Statistical tools:
The genutil Package

General Utilities : genutil

genutil contains statistical (and other) tools such as:

- statistics
- grower
- picker
- udunits

genutil.statistics

- The **statistics** module provides the user with some basic statistics function:
  - (auto)correlation
  - (auto)covariance
  - geometricmean
  - laggedcorrelation
  - laggedcovariance
  - linearregression
  - percentiles
  - meanabsdiff
  - median
  - rank
  - rms
  - std
  - variance
- See CDAT documentation and doc strings for more info:

```
>>> help(genutil.statistics.geometricmean)
```

genutil.statistics.correlation (1)

- `genutil.statistics.correlation()` returns the correlation between 2 slabs. By default on the first dimension, centered and biased by default.
- Slabs must be of the same shape and size.

Usage:

```
result = correlation(slab1, slab2,
   weights=weightoptions, axis=axisoptions,
   centered=centeredoptions,
   biased=biasedoptions)
```

Options:

```
weightoptions
   default = None. If you want to compute the weighted
   correlation, provide the weights here.
NOTE: the weights array must be the same shape and
   size as the slabs.
```

genutil.statistics.correlation (2)

Options (continued):

- `axisoptions` ‘x’ | ‘y’ | ‘z’ | ‘t’ | ‘(dimension_name)’ | 0 | 1 ... | n

  default value = 0. You can pass the name of the dimension or index (integer value 0...n) over which you want to compute the statistic.

- `centeredoptions` None | 0 | 1

  default value = 1 computes and removes the mean first. Set to 0 or None for uncentered.

- `biasedoptions` None | 0 | 1

  default value = 1 returns biased statistic. If want to compute an unbiased statistic pass anything but 1.

genutil.statistics.correlation example

```
>>> import cdms, genutil
>>> f=cdms.open('file1.nc')
>>> var1=f('u_wind') ;  var2=f('v_wind')
>>> print var2.shape
(12, 21, 144, 288)
>>> # Get the overall correlation
>>> print genutil.statistics.correlation(var1, \
           var2, axis="tzyx")
correlation
array(-0.237745400348)
>>> # Now get the gridded array of correlations over
... # time and level
>>> tl_corr=genutil.statistics.correlation(var1, \
           var2, axis="tz")
>>> print tl_corr.shape
(144, 288)
```

genutil.statistics.std (1)

- **genutil.statistics.std()** returns the standard deviation from a slab. By default on first dimension, centered, and biased.

Usage:

```
result = std(slab, weights=weightoptions, axis =
   axisoptions, centered=centeredoptions, biased
   = biasedoptions)
```

Options:

```
weightoptions
   default = None. If you want to compute the weighted
   correlation, provide the weights here.
NOTE: the weights array must be the same shape and
   size as the slab.
```

genutil.statistics.std (2)

Options (continued):

`axisoptions` `‘x’ | ‘y’ | ‘z’ | ‘t’ | ‘(dimension_name)’ | 0 | 1 ... | n`

`default value = 0. You can pass the name of the dimension or index (integer value 0...n) over which you want to compute the statistic.`

`centeredoptions` `None | 0 | 1`

`default value = 1 computes and removes the mean first. Set to 0 or None for uncentered.`

`biasedoptions` `None | 0 | 1`

`default value = 1 returns biased statistic. If want to compute an unbiased statistic pass anything but 1.`

genutil.statistics.std example

```
>>> import cdms, genutil
>>> f=cdms.open(‘file1.nc’)
>>> var1=f(‘u_wind’)
>>> var1.shape # just a linear variable
(9,)
# We want to set some weights to add importance to
# the higher range of values
>>> wghts=[.2, .3, .5, .6, 1.0, 1.0, 1.2, 1.4, 1.0]
>>> std=genutil.statistics.std(var1, weights=wghts)
>>> print std
0.734016655568359065
```

Explaining “biased” options

A number of the statistical functions allow the user to specify:

- This comes from 2 different definition of the standard deviation:
  - exact definition, with a division by "n-1" (biased=0)
  - an approximate one with a div by "n” (biased=1).
- As "n" (number of elements) gets big (quickly) there's no difference and the second approach is faster. But for small number of elements you get a different answer.

genutil.statistics.linearregression (1)

genutil.statistics.linearregression() computes the linear regression between to one-dimensional arrays, where the independent variable (x) can be an axis or values of another data array.

