Temperature Index, LOTI). Read the short note underneath explaining the difference between this and the dTs data.]

This file is in *csv (comma-separated values)* format. Take a quick look at the file (navigate to the directory in Jupyter from the "Files" tab and click on the link). You can see how compact the data is in this format. csv means that individual values are separated by commas. Each line corresponds to data from a different year. At the top are a few lines that indicate which data set it is (always a good thing to know), and what the columns correspond to. Which column has the annual average?

Also note that some of the data points have *** rather than numbers. What do you think this means?

Our goal is to read in this data in some usable format, so we will use a `numpy` *array* to store the data.

As before, we start by specifing that we want plots to be displayed inside the Jupyter Notebook, and load the modules we'll need.

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

The function we will use to read the data is `np.genfromtxt` . Start by using the `help` function to read up a little on this function. In particular, look at the summary info at the top, and scroll through some of the optional arguments you can use.

In [ ]:
```
help(np.genfromtxt)
```

Since the output is long, you may want to enable scrolling on that output. Select that cell, then click on the white space to the left of the cell. You can also go to "Cell"-> "Current Outputs"->"Toggle Scrolling". Or toggle scrolling by pressing "shift"-o in command (as opposed to edit) mode.

The only required input for the `np.genfromtxt` command is `fname` , which is (usually) the name of a file.

In [ ]:
```
gistemp=np.genfromtxt("/data/ATS_301/Data/GLB.Ts_dSST_2018_Sept.csv")
```

Hm. That didn't seem to work too well. Any idea what's wrong?

Take a look at the optional paramters you can specify. If you don't specify a `delimiter`, what does python use? What delimiter is actually used in the csv file? (Hint, what does "csv" stand for?)

Another parameter to note is `skip_header`. How many header rows are there in the file?

Try the command again, this time specifing the correct delimiter and skip_header.

In [ ]:
```
# Fill in the correct values below where there are ????s.

gistemp=np.genfromtxt("/data/ATS_301/Data/GLB.Ts_dSST_2018_Sept.csv", delimiter='???', skip_header=???)
```

In [ ]:
```
# ↔
```

Hopefully, you didn't get any error messages. So, lets take a look at the data.

In [ ]:
```
gistemp
```

Because the array is so large, the output uses `...` to indicate omitted data. Use the `np.shape` function to see the size of the array.

In [ ]:
```
np.shape(gistemp)
```

(Remember that we have to specify the `np.` part in front of `shape` to indicate which module it's in.)

If you just want to see the number of dimensions in a numpy array, use `np.ndim`:

In [ ]:
```
np.ndim(gistemp)
```

Looking at the datafile (using jupyter), which column corresponds to the years? Verify this by printing out just that column. (Remember that the first index is 0, and ":" means everything in that dimension.)

In [ ]:
```
# Figure out the correct indices to use here.
gistemp[???,???]
```

In [ ]:
```
# ↔
```

Now, find the column with the annual average data, and see if it's what you expect.

In [ ]:
```
# Replace with the correct index. (Don't peak below.)
gistemp[???,???]
```

In [ ]:
```
# ↔
```

Notice that the last value is "nan", for "not a number". Why is there not a value for the annual average 2018 temperature anomaly? Fortunately, the `plt.plot` command automatically excludes `nan` data.

Plot the annual average data, like we did in the earlier lab.

In [ ]:
```
plt.plot(gistemp[:,13],'sk-',markersize=5)
```

Now, add the year data to the x-axis, like we did last time, plus the axis labels and title.

In [ ]:
```
plt.plot(gistemp[:,0],gistemp[:,13],'sk-',markersize=5)
plt.ylabel("Temperature Anomaly (°C)")
plt.xlabel("Year")
plt.title("Global Mean Estimates based on Land and Ocean Data (GISTEMP)",size=10)
```

Since we want to focus on the anomalies, adding a zero line on the y axis will help identify years that are greater, less than, or close to the 1951-1980 average value. The `plt.axhline` command does this.

In [ ]:
```
plt.plot(gistemp[:,0],gistemp[:,13],'sk-',markersize=5)
plt.axhline(color='k')
plt.ylabel("Temperature Anomaly (°C)")
plt.xlabel("Year")
plt.title("Global Mean Estimates based on Land and Ocean Data (GISTEMP)",size=10)
```

Looks pretty good!

Before moving on, let's save the citation info, so have documentation of what we did in one place:

- GISTEMP Team, 2018: GISS Surface Temperature Analysis (GISTEMP). NASA Goddard Institute for Space Studies. Dataset accessed 2018-09-16 at http://data.giss.nasa.gov/gistemp/ (http://data.giss.nasa.gov/gistemp/).
- Hansen, J., R. Ruedy, M. Sato, and K. Lo, 2010: Global surface temperature change, Rev. Geophys., 48, RG4004, doi:10.1029/2010RG000345.

Hemispheric Data

Next, read in the data for the Northern Hemisphere and Southern Hemisphere averages. Repeat the steps above, saving the data to different variable names (`gistemp_nh` and `gistemp_sh` , for example).

In [ ]:
```
# Add commands to read in NH and SH data here. Replace the XXXXX's.

gistemp_nh=np.genfromtxt("XXXXXX",delimiter='XXXXXX', skip_header=XXXXXX)
gistemp_sh=np.genfromtxt("XXXXXX",delimiter='XXXXXX', skip_header=XXXXXX)
```

In [ ]:
```
# ↔
```

After you have read the data into arrays, plot all three averages on one plot. Unless you tell python to start a new figure, it will continue adding lines onto the same plot, so we can just use three `plt.plot` commands in a row. Use different colors to indicate the different data sets. Also, I decreased the `markersize` so that the plot doesn't look too crowded.

Note the `label=` options below. This allows us to use `plt.legend` to add a legend.

In [ ]:
```
plt.plot(gistemp[:,0],gistemp[:,13],'sk-',markersize=3,label="Global")
plt.plot(gistemp_nh[:,0],gistemp_nh[:,13],'sb-',markersize=3,label="NH")
plt.plot(gistemp_sh[:,0],gistemp_sh[:,13],'sr-',markersize=3,label="SH")
plt.axhline(color='k')
plt.ylabel("Temperature Anomaly (°C)")
plt.xlabel("Year")
plt.title("Mean Estimates based on Land and Ocean Data (GISTEMP)",size=10)
plt.legend(loc="best")
```

If you want to get rid of the `<matlplotlib.legend ...>` text, add the `plt.show()` command, which wraps up the figure cleanly. It is not necessary, though.

In [ ]:
```
plt.plot(gistemp[:,0],gistemp[:,13],'sk-',markersize=3,label="Global")
plt.plot(gistemp_nh[:,0],gistemp_nh[:,13],'sb-',markersize=3,label="NH")
plt.plot(gistemp_sh[:,0],gistemp_sh[:,13],'sr-',markersize=3,label="SH")
plt.axhline(color='k')
plt.ylabel("Temperature Anomaly (°C)")
plt.xlabel("Year")
plt.title("Mean Estimates based on Land and Ocean Data (GISTEMP)",size=10)
plt.legend(loc="best")
plt.show()
```

In [ ]:
```

```