

Using Python in climate data analysis (and plotting using NCL)

Baird Langenbrunner
AOS 218
Oct. 24th, 2013

Outline

1. Python – what is it, why use it?

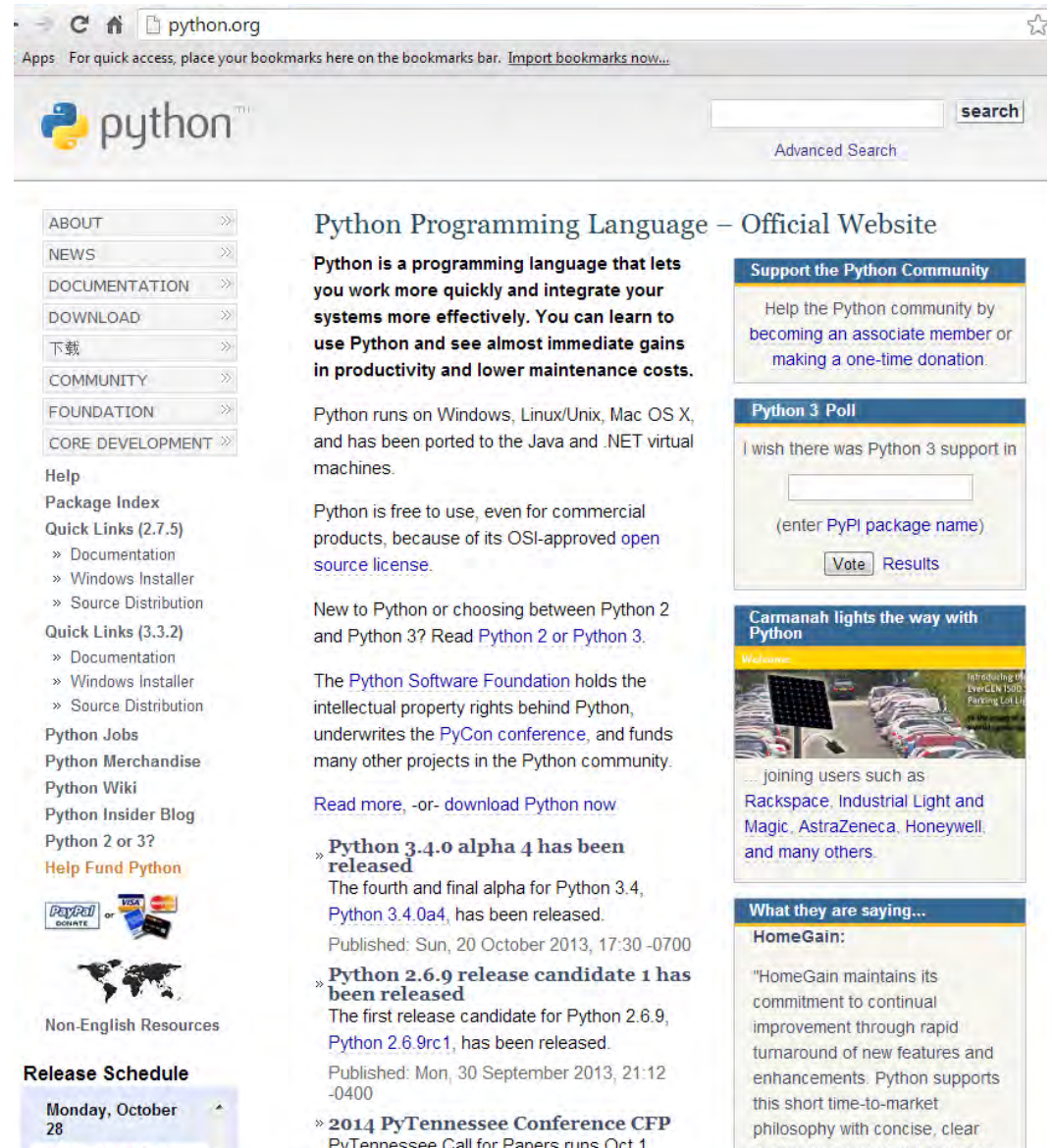
- a) Intro to Python and the SciPy “ecosystem” (SciPy *library*, NumPy, Matplotlib, etc.)
- b) UVCDAT, pyngl, pyclim, geopy - extra stuff
- c) NCL for plotting NetCDF files

2. Model uncertainties in climate change projections

- a) Intermodel disagreement/uncertainty on future projections; Knutti and Sedláček, 2012
- b) US West Coast precipitation change as a hotbed of model disagreement; storm tracks as possible cause

Python

- Programming language that began in the 1980s
- Conceived by Guido van Rossum, first official released was in 1989
- Philosophy emphasizes code readability
- Free, open source, runs on Windows, Linux/Unix, Mac
- Comes preinstalled on Mac OS X



The screenshot shows the Python.org website in a browser window. The address bar displays 'python.org'. The page features the Python logo and a search bar. A navigation menu on the left includes links for ABOUT, NEWS, DOCUMENTATION, DOWNLOAD, 下載, COMMUNITY, FOUNDATION, and CORE DEVELOPMENT. Below the menu are sections for Help, Package Index, Quick Links (2.7.5 and 3.3.2), Python Jobs, Python Merchandise, Python Wiki, Python Insider Blog, Python 2 or 3?, Help Fund Python, Non-English Resources, and Release Schedule. The main content area is titled 'Python Programming Language – Official Website' and contains several articles and announcements, including 'Python 3.4.0 alpha 4 has been released', 'Python 2.6.9 release candidate 1 has been released', and '2014 PyTennessee Conference CFP'. On the right side, there are three widgets: 'Support the Python Community', 'Python 3 Poll', and 'Carmanah lights the way with Python'. The 'Support the Python Community' widget encourages users to help the Python community by becoming an associate member or making a one-time donation. The 'Python 3 Poll' widget asks if users wish there was Python 3 support in a specific package, with a search box for the package name and buttons for 'Vote' and 'Results'. The 'Carmanah lights the way with Python' widget features a photo of solar panels and text about joining users such as Rackspace, Industrial Light and Magic, AstraZeneca, Honeywell, and many others. The 'What they are saying...' widget includes a 'HomeGain' quote: 'HomeGain maintains its commitment to continual improvement through rapid turnaround of new features and enhancements. Python supports this short time-to-market philosophy with concise, clear'.

