

A case study on the use of a modern computer language in the analysis of climate data: PyClimate

Jon Sáenz wdpsaagj@lg.ehu.es
 Jesús Fernández chus@wm.lc.ehu.es
 Juan Zubillaga wmpzuesj@lg.ehu.es

The University of the Basque Country
 Dept. Applied Physics II



<http://www.pyclimate.org>

Toulouse 12-15 Nov 2002

Abstract

A set of routines specifically built for the analysis of climate data that make intensive use of the numerical extensions to the computer language Python are presented.

The routines perform some typical tasks during multivariate analysis of observed climate data or model output, such as EOF analysis and related tasks (truncation rules by means of analytical and Monte Carlo techniques). Other functions perform singular value decomposition of covariance matrices and canonical correlation analysis for the coupled variability of climate data. The package interfaces to a library of statistical distribution functions and also includes multivariate digital filters, time handling routines, kernel based probability density function estimation and differential operators over the sphere for gridded data sets.

The routines rely on the numerical extensions to Python language, and they are fast for numerical analysis. The programs make the analysis of data both easier and faster due to a much shorter development cycle.

Python

Python is a modern computer language featuring:

- Interactive interpreter
- Simple syntax
- Object orientation
- Multipatform support
- Fast numerical routines
- Easily extendible and embeddable
- Distribution free of charge

Taking advantage of these computing capabilities and ease of use our package PyClimate provides python with routines to perform several usual tasks in the analysis of climate data.

PyClimate is distributed under GNU's GPL license, version 2. The user can use and/or freely distribute or modify our code.

PyClimate Modules

readdat writedat ncstruct nciterator	Functions for easy access to ASCII and NetCDF stored data
svdeofs hdseofs svd bpcca	Multivariate analysis pattern decomposition tools including PCA, SVD and CCA
LinearFilter LanczosFilter KZFilter	Extensible low-pass, high-pass and band-pass linear filters
JDTIME JDTimeHandler	Time handling routines
analog atmosphericmoisture diffoperators mvarstatools pydcdflib tools	Several other modules performing analog searches, atmospheric moisture related functions, differential operators for gridded data on the sphere, access to the DCDFLIB.C statistical library...

NetCDF I/O

NetCDF is a binary auto-descriptive file format independent of the computer low-endian or big-endian architecture. PyClimate provides an easy way to create and duplicate the structure of COARDS-compliant NetCDF files.

The script below opens a NetCDF file: data/data.nc and after computing the time average of the slp variable stores it in another NetCDF file with the same structure of latitudes and longitudes.

```
import Numeric
from Scientific.IO.NetCDF import NetCDFFile
from pyclimate.ncstruct import nccpstruct

ncf = NetCDFFile("data/data.nc")
var = ncf.variables["slp"][:]
ave = Numeric.sum(var) / float(len(var))
dims = ("lat", "lon")
onc = nccpstruct("data/avedata.nc", ncf,
               dims, dims, dims)
onc.title = "My new NetCDF"
ncvar = onc.createVariable("ave", "d", dims)
ncvar.long_name = "My averaged variable"
ncvar[:, :] = ave
onc.close()
```

PCA

Principal Component Analysis routines compute the EOFs with several scalings (orthonormal, variance carrying, as correlation of the field with the PCs, as the spatial variance explained by each mode). Several tests for the stability of the modes are also implemented, such as the North et al. (1982) sampling errors, a Monte Carlo test on the congruence of sub-sampled EOFs or the Bartlett test.

There are two versions, svdeofs, which computes the EOFs directly from the data matrix and hdseofs, which uses a more careful approach to allow the computation of EOFs from huge data sets.

```
from pyclimate.readdat import readdat
from pyclimate.svdeofs import SVDEOFs

precip = readdat("data/precip.ascii")
PCA = SVDEOFs(precip)
eofs = PCA.eofs()
pcs = PCA.pcs()
north_sampling_errors = PCA.northTest()
congruences = PCA.MCTest(100, length=150)
reofs = PCA.eofsAsCorrelation()
veofs = PCA.eofsAsExplainedVariance()
```

CCA

Canonical Correlation Analysis with PCA pre-filtering is performed by the module bpcca.

