JGOFS data system, UNIX: Beginner's Guide

Looking at Data Objects

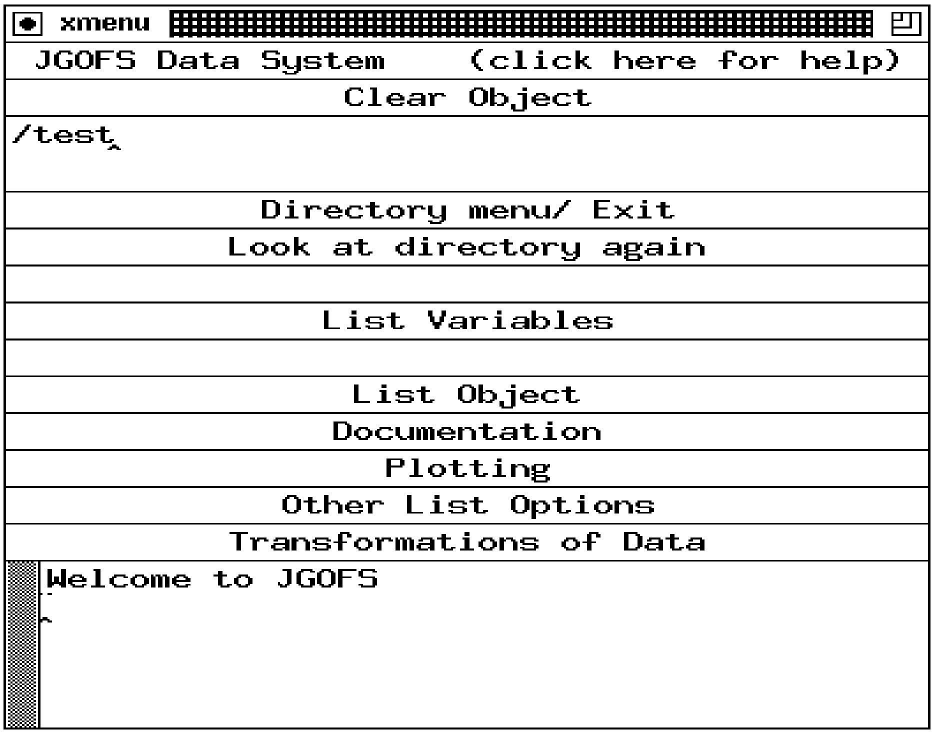
The most important thing to learn about the JGOFS data system is how to name data objects and place restrictions on the information you'd like returned. Once you know this, you can retrieve the data you need from any data object in the system. To illustrate how objects are named, we will use the menu system and a sample data object called `/test`.

Start the menu system by typing

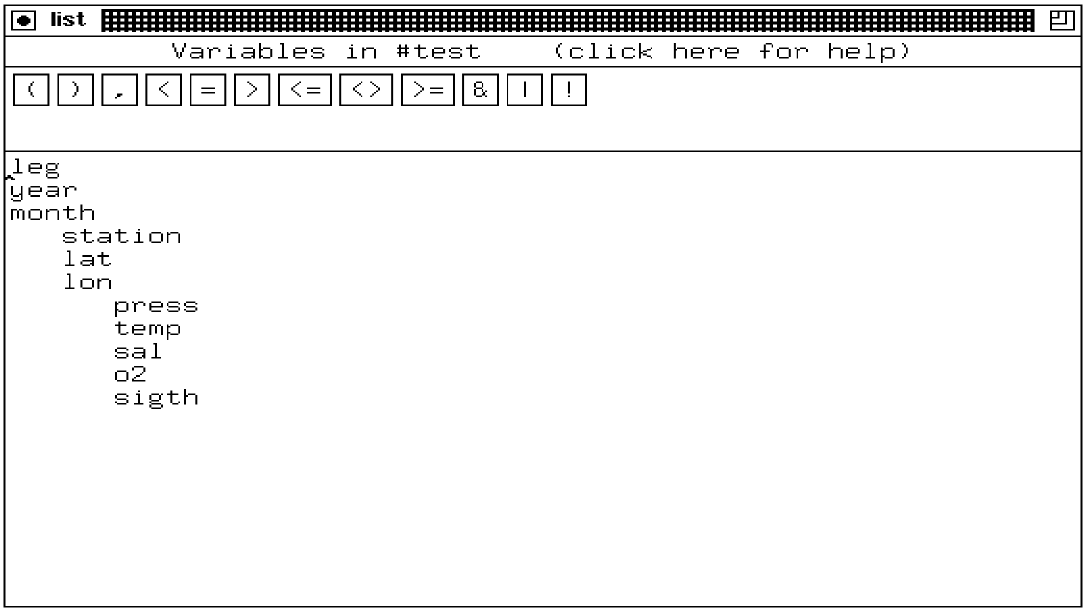
```
source ~jgofs/jgofs
```

which puts up an output window (lower left corner), the main menu window (upper left), a menu for lists (upper right).