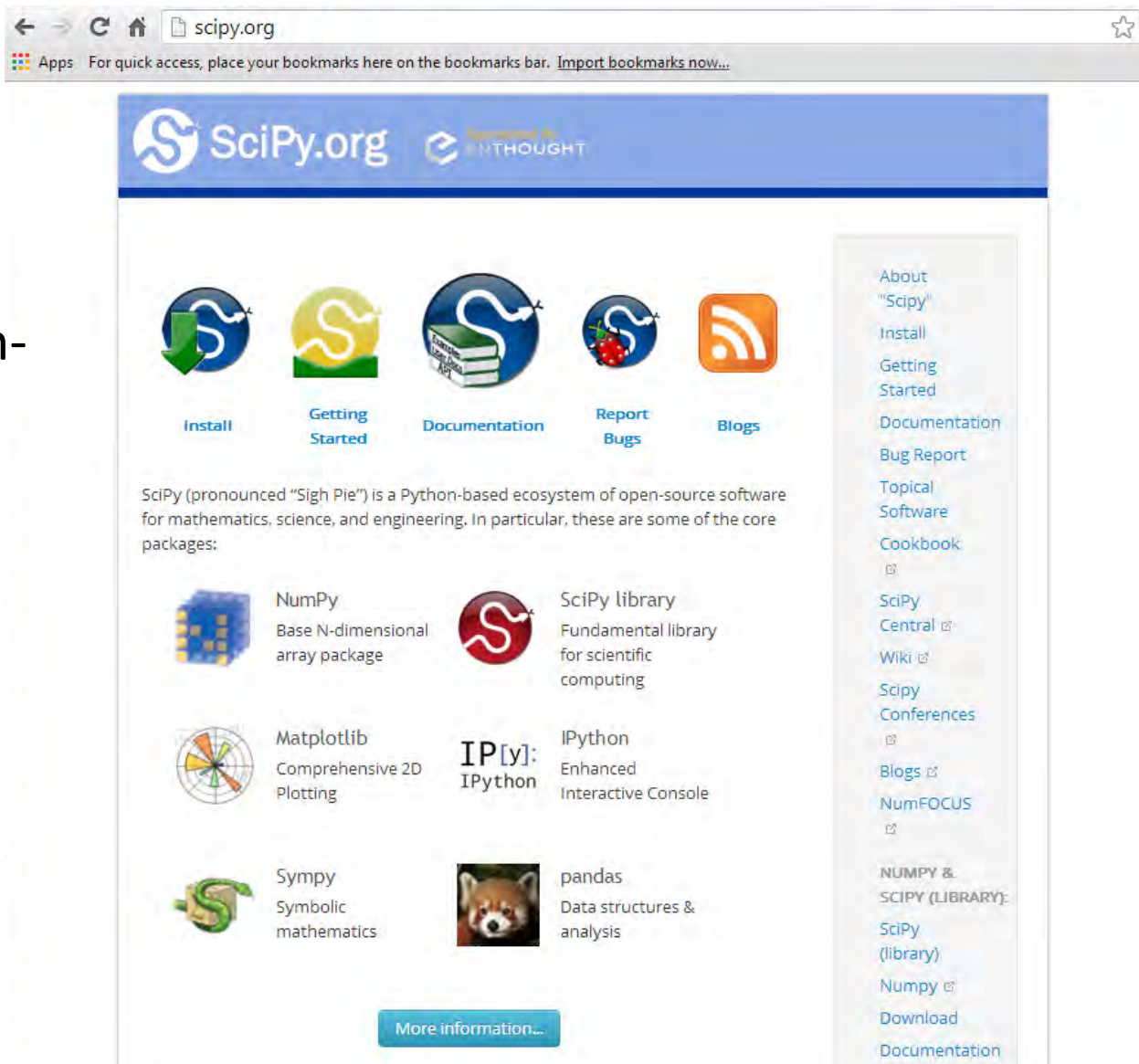
The “Zen of Python”

The Zen of Python







```
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than *right* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!
```

SciPy and the “SciPy Stack”

- SciPy = “scientific python”
- The stack is a collection of open-source software and scientific computing tools for Python



The screenshot shows the SciPy.org website. The header includes the SciPy logo and the text "Sponsored By INTHOUGHT". Below the header, there are five main navigation icons: "Install", "Getting Started", "Documentation", "Report Bugs", and "Blogs". A central text block states: "SciPy (pronounced 'Sigh Pie') is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:". Below this, there are six package cards arranged in a 3x2 grid:

 NumPy Base N-dimensional array package	 SciPy library Fundamental library for scientific computing
 Matplotlib Comprehensive 2D Plotting	 IP[y]: IPython IPython Enhanced Interactive Console
 Sympy Symbolic mathematics	 pandas Data structures & analysis

At the bottom center, there is a button labeled "More information...". On the right side, there is a vertical sidebar with a list of links: "About 'Scipy'", "Install", "Getting Started", "Documentation", "Bug Report", "Topical Software", "Cookbook", "SciPy Central", "Wiki", "SciPy Conferences", "Blogs", "NumFOCUS", "NUMPY & SCIPY (LIBRARY)", "SciPy (library)", "Numpy", "Download", and "Documentation".

SciPy and the “SciPy Stack”

- the SciPy Stack:
 - **(Python)**
 - **SciPy library**
 - **NumPy**
 - **Matplotlib**
 - *IPython, pandas, SymPy, nose*
- **Python** – the basic language on which the rest of the stack is built
- **SciPy library** – the package that provides high-end statistics and linear algebra functions; uses NumPy
- **NumPy** – “numerical python” – the package that gives Python the capability of handling large arrays and performing quick/meaty calculations
- **Matplotlib** – “mathematical plotting library” – what you use if you want to plot figures within Python

Python/SciPy packages for data analysis

Data input/output

- **NumPy input/output module** (“import numpy.io”)
 - “npz”, text, binary
- **SciPy input/output module** (“import scipy.io”)
 - Read/write/save MATLAB, IDL, and NetCDF files

Analyzing data

- **Statistics module** (“import scipy.stats”)
 - Mean, standard deviation, correlation and covariance, etc.
- **Linear algebra module** (“import scipy.linalg”)
 - SVD, CCA, EOF analysis
 - Other matrix decompositions
- **Other SciPy modules for:**
 - Fourier transforms, interpolation, optimization, integration, signal processing, etc.

Python/SciPy packages specific to NetCDF files

NetCDF input/output

- “**UV-CDAT**” (separate download)
- “**scipy.io.netcdf**” (SciPy input/output module [as on previous slide])
- “**netcdf4-python**” (separate download)
- “**PyNIO**” (separate download, from NCAR)

NetCDF plotting

- “**UV-CDAT**” (separate download)
- “**PyNGL**” (Python interface to the NCL Graphics Library, from NCAR)
- Matplotlib “**basemap**” toolkit (requires netcdf4-python)

UV-CDAT

“Ultrascale Visualization – Climate Data Analysis Tools”

- Package for Python that is great for data processing
- Developed by PCMDI at LLNL

Climate Data Analysis Tools
CDAT is deprecated and is now part of [UV-CDAT](#)

CDAT makes use of an open-source, object-oriented, easy-to-learn scripting language (Python) to link together separate software subsystems and packages to form an integrated environment for data analysis. Outside collaborators work independently and contribute on an equal basis with PCMDI.

CDAT's Modularity

User selects desired functionality... Code created...

OPeNDAPg: Globus Metadata Services

CDAT's major subsystems are:

- `cdms` - Climate Data Management System (*file I/O, variables, types, metadata, grids*)
- `cdutil` - Climate Data Specific Utilities (*spatial and temporal averages, custom seasons, climatologies*)
- `genutil` - General Utilities (*statistical and other convenience functions*)
- `numpy` - Numerical Python (*large-array numerical operations*)
- `vcs` - Visualization and Control System (*manages graphical window: picture template, graphical methods, data*)

UV-CDAT

“Ultrascale Visualization – Climate Data Analysis Tools”

- **CDAT** is the most important part of UV-CDAT
 - Has 4 pieces:
 - **cdms** – climate data management system (file i/o, variables, types, etc.)
 - **cdutil** – climate data specific utilities (spatial/temporal averages, custom seasons, climatologies)
 - **genutil** – general utilities (some rudimentary statistics)
 - **vcs** – visualization and control system (graphics/plotting) [deprecated in UV-CDAT...]