Usage:

```
result = linearregression(y, axis=axisoptions,
    x=xvalues, error=erroroptions,
    probability=probabilityoptions,
    nointercept=nointerceptoptions,
    noslope=noslopeoptions)
```

Options:

axisoptions - ‘x’ | ‘y’ | ‘z’ | ‘t’ | ‘(dimension_name)’ | 0 | 1 ... | n
default value = 0. You can pass the name of the dimension or index (integer value 0...n) over which you want to treat the array as the dependent variable.

xvalues - default = None. You can pass an array of values that are to be used as the independent axis x.

genutil.statistics.linearregression - example

```
>>> import genutil.statistics as gs
>>> f1=cdms.open('~/my_cdat_files/data/lsp_200001.nc')
>>> lsp1=f1('lsp', squeeze=1)
>>> f2=co('~/my_cdat_files/data/lsp_200007.nc')
>>> lsp2=f2('lsp', squeeze=2)
>>> lsp1.shape
>>> gs.linearregression(lsp1, axis="y")
[slope
array([  2.30989548e-08,  2.56871230e-08,  ...])
, intercept
array([  7.28723094e-06,  7.50319075e-06,  ...])
>>> gs.linearregression(lsp1[0], x=lsp2[0])
[slope
array(-0.1875)
, intercept
array(1.07485055923e-05)
   ]
```

The “grower” function

- **grower** is an unusual function that grows 2 variables to their combined largest shape by replicating data values in any dimension required:

grower(x, y, singleton*=0)

```
>>> a.shape
(288,)
>>> b.shape
(1, 20, 144, 288)
>>> c,d=grower(a,b)
>>> c.shape
(288, 1, 20, 144)
>>> d.shape
(288, 1, 20, 144)
```

***singletonoption** is 0 or 1 - Default = 0 If singletonoption is set to 1 then an error is raised if one of the dims is not a singleton dimension.

The “picker” function (1)

- “picker” allows to select non contiguous values of an axis, for example:
```
>>> mypick=genutil.picker(level=(100,850,200))
>>> picked=var(mypick)
>>> print picked.getLevel()[:]
[ 100., 850., 200.,]
```
- An additional “**match**” keyword can be provided to the picker.

- If “**match**” is set to 1 then all requested values must be present, if set to 0 then non-existent values will be returned with “missing_value” everywhere, if set to -1, then non-existent requested values will be skipped.

The “picker” function (2)

- This **picker** example shows how you can do strange things to your variable very quickly and easily. Suppose you wanted to select a discrete number of latitudes (10°N, 43°N, 86°N and 90°N):

```
>>> import cdms, genutil
>>> var=cdms.open(‘myfile.nc’)(‘myvariable’)
>>> mypick=genutil.picker(latitude=(10, 43, 86, 90))
>>> newvar=var(mypick)
>>> print newvar.getLatitude()[:]
[ 10., 43., 86., 90.,]
```

genutil - “udunits”

- The “**udunits**” module is a python port of the C/Fortran “udunits” conversion package:

```
>>> from genutil import udunits
>>> # print udunits.__doc__ # OR   help(udunits)
>>> from genutil import udunits
>>> myunit=udunits(5.6, "m s**-1")
>>> myunit.to("knot")
udunits(10.8855291577,"knot")
```

- To search units for keyword “meter”:

```
>>> for unit in myunit.known_units():
...     if unit.find("meter")>-1:
...         print unit
```

genutil – the rest

- Other sub-components of genutil:
  - minmax
  - filters
  - statusbar
  - color