The example below reads two fields (one from a NetCDF and the other from an ASCII file) and carries out the CCA (see e.g. Bretherton et al. 1992) retaining 4 EOFs of the first field and 2 of the second one. The first field is passed in its multidimensional shape (t,lat,lon) and the corresponding canonical patterns L are returned in the same way: (lat,lon,canonicalIndex).

```
from pyclimate.bpcca import BPPCCA
from pyclimate.readdat import readdat
from Scientific.IO.NetCDF import *

slp = NetCDFFile("data/data.nc").variables["slp"]
precip = readdat("data/precip.ascii")

CCA = BPPCCA(slp, precip, (4,2))

L = CCA.leftPatterns()
R = CCA.rightPatterns()
r = CCA.correlation()
a = CCA.rightExpCoeffs()
b = CCA.leftExpCoeffs()
```

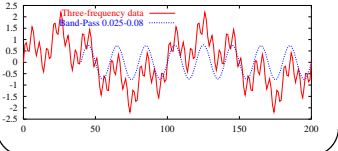
Linear Filters

The modules LanczosFilter and KZFilter allow frequency filtering of generically-shaped data. They are implemented to allow processing large data sets.

```
from pyclimate.readdat import readdat
from pyclimate.lanczosfilter import *

a = readdat("data/filter.in.xy")

npoints = 40
file = open("data/filter.out.xy", "w")
bp = LanczosFilter('bp', 0.025, 0.08, npoints)
for i in range(len(a)):
    bfa = bp.getfiltered(a[i])
    if bfa:
        file.write("%d %d\n"%(i-npoints, bfa[i]))
file.close()
```



pydcdflib

The module pydcdflib provides some bindings to an external C library (DCDFLIB.C) which provides several cumulative distribution functions that can be used in hypothesis testing.

The implemented CDFs are: beta, binomial, χ^2 , non-central χ^2 , F, non-central F, gamma, negative binomial, normal, Poisson, T and non-central T.

As an example, the code below prints to standard output a small table with the Normal CDF.

```
from pyclimate.pydcdflib import CDFNor, pydcfnor

ncdf = CDFNor()
ncdf.which = 1 # Get the cumulated
              # probability out of the
              # abscissa, mean and std

ncdf.mean = 0.0
ncdf.sd = 1.0
for ncdf.x in [0,1,2,3,4,5]:
    pydcfnor(ncdf) # Processes the 'ncdf'
                  # object to obtain the
                  # missing parameter
    print "%3.1f %8.5f %8.5f" % (ncdf.x, 100. * ncdf.p)
```

Atmospheric moisture

The module atmosphericmoisture provides functions for several computations related to atmospheric moisture, like the saturation pressure of water vapor over water and ice as a function of temperature, mixing ratio and specific humidity, dew point depression and conversion functions amongst these quantities.

```
import Numeric
from pyclimate.asciidat import readcol
from pyclimate.atmosphericmoisture import *

# Load the pressure level data
P = readcol("data/sounding98060400.dat", 1)
T = readcol("data/sounding98060400.dat", 2)
Td = readcol("data/sounding98060400.dat", 3)
dpd = T-Td

levels = len(P)

rh = Numeric.zeros((levels), Numeric.Float64)
ppw = Numeric.zeros((levels), Numeric.Float64)
for i in xrange(levels):
    rh = dewpointdepressionrh(dpd[i], P[i], T[i])
    ppw = q2e(rh2shum(rh, P[i], T[i]), P[i])
    Tv = T[i]/(1.-(ppw/P[i]))*(1-epsilon))
    print P[i], rh, ppw, Tv
```

diffoperators

This module can be used to compute the horizontal gradient of a scalar field, the divergence of a horizontal vector field and the vertical component of the rotational of a vector field over the sphere. The quantities are computed by means of a second-order centered finite-difference scheme on a regular grid of arbitrary shape as a two, three or even four-dimensional field with longitude/latitude increasing (decreasing) to the North/South or East/West. The only requirement on the input data is that it is arranged as in a COARDS-compliant file (t,z,lat,lon).