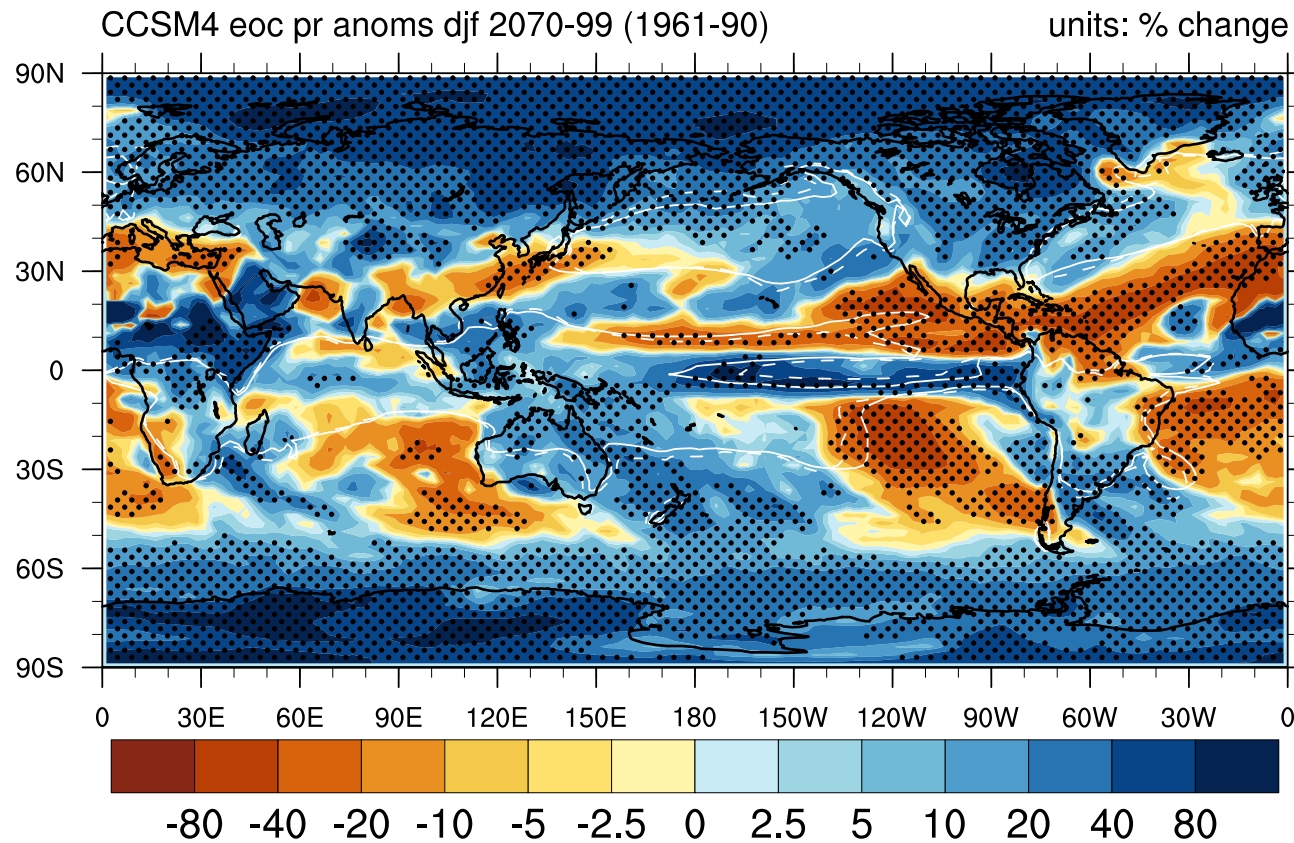
Example of a Python script

calculate DJF end-of-century precipitation change, and compute the statistical significance of this

- Plot the end-of-century precipitation change for a given model (CCSM4)
- Plot the rainfall change patterns with “stippling” where they are statistically significant at the 95% confidence level (based on a *t*-test)

Example of a Python script

calculate DJF end-of-century precipitation change, and compute the statistical significance of this



white solid line: 1961-90 4mm/day contour

white dashed line: 2070-99 4mm/day contour

Example of a Python script

calculate DJF end-of-century precipitation change, and compute the statistical significance of this

Import the necessary packages into the Python script

```
class_example_calculating_eoc_statistics.py
File Path: ~/Google Drive/sy_2013_2014/aos_218/class_example_calculating_eoc_statistics.py
~/class_example_calculating_eoc_statistics.py (no symbol selected)
#!/usr/local/uvcdat/1.2.0/bin/python2.7
# type the line below into the terminal shell for cdat to work fully:
# source /usr/local/uvcdat/1.2.0/bin/setup_cdat.sh, shell=True); import vcs
import cdutil, cftime
import cdms2 as cdms
import MV2 as MV
import numpy, numpy.linalg
import scipy.stats

#####
# CREATE A SAVING FUNCTION FOR NETCDF FILES
#####
def save_file(data=None, filename=None, var_id=None):
    if (var_id!=None and filename!=None and var_id!=None):
        save_file = cdms.open(filename, 'w')
        save_file.write(data, id=var_id)
        save_file.close()
        print "-----"
        print "File saved as", filename
        print "-----"
    else:
        print "Specify all variables (data, filename, var_id)"
cdms.setNetcdfShuffleFlag(0)
cdms.setNetcdfDeflateFlag(0)
cdms.setNetcdfDeflateLevelFlag(0)

#####
# DEFINE REGIONS FOR ANALYSIS
#####
globalI9090 = cdutil.region.domain(latitude=(-90.,90.), longitude=(0.,360.))
#tropics2525 = cdutil.region.domain(latitude=(-25.,25.), longitude=(0.,360.))
#nino34 = cdutil.region.domain(latitude=(-5.,5.), longitude=(190.,240.)) # nino34 box
#global4545 = cdutil.region.domain(latitude=(-45.,45.), longitude=(0.,360.))

#####
# DEFINE SEASONS (NOTE DJF, JJA, MAM, SON ARE ALREADY DEFINED)
#####
ONDJFM = cdutil.times.Seasons('ONDJFM')

#####
# IMPORT DATA
#####
file_example =
    cdms.open('/net/nino/ninod/cmip5/ma/bcc/rcp8.5/run1/pr/pr_185001-209912_r1i1p1_timefixed_2.5sregrid.nc')('pr')

model_names = ['CCSM4']

seasons_list = ['djf', 'mam', 'jja', 'son', 'annual', 'ondjfm']
season_to_use = 0

# ROOT OF DIRECTORY WHERE MY NETCDF FILES ARE STORED
file_root = '/net/nino/ninod/baird/cmip5/concat_nc_files/pr_regrid_2.5x2.5/'
pr_hist = ['pr_Amon_CCSM4_historical_r1i1p1_185001-200512_2.5x2.5sregrid.nc']
pr_rcp85 = ['pr_Amon_CCSM4_rcp85_r1i1p1_200601-210012_2.5x2.5sregrid.nc']

#####
# DEFINE TIME PERIOD FOR ANALYSIS (END-OF-CENTURY AND BASE PERIOD)
#####
eoc_years = numpy.array([[2070,2100]])
base_years = numpy.array([[1961,1991]])

# CONVERT THE TIME PERIODS INTO CDAT-READABLE YEARS
eoc_start_time = cftime.comptime(eoc_years[0])
eoc_end_time = cftime.comptime(eoc_years[1])
base_start = cftime.comptime(base_years[0])
base_end = cftime.comptime(base_years[1])
```


Example of a Python script

calculate DJF end-of-century precipitation change, and compute the statistical significance of this