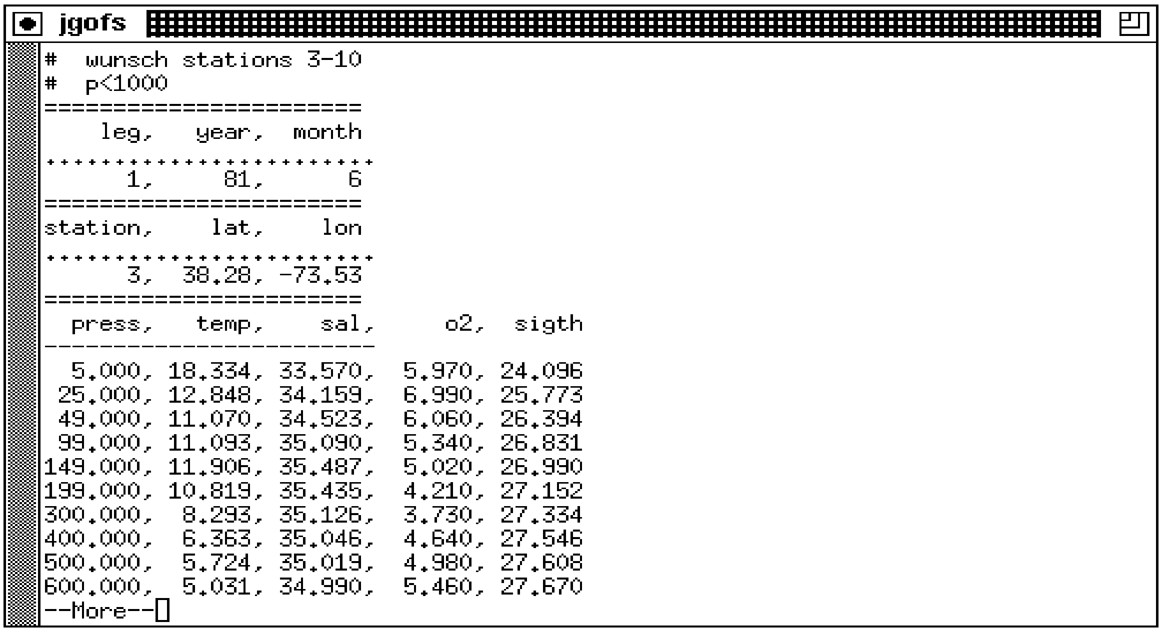
We begin by listing the sample object and various subsets. Click on the "Make listing, plots, ..." box to get into the working menu. (We'll discuss directories later.) Move the mouse cursor to the "object box" (below the "Clear Object" menu item) and type the object name, `/test` (no CR). The menu then looks like Figure 1.



Objects contain variable names and data (and other things such as comments and attributes). To see what variables are contained in the sample object, put the mouse cursor on the "List Variables" item and click the left mouse button. After a pause, the list menu will show the variables (Figure 2). Note the indentation: the data have a hierarchical structure with the most rapidly changing variables in the inner most level.



To list the data, put the mouse cursor on the "List Object" item and click the left mouse button. The listing begins to appear in the output window (figure 3). You now move the mouse down to that window. To continue the listing, hit the space bar (or CR for a single new line); to stop, hit "q" just as in the UNIX "more" program. Note the way the hierarchical structure reflects itself in the output list. At the end of the list, a line of asterisks is printed, and the black bar on the menu item clears. The latter is the visual signal that the commands are complete.



Restrictions

Row Restrictions

Now we are going to illustrate how to place restrictions on the data to be returned. For each of the following, we move the mouse to the object box, edit the name, and then click on the "List Object" item. For example, move to the object box and type the characters `(station=5)` after the object name so that the result looks like

```
/test(station=5)
```

Click on the "List Object" item. Note that we only see the data from the one station requested. Edit the object name to read

```
/test(station=5|station=7)
```

shows the data from station 5 or from station 7. The `|` is the symbol for "or," though you may also use the C convention `||`. Another way of picking information from an object is

```
/test((station=5|station=7)&press<500)
```

The `&` is the symbol for "and," with the other form `&&`. The types of restrictions discussed so far could be called "row restrictions," since, in the simple case of a table, this indicates that only certain rows will be returned. In database language, it is a "selection."

