

# Demo document with computer code

HPL

Feb 28, 2016

## 1 Data file

Suppose we have some data in a file. The final result of including this file with `@@@CODE mydat.txt` (which implies a code environment starting with `!bc dat`) looks like this:

#	A	B	C	D	E
	-0.5253	-0.9315	-0.3427	-0.1613	-0.8472
	-0.9740	-0.2558	-0.5622	-0.7635	-0.0914
	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`):

```
1  import numpy as np
   def readfile(filename):
4     """Read tabular data from file and return as numpy array."""
       f = open(filename, 'r')
       data = [] # list of rows in table
7     for line in f:
           if line.startswith('#'):
               continue # drop comment lines
10          numbers = [float(w) for w in line.split()]
              data.append(numbers)
           return np.array(data)
13
   def analyze(data):
       """Return statistical measures of an array data."""
16     return np.mean(data), \
```

```

        np.std(data), \
        np.corrcoef(data)
19
if __name__ == '__main__':
    data = readfile('mydat.txt')
22    # Treat each column as a variable
    m, s, c = analyze(data.transpose())
    print """
25 mean=%f
    st.dev=%f
    correlation matrix:
28 %s
    """ % (m, s, c)

```

The output becomes (typeset with !bc sys):

---

Terminal

---

```

Terminal> python fileread.py

mean=-0.006005
st.dev=0.583542
correlation matrix:
[[ 1.          0.0509676  0.52406366  0.20964645  0.1574504 ]
 [ 0.0509676   1.        -0.30920845 -0.12129049  0.7611538 ]
 [ 0.52406366 -0.30920845  1.         0.49355806 -0.42263817]
 [ 0.20964645 -0.12129049  0.49355806  1.         -0.38286589]
 [ 0.1574504   0.7611538  -0.42263817 -0.38286589  1.          ]]

```

---

### 3 Code snippet

Fortran 77 is also sometimes handy. Snippets in that language are typeset inside !bc fcod environments.

**Fortran code box.**

$$r_i = ca_i, \quad i = 1, \dots, n$$

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