Create a function that saves data in NetCDF format

```
class_example_calculating_eoc_statistics.py
File Path: ~/Google Drive/sy_2013_2014/aos_218/class_example_calculating_eoc_statistics.py
~/class_example_calculating_eoc_statistics.py (no symbol selected)
#!/usr/local/uvcdat/1.2.0/bin/python2.7
# type the line below into the terminal shell for cdat to work fully:
# source /usr/local/uvcdat/1.2.0/bin/setup_cdat.sh, shell=True); import vcs
import cdutil, cftime
import cdms2 as cdms
import MV2 as MV
import numpy, numpy.linalg
import scipy.stats

#####
# CREATE A SAVING FUNCTION FOR NETCDF FILES
#####
def save_file(data=None, filename=None, var_id=None):
    if (var_id!=None and filename!=None and var_id!=None):
        save_file = cdms.open(filename, 'w')
        save_file.write(data, id=var_id)
        save_file.close()
        print "-----"
        print "File saved as", filename
        print "-----"
    else:
        print "Specify all variables (data, filename, var_id)"

cdms.setNetcdfShuffleFlag(0)
cdms.setNetcdfDeflateFlag(0)
cdms.setNetcdfDeflateLevelFlag(0)

#####
# DEFINE REGIONS FOR ANALYSIS
#####
globalI9090 = cdutil.region.domain(latitude=(-90.,90.), longitude=(0.,360.))
#tropics2525 = cdutil.region.domain(latitude=(-25.,25.), longitude=(0.,360.))
#nino34 = cdutil.region.domain(latitude=(-5.,5.), longitude=(190.,240.)) # nino34 box
#global4545 = cdutil.region.domain(latitude=(-45.,45.), longitude=(0.,360.))

#####
# DEFINE SEASONS (NOTE DJF, JJA, MAM, SON ARE ALREADY DEFINED)
#####
ONDJFM = cdutil.times.Seasons('ONDJFM')

#####
# IMPORT DATA
#####
file_example =
    cdms.open('/net/nino/ninod/cmip5/ma/bcc/rcp8.5/run1/pr/pr_185001-209912_r1i1p1_timefixed_2.5sregid.nc')('pr')

model_names = ['CCSM4']

seasons_list = ['djf', 'mam', 'jja', 'son', 'annual', 'ondjfm']
season_to_use = 0

# ROOT OF DIRECTORY WHERE MY NETCDF FILES ARE STORED
file_root = '/net/nino/ninod/baird/cmip5/concat_nc_files/pr_regrid_2.5x2.5/'
pr_hist = ['pr_Amon_CCSM4_historical_r1i1p1_185001-200512_2.5x2.5sregid.nc']
pr_rcp85 = ['pr_Amon_CCSM4_rcp85_r1i1p1_200601-210012_2.5x2.5sregid.nc']

#####
# DEFINE TIME PERIOD FOR ANALYSIS (END-OF-CENTURY AND BASE PERIOD)
#####
eoc_years = numpy.array([[2070,2100]])
base_years = numpy.array([[1961,1991]])

# CONVERT THE TIME PERIODS INTO CDAT-READABLE YEARS
eoc_start_time = cftime.comptime(eoc_years[0])
eoc_end_time = cftime.comptime(eoc_years[1])
base_start = cftime.comptime(base_years[0])
base_end = cftime.comptime(base_years[1])
```


Example of a Python script

calculate DJF end-of-century precipitation change, and compute the statistical significance of this

Define seasons (regular seasons are already defined)

Insert the file names that you want to open (here, CCSM4 historical and RCP8.5 monthly data, regridded)

Define your base and end-of-century time periods

```
class_example_calculating_eoc_statistics.py
File Path: ~/Google Drive/sy_2013_2014/aos_218/class_example_calculating_eoc_statistics.py
~/class_example_calculating_eoc_statistics.py (no symbol selected)
#!/usr/local/uvcdat/1.2.0/bin/python2.7
# type the line below into the terminal shell for cdat to work fully:
# source /usr/local/uvcdat/1.2.0/bin/setup_cdat.sh, shell=True); import vcs
import cdutil, cftime
import cdms2 as cdms
import MV2 as MV
import numpy, numpy.linalg
import scipy.stats

#####
# CREATE A SAVING FUNCTION FOR NETCDF FILES
#####
def save_file(data=None, filename=None, var_id=None):
    if (var_id!=None and filename!=None and var_id!=None):
        save_file = cdms.open(filename, 'w')
        save_file.write(data, id=var_id)
        save_file.close()
        print "====="
        print "File saved as", filename
        print "====="
    else:
        print "Specify all variables (data, filename, var_id)"
cdms.setNetcdfShuffleFlag(0)
cdms.setNetcdfDeflateFlag(0)
cdms.setNetcdfDeflateLevelFlag(0)

#####
# DEFINE REGIONS FOR ANALYSIS
#####
global19090 = cdutil.region.domain(latitude=(-90.,90.), longitude=(0.,360.))
#tropics2525 = cdutil.region.domain(latitude=(-25.,25.), longitude=(0.,360.))
#nino34 = cdutil.region.domain(latitude=(-5.,5.), longitude=(190.,240.)) # nino34 box
#global4545 = cdutil.region.domain(latitude=(-45.,45.), longitude=(0.,360.))

#####
# DEFINE SEASONS (NOTE DJF, JJA, MAM, SON ARE ALREADY DEFINED)
#####
ONDJFM = cdutil.times.Seasons('ONDJFM')

#####
# IMPORT DATA
#####
file_example =
    cdms.open('/net/nino/ninod/cmip5/ma/bcc/rcp8.5/run1/pr/pr_185001-209912_r1i1p1_timefixed_2.5sregrid.nc')('pr')

model_names = ['CCSM4']

seasons_list = ['djf', 'mam', 'jja', 'son', 'annual', 'ondjfm']
season_to_use = 0

#####
# ROOT OF DIRECTORY WHERE MY NETCDF FILES ARE STORED
#####
file_root = '/net/nino/ninod/baird/cmip5/concat_nc_files/pr_regrid_2.5x2.5/'
pr_hist = ['pr_Amon_CCSM4_historical_r1i1p1_185001-200512_2.5x2.5sregrid.nc']
pr_rcp85 = ['pr_Amon_CCSM4_rcp85_r1i1p1_200601-210012_2.5x2.5sregrid.nc']

#####
# DEFINE TIME PERIOD FOR ANALYSIS (END-OF-CENTURY AND BASE PERIOD)
#####
eoc_years = numpy.array([[2070,2100]])
base_years = numpy.array([[1961,1991]])

#####
# CONVERT THE TIME PERIODS INTO CDAT-READABLE YEARS
#####
eoc_start_time = cftime.comtime(eoc_years[0])
eoc_end_time = cftime.comtime(eoc_years[1])
base_start = cftime.comtime(base_years[0])
base_end = cftime.comtime(base_years[1])
```

Example of a Python script

calculate DJF end-of-century precipitation change, and compute the statistical significance of this

Create lists to store climatologies, etc.