The finite difference computations have been implemented in a matrix form and are very fast.

```
from Numeric import arange, pi, exp, NewAxis
from pyclimate.diffoperators import HGRADIENT

x = arange( 0., 360., 2.5) * pi/180.
y = arange(-90., 90., 2.5) * pi/180.

# Get the gradient of z(x,y)=x*exp[-(x^2+y^2)]
xz, yz = x*x, y*y
z = x * exp(-x2>NewAxis,]-y2[.NewAxis)
hgrad=HGRADIENT(y, x, asdegrees=0, pflon=0)
gx, gy = hgrad.gradient(z)
```

Analog search

The module analog provides some tools for searching the past history of a variable looking for a base case resembling situation. Analogs can be searched in a PCA truncated space or in a CCA truncated one. The search can also be performed by averaging and/or weighting analogs and by means of different distances (euclidean, Mahalanobis, ...)

In the example below, precipitation data for the record 200 onwards are reconstructed by looking for analogs (in the 4 leading EOF's space) in a library built with the first 200 records.

```
from pyclimate.analog import *
from pyclimate.readdat import readdat

precip = readdat("data/precip.ascii")
precip_library = precip[:200]
base_cases = precip[200:]

precipA = EOFANALOG(precip_library, neofs=4)

AS = ANALOGSelector(precipA, base_cases)
precip_recons = AS.returnAnalog()
```

JDTIME and JDTimeHandler

Time handling is a must in the analysis of climate data. JDTime (written in C) is a core set of functions to handle Julian Days. JDTimeHandler is a Python class which implements functions to parse the units attribute of the time variable in NetCDF COARDS-compliant files and retrieve in a simple, precise and fast way, the fields (year, month, day, hour, minute and second) of the time value stored in this NetCDF variable. The combination of both modules provides a very easy handling of (for instance) monthly time steps.

```
from pyclimate.JDTimeHandler import JDTimeHandler
from pyclimate.JDTime import monthlstep

mstep = 24 * monthlstep() # A month in hours

# Handling monthly data: the WRONG way
jdt = JDTimeHandler("hours since 1948-01-01")
for t in [0, 1, 2, 3]:
    print jdt.getdatefields(t * mstep, 3)

# Handling monthly data: the RIGHT way
jdt = JDTimeHandler("months since 1940-2-12")
for t in [0, 1, 2, 3]:
    print jdt.getdatefields(t * mstep, 3)
```

Exceptions

The package has its own set of exceptions, which are very informative of the reasons why the computations are crashing. In those cases where the exceptions happen in parts of the code which are not under the direct control of PyClimate (like the LAPACK routines, external libraries or floating point arithmetic operations), Python's standard behavior (exception, nan or exit(), depending of the exception, platform and compiler settings) propagates to the caller routine.

```
from pyclimate.pyclimateexcept \
import KZEvenPoints, JDTHBrokenUnits
from pyclimate.KZFilter import KZFilter
from pyclimate.JDTimeHandler import JDTimeHandler

try:
    kzf=KZFilter(4,3,1)
except KZEvenPoints, e:
    print e

try:
    th=JDTimeHandler("months since 1940-2-12")
except JDTHBrokenUnits, e:
    print e
```

More info

More detailed information about PyClimate can be found in our website:

<http://www.pyclimate.org>

Full details about python are available through its official website:

<http://www.python.org>

Other interesting URLs hosting python software for the analysis of climate data are:

CDAT:

<http://esg.llnl.gov/cdat>

mtaCDF:

<http://www.ifm.uni-kiel.de/.../mtaCDF.html>

References

- C. S. Bretherton, C. Smith and J. M. Wallace
An Intercomparison of Methods for Finding Coupled Patterns in Climate Data
 Journal of Climate, 5:541-560, 1992
- G. R. North et al.
Sampling Errors in the Estimation of Empirical Orthogonal Functions
 Monthly Weather Review, 110:699-706, 1982
- J. Sáenz, J. Zubillaga and J. Fernández
Geophysical data analysis using Python
 Computers & Geosciences, 28(4):457-465, 2002

Acknowledgments

The authors wish to thank the financial support offered by the UCAR to attend this meeting. Financial support was also offered by the *Ministerio de Ciencia y Tecnología*, projects number CL198-0236 and REN2002-04584C04-04 and the *Euskal Meteorologia Zerbitua*. Jesús Fernández is granted by the *Departamento de Educación, Universidades e Investigación, Gobierno Vasco*.