Demo document with computer code

HPL

Aug 4, 2015

1 Data file

Suppose we have some data in a file typeset with `!bc dat`:

```
1   #      A          B          C          D          E
2      -0.5253    -0.9315    -0.3427    -0.1613    -0.8472
3      -0.9740    -0.2558    -0.5622    -0.7635    -0.0914
4       0.9216     0.7702    -0.4818     0.2155     0.2967
```

2 Complete program and terminal output

The following program (which breaks a page) reads the data in the file and performs analysis (typeset with `!bc pypro`):

```python
1    #!/usr/bin/env python
2
3    import numpy as np
4
5    def readfile(filename):
6        """Read tabular data from file and return as numpy array."""
7        f = open(filename, 'r')
8        data = []   # list of rows in table
9        for line in f:
10           if line.startswith('#'):
11               continue    # drop comment lines
12           numbers = [float(w) for w in line.split()]
13           data.append(numbers)
14       return np.array(data)
15
```

```
16   def analyze(data):
17       """Return statistical measures of an array data."""
18       return np.mean(data), \
19               np.std(data), \
20               np.corrcoef(data)
21
22   if __name__ == '__main__':
23       data = readfile('mydat.txt')
24       # Treat each column as a variable
25       m, s, c = analyze(data.transpose())
26       print """
27   mean=%f
28   st.dev=%f
29   correlation matrix:
30   %s
31   """ % (m, s, c)
```

The output becomes (typeset with `!bc sys`):

```
1    Terminal> python fileread.py
2
3    mean=-0.006005
4    st.dev=0.583542
5    correlation matrix:
6    [[ 1.          0.0509676   0.52406366  0.20964645  0.1574504 ]
7     [ 0.0509676   1.         -0.30920845 -0.12129049  0.7611538 ]
8     [ 0.52406366 -0.30920845  1.          0.49355806 -0.42263817]
9     [ 0.20964645 -0.12129049  0.49355806  1.         -0.38286589]
10    [ 0.1574504   0.7611538  -0.42263817 -0.38286589  1.        ]]
```

3   Code snippet

Fortran 77 is also sometimes handy (typeset with `!bc fcod`):

```
1           subroutine process(a, n, c, r)
2    C      Return array r = c*a
3           integer n
4           real*8 a(n), c, r(n)
5           integer i
6           do i = 1,n
7              r(i) = c*a(i)
8           end do
9           return
10          end
```