Open up data, extract the relevant time periods, calculate climatologies, anomalies, *t*-tests on these anomalies

```
class_example_calculating_eoc_statistics.py
File Path: ~/Google Drive/sy_2013_2014/aos_218/class_example_calculating_eoc_statistics.py
class_example_calculating_eoc_statistics.py (no symbol selected)

77
78
79 #####
80 # THE LINES BELOW OPEN THE MODEL'S HISTORICAL AND RCP8.5 DATA,
81 # PLACE THE CLIMATOLOGIES, ANOMALIES, PVALS, AND STDEVS INTO LISTS,
82 # AND THEN SAVES EACH OF THEM AS A SEPARATE NETCDF FILE
83 #####
84 print "Calculating departures from base period for models"
85
86
87 # SPECIFY WHICH SEASON YOU'RE CALCULATING FOR
88 # THIS STRING WILL GO INTO FILE NAMES LATER
89 pr_baseclim_list = []
90 pr_eocclim_list = []
91 pr_climdepartures_list = []
92 pr_baseseason_list = []
93 pr_eocseason_list = []
94 pr_departures_list = []
95 base_stdevs_list = []
96 eoc_stdevs_list = []
97 pvals_list = []
98
99 print "Calculating departures for the", seasons_list[season_to_use], "season"
100 # SPECIFY WHICH MODELS YOU WANT TO DO THESE CALCULATIONS FOR
101 which_mods = 0
102 i = which_mods
103
104
105 print "Opening mod", model_names[i]
106 pr_eoc = cdms.open(file_root+pr_rcp85[i])('pr')
107 pr_base = cdms.open(file_root+pr_hist[i])('pr')
108 # CONVERT TO MM/DAY (DATA ORIGINALLY IN MM/SEC)
109 pr_eoc = pr_eoc*86400
110 pr_base = pr_base*86400
111 pr_eoc = pr_eoc(time=(eoc_start_time, eoc_end_time, 'ca'))(global9090)
112 pr_base = pr_base(time=(base_start, base_end, 'ca'))(global9090)
113 nrows,ncols = pr_eoc.shape[1:3]
114
115 pr_baseseason_list.append(cdutil.DJF(pr_base))
116 pr_eocseason_list.append(cdutil.DJF(pr_eoc))
117 pr_baseclim_list.append(cdutil.DJF.climatology(pr_base))
118 pr_eocclim_list.append(cdutil.DJF.climatology(pr_eoc))
119
120 pr_departures_list.append(cdutil.DJF.departures(pr_eocseason_list[-1], ref=pr_baseclim_list[-1]))
121 pr_climdepartures_list.append(cdutil.DJF.departures(pr_eocclim_list[-1], ref=pr_baseclim_list[-1]))
122
123
124
125
126 #####
127 # CALCULATE INTERNAL VARIABILITY (STDEVS AT EACH GRIDPOINT FOR BOTH BASE AND END-OF-CENTURY)
128 #####
129 base_stdevs = numpy.zeros((nrows,ncols))
130 eoc_stdevs = numpy.zeros((nrows,ncols))
131 for j in range(nrows):
132     for k in range(ncols):
133         base_stdevs[j,k] = numpy.std(pr_baseseason_list[-1].data[:,j,k], ddof=1)
134         eoc_stdevs[j,k] = numpy.std(pr_eocseason_list[-1].data[:,j,k], ddof=1)
135 base_stdevs_list.append(base_stdevs)
136 eoc_stdevs_list.append(eoc_stdevs)
137
138 #####
139 # CALCULATE STATISTICAL SIGNIFICANCE OF CHANGE IN MEAN FOR BASE AND EOC PRECIP
140 # THIS IS A TWO-TAILED T-TEST FOR INDEPENDENT MEANS
141 #####
142 pvals = numpy.zeros((nrows,ncols))
143 for j in range(nrows):
144     for k in range(ncols):
145         pvals[j,k] = scipy.stats.ttest_ind(pr_baseseason_list[-1].data[:,j,k], pr_eocseason_list[-1].data[:,j,k])[1]
146 pvals_list.append(pvals)
147
148 print pr_climdepartures_list[0][0][0,:].shape, "is the shape of the eoc pr anom field"; print "-----"
149
150
151
152
153
Line 124 Col 1 Python Unicode (UTF-8) Unix (LF) Last saved: 10/23/13 8:25:29 PM 10.637 / 790 / 211
```


Example of a Python script

calculate DJF end-of-century precipitation change, and compute the statistical significance of this

Save climatologies, anomalies, and results of t -test as separate NetCDF files

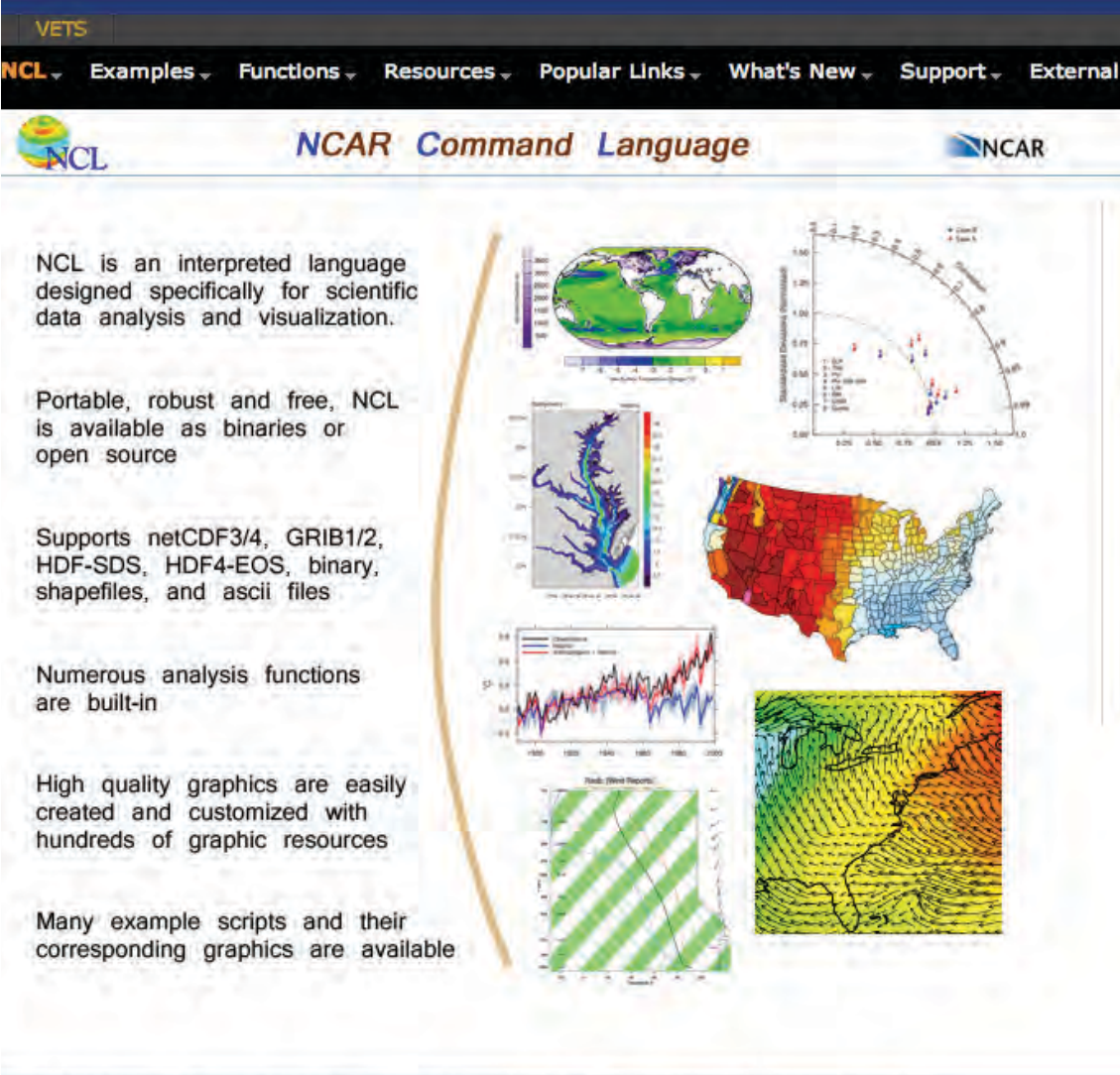
```
class_example_calculating_eoc_statistics.py
File Path: ~/Google Drive/sy_2013_2014/aos_218/class_example_calculating_eoc_statistics.py
class_example_calculating_eoc_statistics.py (no symbol selected)

153 #####
154 # SAVE END-OF-CENTURY PRECIPITATION CHANGES
155 #####
156 # SAVE END-OF-CENTURY PRECIPITATION CHANGES
157 #####
158 model_num = i
159 saveas = '/net/nino/ninod/baird/cmip5/eoc_calculations/pr/' + model_names[model_num] + '_pr_eoc_anoms_2070-99_1961-90_'
160 + seasons_list[season_to_use] + '.nc'
161 save_file(data=pr_climdepartures_list[-1][0,:,:], filename=saveas, var_id='pr')
162 #####
163 # SAVE BASE PERIOD CLIMATOLOGY (FOR PLOTTING 4MM/DAY CONTOURS)
164 #####
165 model_num = i
166 saveas = '/net/nino/ninod/baird/cmip5/eoc_calculations/pr/' + model_names[model_num] + '_pr_1961-90_climatology_' +
167 seasons_list[season_to_use] + '.nc'
168 save_file(data=pr_baseclim_list[-1][0,:,:], filename=saveas, var_id='pr')
169 #####
170 # SAVE END-OF-CENTURY PRECIPITATION CLIMATOLOGY
171 #####
172 model_num = i
173 saveas = '/net/nino/ninod/baird/cmip5/eoc_calculations/pr/' + model_names[model_num] + '_pr_2070-99_climatology_' +
174 seasons_list[season_to_use] + '.nc'
175 save_file(data=pr_eocclim_list[-1][0,:,:], filename=saveas, var_id='pr')
176 #####
177 # SAVE BASE PERIOD STDEVS (MEASURE OF INTERNAL VARIABILITY FOR THAT SEASON) AS NETCDF FILE
178 #####
179 model_num = i
180 vec = base_stdevs_list[-1]
181 ID = 'pr'
182 stdevs_nc = MV.array(vec, typecode='F', id=ID)
183 stdevs_nc.setAxis(0, file_example.getLatitude())
184 stdevs_nc.setAxis(1, file_example.getLongitude())
185 saveas = '/net/nino/ninod/baird/cmip5/eoc_calculations/pr/' + model_names[model_num] + '_pr_1961-90_stdevs_' +
186 seasons_list[season_to_use] + '.nc'
187 save_file(data=stdevs_nc, filename=saveas, var_id='pr')
188 #####
189 # SAVE STDEVS FOR EOC PERIOD AS NETCDF FILE
190 #####
191 model_num = i
192 vec = eoc_stdevs_list[-1]
193 ID = 'pr'
194 stdevs_nc = MV.array(vec, typecode='F', id=ID)
195 stdevs_nc.setAxis(0, file_example.getLatitude())
196 stdevs_nc.setAxis(1, file_example.getLongitude())
197 saveas = '/net/nino/ninod/baird/cmip5/eoc_calculations/pr/' + model_names[model_num] + '_pr_2070-99_stdevs_' +
198 seasons_list[season_to_use] + '.nc'
199 save_file(data=stdevs_nc, filename=saveas, var_id='pr')
200 #####
201 # SAVE P-VALUES FROM T-TEST AS A NETCDF FILE
202 #####
203 model_num = i
204 vec = pvals_list[-1]
205 ID = 'pvals'
206 pvals_nc = MV.array(vec, typecode='F', id=ID)
207 pvals_nc.setAxis(0, file_example.getLatitude())
208 pvals_nc.setAxis(1, file_example.getLongitude())
209 #pvals_nc.mask = pvals_nc>0.05
210 saveas = '/net/nino/ninod/baird/cmip5/eoc_calculations/pr/' + model_names[model_num] + '_pr_eoc_change_pvals_' +
211 seasons_list[season_to_use] + '.nc'
212 save_file(data=pvals_nc, filename=saveas, var_id='pvals')
```

NCL

now plot your NetCDF file using NCL

- NCL – NCAR Command Language
- Really versatile language; easy/free to download and install, great support community
- Created for “scientific data analysis and visualization”
- Easily reads in NetCDF “.nc” files, handles attributes like latitude and longitude well



The screenshot shows the NCL website interface. At the top, there is a navigation bar with the following items: VETS, NCL (with a dropdown arrow), Examples (with a dropdown arrow), Functions (with a dropdown arrow), Resources (with a dropdown arrow), Popular Links (with a dropdown arrow), What's New (with a dropdown arrow), Support (with a dropdown arrow), and External. Below the navigation bar is the NCL logo on the left, the text "NCAR Command Language" in the center, and the NCAR logo on the right. The main content area features a grid of scientific plots and text descriptions. The text descriptions include: "NCL is an interpreted language designed specifically for scientific data analysis and visualization.", "Portable, robust and free, NCL is available as binaries or open source", "Supports netCDF3/4, GRIB1/2, HDF-SDS, HDF4-EOS, binary, shapefiles, and ascii files", "Numerous analysis functions are built-in", "High quality graphics are easily created and customized with hundreds of graphic resources", and "Many example scripts and their corresponding graphics are available". The plots include a world map, a polar plot, a river network map, a US map with a color scale, a time-series plot, a contour plot, and a vector field plot.

©2013 UCAR | Privacy Policy | Terms of Use | Contact the Webmaster | Sponsored by NSF

NCL

now plot your NetCDF file using NCL

Load NCL code necessary for plotting

Define the file names for what you will be plotting

Open the anomalies, the t -test results...

```
class_example_plotting_pr_change_with_pvals.ncl
File Path: ~/Google Drive/sy_2013_2014/aos_218/class_example_plotting_pr_change_with_pvals.ncl
1
2 Script for plotting*:
3 (1) End-of-century precipitation changes for a specific model, with an overlay of
4 (2) A dot pattern where these changes are statistically signif. at the 95% confidence interval,
5 (3) The base period climatological 4mm/day precip contour as a solid line (storm tracks in base period), and
6 (4) The end-of-century climatological 4mm/day contour as a dotted line (to see changes in storm tracks)
7
8 * Note all of these are plotted for the DJF season
9
10 Created by Baird Langenbrunner for NCL version 6.0.0
11 October 23, 2013
12 AOS 218 Presentation
13
14
15 load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_code.ncl"
16 load "$NCARG_ROOT/lib/ncarg/nclscripts/csm/gsn_csm.ncl"
17
18 begin
19 ; OPEN FILE AND READ IN DATA
20
21 season = "djf"
22
23 model_names = (/ "ACCESS1-0", "ACCESS1-3", "bcc-csm1-1", "bcc-csm1-1-m", "BNU-ESM",
24 "CanESM2", "CCSM4", "CESM1-BGC", "CESM1-CAM5", "CMCC-CESM", "CMCC-CM", "CMCC-CMS", "CNRM-CM5",
25 "CSIRO-Mk3-6-0", "EC-EARTH", "FGOALS-g2", "GFDL-CM3", "GFDL-ESM2G", "GFDL-ESM2M", "GISS-E2-H",
26 "GISS-E2-R", "HadGEM2-AO", "HadGEM2-CC", "HadGEM2-ES", "inmcm4", "IPSL-CM5A-LR", "IPSL-CM5A-MR",
27 "IPSL-CM5B-LR", "MIROC5", "MIROC-ESM", "MIROC-ESM-CHEM", "MPI-ESM-LR", "MPI-ESM-MR", "MRI-CGCM3",
28 "NorESM1-M", "NorESM1-ME" /)
29
30 do i=0,35
31
32 ; SPECIFY THE FILES YOU WANT TO OPEN (NETCDF FILES HERE, PREVIOUSLY SAVED IN PYTHON)
33
34 model_num = i
35 model = model_names(model_num)
36
37 file_root = "~/Dropbox/cmip5_data/temp_nc_files/"
38 filename_pr_base_stdevs = file_root + model + "_pr_1961-90_stdevs_"+season+".nc"
39 filename_pr_eoc_anoms = file_root + model + "_pr_eoc_anoms_2070-99_1961-90_"+season+".nc"
40 filename_pr_eoc_pvals = file_root + model + "_pr_eoc_change_pvals_"+season+".nc"
41 filename_pr_base_clim = file_root + model + "_pr_1961-90_climatology_"+season+".nc"
42 filename_pr_eoc_clim = file_root + model + "_pr_2070-99_climatology_"+season+".nc"
43
44 ; OPEN FILES, SPECIFY THE LATITUDES AND LONGITUDES JUST TO BE SAFE
45
46 setfileoption("nc", "MissingToFillValue", False)
47
48 ncfile1 = addfile(filename_pr_eoc_anoms, "r")
49 ncfile2 = addfile(filename_pr_eoc_pvals, "r")
50 ncfile3 = addfile(filename_pr_base_clim, "r")
51 ncfile4 = addfile(filename_pr_eoc_clim, "r")
52
53 ; PR END OF CENTURY ANOMALIES
54 pr_eoc_anoms = ncfile1->pr(:,,:)
55 pr_eoc_anoms@units = ""
56 lat = ncfile1->lat
57 lon = ncfile1->lon
58 pr_eoc_anoms!0 = "lat"
59 pr_eoc_anoms!1 = "lon"
60 lat@units = "degrees_north"
61 lon@units = "degrees_east"
62 pr_eoc_anoms&lat = lat
63 pr_eoc_anoms&lon = lon
64
65 ; PR EOC PVALS FOR TTEST OF INDEPENDENT MEANS
66 pr_eoc_pvals = ncfile2->pvals(:,,:)
67 pr_eoc_pvals@units = ""
68 lat = ncfile2->lat
69 lon = ncfile2->lon
70 pr_eoc_pvals!0 = "lat"
71 pr_eoc_pvals!1 = "lon"
72 lat@units = "degrees_north"
73 lon@units = "degrees_east"
74 pr_eoc_pvals&lat = lat
75 pr_eoc_pvals&lon = lon
76
77
```

NCL

now plot your NetCDF file using NCL

... and the base/end-of-century climatologies

Define what you want to save the plot as

Open a work station

Declare the resources for each component of the plot individually (anomalies, the base map, the significance test results, and the end-of-century and base period climatologies)

```
class_example_plotting_pr_change_with_pvals.ncl
File Path: ~/Google Drive/sy_2013_2014/aos_218/class_example_plotting_pr_change_with_pvals.ncl
73 ; PR BASE PERIOD CLIMATOLOGY
74 pr_base_clim = ncfile3->pr(:,:)
75 pr_base_clim@units = ""
76 lat = ncfile3->lat
77 lon = ncfile3->lon
78 pr_base_clim!0 = "lat"
79 pr_base_clim!1 = "lon"
80 lat@units = "degrees_north"
81 lon@units = "degrees_east"
82 pr_base_clim&lat = lat
pr_base_clim&lon = lon

85 ; PR END OF CENTURY CLIMATOLOGY
86 pr_eoc_clim = ncfile4->pr(:,:)
87 pr_eoc_clim@units = ""
88 lat = ncfile4->lat
89 lon = ncfile4->lon
90 pr_eoc_clim!0 = "lat"
91 pr_eoc_clim!1 = "lon"
92 lat@units = "degrees_north"
93 lon@units = "degrees_east"
94 pr_eoc_clim&lat = lat
95 pr_eoc_clim&lon = lon

97 ; CONVERT EOC ANOMALIES INTO A PERCENT CHANGE BY DIVIDING THEM BY THE MODEL'S BASE PERIOD CLIMATOLOGY
98 pr_eoc_anoms = (pr_eoc_anoms/pr_base_clim)*100.

100 -----
101 ; SET THE DIRECTORY AND FILE NAME FOR THE FINISHED FIGURE
102 -----
103 root = "~/Dropbox/plots_raw/eoc_plots/pr/perc_change_with_pvals/"
104 saveas = root + "pr_eoc_anoms_" + model + "_" + season + "_1961-90_2070-99"

106 -----
107 ; CREATE COLOR BAR AND A "WORK STATION"
108 -----
109 wks = gsn_open_wks("x11", saveas)
110 gsn_define_colormap(wks, "BlueYellowRed")
111 gsn_reverse_colormap(wks)

113 -----
114 ; CREATE THE PLOTTING RESOURCES FOR THE ANOMALIES
115 -----
116 res_anoms = True
117 res_anoms@gsnDraw = False
118 res_anoms@gsnFrame = False
119 res_anoms@lbLabelBarOn = True
120 res_anoms@cnFillOn = False
121 res_anoms@cnLinesOn = False
122 res_anoms@cnLineLabelsOn = False
123 res_anoms@cnFillOn = True
124 res_anoms@cnLinesOn = False
125 res_anoms@cnLineLabelsOn = False
126 res_anoms@gsnSpreadColors = True
127 res_anoms@lbLabelAutoStride = True
128 res_anoms@lbAutoManage = False
129 res_anoms@lbLabelFontHeightF = 0.015
130 res_anoms@pmLabelBarOrthogonalPosF = 0.055

132 -----
133 ; SET CONTOUR LEVELS FOR PLOT EXPLICITLY
134 res_anoms@cnLevelSelectionMode = "ExplicitLevels"
135 res_anoms@cnLevels = (/ -80., -40., -20., -10., -5., -2.5, 0, 2.5, 5., 10., 20., 40., 80. /)

137 -----
138 ; CREATE BASE MAP RESOURCES
139 -----
140 res_map = True
141 res_map@mpFillOn = True
142 res_map@gsnMaximize = True
143 res_map@gsnPaperOrientation = "auto"
144 res_map@mpOutlineOn = True
145 res_map@gsnDraw = False
146 res_map@gsnFrame = False
147 res_map@mpGeophysicalLineThicknessF = 3.
148 ; SPECIFY GLOBAL PLOTTING DOMAIN
149 res_map@mpMinLatF = -90.0
res_map@mpMaxLatF = 90.0

line 185 Col 45 | NCL Command Language | Unicode (UTF-8) | Linux (LF) | Last saved: 10/23/13 5:50:54 PM | 9,272 / 813 / 227
```


NCL

now plot your NetCDF file using NCL

continued: Declare the resources for each component of the plot individually (anomalies, the base map, the significance test results, and the end-of-century and base period climatologies)

```
class_example_plotting_pr_change_with_pvals.ncl
File Path: ~/Google Drive/sy_2013_2014/aos_218/class_example_plotting_pr_change_with_pvals.ncl
class_example_plotting_pr_change_with_pvals.ncl
149 res_map@mpMaxLatF = 90.0
150 res_map@mpMinLonF = 0.
151 res_map@mpMaxLonF = 360.
152 ;; FOR GLOBAL PLOTS
153 res_map@mpLimitMode = "LatLon"
154 res_map@mpRelativeCenterLat = True
155 res_map@mpRelativeCenterLon = True
156
157
158 ; CREATE 4mm/day CLIMATOLOGY LINE FOR BASE PERIOD
-----
159 res_base_4mm = True
160 res_base_4mm@gsnDraw = False
161 res_base_4mm@gsnFrame = False
162 res_base_4mm@cnLevelSelectionMode = "ExplicitLevels"
163 res_base_4mm@cnLevels = 4.0
164 res_base_4mm@cnLineColor = "White"
165 res_base_4mm@cnLineThicknessF = 2.
166 res_base_4mm@lbLabelBarOn = False
167 res_base_4mm@cnLineLabelsOn = False
168 res_base_4mm@cnInfoLabelOn = False
169 res_base_4mm@gsnContourPosLineDashPattern = 0
170
171
172 ; CREATE 4mm/day CLIMATOLOGY LINE FOR EOC PERIOD (dotted)
-----
173 res_eoc_4mm = True
174 res_eoc_4mm@gsnDraw = False
175 res_eoc_4mm@gsnFrame = False
176 res_eoc_4mm@cnLevelSelectionMode = "ExplicitLevels"
177 res_eoc_4mm@cnLevels = 4.0
178 res_eoc_4mm@cnLineColor = "White"
179 res_eoc_4mm@cnLineThicknessF = 2.
180 res_eoc_4mm@lbLabelBarOn = False
181 res_eoc_4mm@cnLineLabelsOn = False
182 res_eoc_4mm@cnInfoLabelOn = False
183 res_eoc_4mm@gsnContourPosLineDashPattern = 14
184
185
186 ; CREATE RESOURCES FOR T-TEST
-----
187
188 res_pvals = True
189 res_pvals@gsnDraw = False
190 res_pvals@gsnFrame = False
191 res_pvals@cnLinesOn = False
192 res_pvals@lbLabelBarOn = False
193 res_pvals@cnLineLabelsOn = False
194 res_pvals@cnInfoLabelOn = False
195 res_pvals@cnLevelSpacingF = 0.05
196 res_pvals@cnFillDotSizeF = 0.004
197 res_pvals@plot = gsn_csm_contour(wks,pr_eoc_pvals,res_pvals)
198 opt_pvals = True
199 opt_pvals@gsnShadeFillType = "pattern"
200 opt_pvals@gsnShadeLow = 17
201
202
203
204 ;
205 ; INSTANTIATE THE PLOT
206 ;
207 datestr = systemfunc("date")
208
209 title_leftalign = model+ " eoc pr anom " +season+ " 2070-99 (1961-90)"
210 res_map@gsnLeftString = title_leftalign
211 res_map@gsnRightString = "units: % change"
212
213 plot_map = gsn_csm_map_ce(wks,res_map)
214 plot_anoms = gsn_csm_contour(wks,pr_eoc_anoms,res_anoms)
215 plot_eoc_pvals = gsn_contour_shade(eoc_pvals_plot,0.05,999.,opt_pvals)
216 plot_base_4mm = gsn_csm_contour(wks,pr_base_clim,res_base_4mm)
217 plot_eoc_4mm = gsn_csm_contour(wks,pr_eoc_clim,res_eoc_4mm)
218 overlay(plot_map, plot_anoms)
219 overlay(plot_map, plot_base_4mm)
220 overlay(plot_map, plot_eoc_4mm)
221 overlay(plot_map, plot_eoc_pvals)
222 draw(plot_map)
223 frame(wks)
224 end do
225 end
```

NCL

now plot your NetCDF file using NCL

Now plot each component (the map, the anomalies, the p-values, the base and end-of-century 4mm/day contours)

Lastly, overlay them all on top of the map itself

```
class_example_plotting_pr_change_with_pvals.ncl
File Path: ~/Google Drive/sy_2013_2014/aos_218/class_example_plotting_pr_change_with_pvals.ncl
class_example_plotting_pr_change_with_pvals.ncl
149 res_map@mpMaxLatF = 90.0
150 res_map@mpMinLonF = 0.
151 res_map@mpMaxLonF = 360.
152 ;; FOR GLOBAL PLOTS
153 res_map@mpLimitMode = "LatLon"
154 res_map@mpRelativeCenterLat = True
155 res_map@mpRelativeCenterLon = True
156
157
158 ; CREATE 4mm/day CLIMATOLOGY LINE FOR BASE PERIOD
159
160 res_base_4mm = True
161 res_base_4mm@gsnDraw = False
162 res_base_4mm@gsnFrame = False
163 res_base_4mm@cnLevelSelectionMode = "ExplicitLevels"
164 res_base_4mm@cnLevels = 4.0
165 res_base_4mm@cnLineColor = "White"
166 res_base_4mm@cnLineThicknessF = 2.
167 res_base_4mm@lbLabelBarOn = False
168 res_base_4mm@cnLineLabelsOn = False
169 res_base_4mm@cnInfoLabelOn = False
170 res_base_4mm@gsnContourPosLineDashPattern = 0
171
172
173 ; CREATE 4mm/day CLIMATOLOGY LINE FOR EOC PERIOD (dotted)
174
175 res_eoc_4mm = True
176 res_eoc_4mm@gsnDraw = False
177 res_eoc_4mm@gsnFrame = False
178 res_eoc_4mm@cnLevelSelectionMode = "ExplicitLevels"
179 res_eoc_4mm@cnLevels = 4.0
180 res_eoc_4mm@cnLineColor = "White"
181 res_eoc_4mm@cnLineThicknessF = 2.
182 res_eoc_4mm@lbLabelBarOn = False
183 res_eoc_4mm@cnLineLabelsOn = False
184 res_eoc_4mm@cnInfoLabelOn = False
185 res_eoc_4mm@gsnContourPosLineDashPattern = 14
186
187
188 ; CREATE RESOURCES FOR T-TEST
189
190 res_pvals = True
191 res_pvals@gsnDraw = False
192 res_pvals@gsnFrame = False
193 res_pvals@cnLinesOn = False
194 res_pvals@lbLabelBarOn = False
195 res_pvals@cnLineLabelsOn = False
196 res_pvals@cnInfoLabelOn = False
197 res_pvals@cnLevelSpacingF = 0.05
198 res_pvals@cnFillDotSizeF = 0.004
199 res_pvals@gsnContour(wks,pr_eoc_pvals,res_pvals)
200 opt_pvals = True
201 opt_pvals@gsnShadeFillType = "pattern"
202 opt_pvals@gsnShadeLow = 17
203
204
205 ; INSTANTIATE THE PLOT
206
207 datestr = systemfunc("date")
208
209 title_leftalign = model+ " eoc pr anom " +season+ " 2070-99 (1961-90)"
210 res_map@gsnLeftString = title_leftalign
211 res_map@gsnRightString = "units: % change"
212
213
214 plot_map = gsn_csm_map_ce(wks,res_map)
215 plot_anoms = gsn_csm_contour(wks,pr_eoc_anoms,res_anoms)
216 plot_eoc_pvals = gsn_contour_shade(eoc_pvals_plot,0.05,999.,opt_pvals)
217 plot_base_4mm = gsn_csm_contour(wks,pr_base_clim,res_base_4mm)
218 plot_eoc_4mm = gsn_csm_contour(wks,pr_eoc_clim,res_eoc_4mm)
219 overlay(plot_map, plot_anoms)
220 overlay(plot_map, plot_base_4mm)
221 overlay(plot_map, plot_eoc_4mm)
222 overlay(plot_map, plot_eoc_pvals)
223 draw(plot_map)
224 frame(wks)
225 end do
226 end
```

Downloading and more info

Language/package	Website
Python	http://python.org/
SciPy Stack	http://scipy.org/ (has links to NumPy, Matplotlib, etc.)
UV-CDAT	http://uv-cdat.llnl.gov/
NCL	https://www.earthsystemgrid.org/dataset/ncl.html

- Some of the most useful tips for learning these languages and packages are contained within the “tutorials” on their separate websites. My own advice is always to start with a tutorial and build up from there.

Other helpful links

Python/SciPy/NCL/NCO tips and tricks

NumPy “for MATLAB users”

http://wiki.scipy.org/NumPy_for_Matlab_Users

Python for the Atmospheric and Oceanic Sciences

<http://pyaos.johnny-lin.com/>

- The PyAOS blog mentions that there are beginner, intermediate, and advanced courses on using Python in Climate and Meteorology at the 2014 AMS meeting in Atlanta

Think Python (learn the language)

<http://www.greenteapress.com/thinkpython/>

Python in Hydrology (learn Python with examples in hydrology)

<http://www.greenteapress.com/pythonhydro/pythonhydro.html>

Great website with NCO tricks

<http://jisao.washington.edu/data/nco/>

Other links???

ERRRGGGHHHH

troubleshooting

- Ask real people first!
 - Students in AOS use Python/SciPy, NCO, NCL, MATLAB, etc.
 - One of us has potentially already written a script to do what you need
- Or look within stackoverflow.com
 - “Question and answer site for professional and enthusiast programmers”
 - Big Python user base