Column Restrictions

We can ask data objects to return only a limited set of their variables. In database language, this is called a "projection." For example,

```
/test(station,lat,lon)
```

Shows only selected variables. Since these are all header type variables, the listing shows none of the profile data (and is much faster).

```
/test(station,press,o2)
```

returns selected header and profile data.

```
/test(station=5,press,o2)
```

This illustrates the combination of row and column restrictions. Note that the variable "station" will not be returned as part of the output, since it is not in the list.

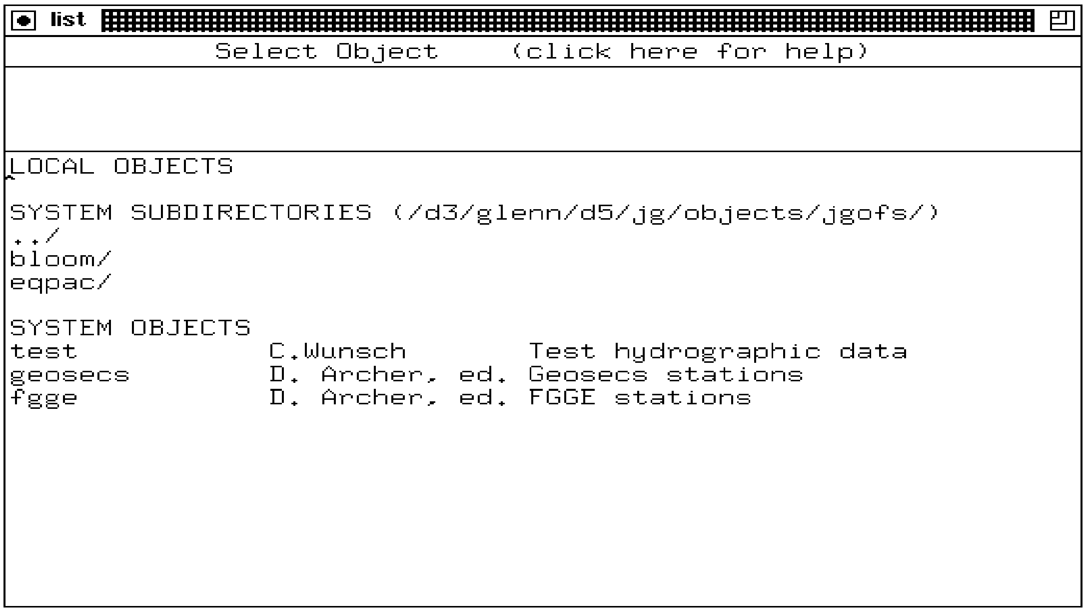
```
/test(station=5&press<500,press,o2)
```

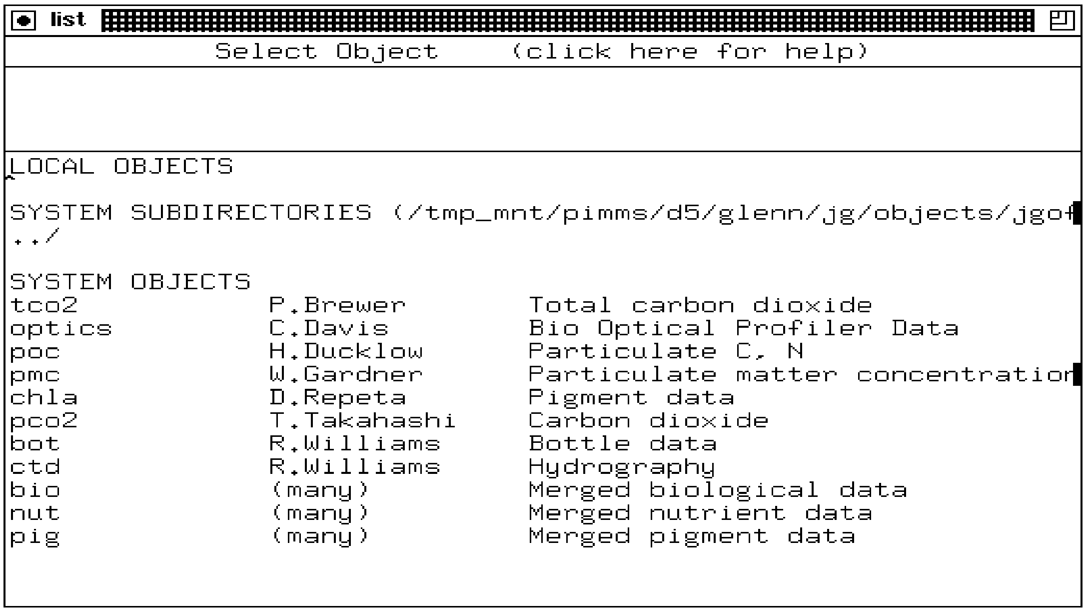
is another example.

All data objects are expected to respond to these two kinds of requests.

Object file system

Like files, objects are organized into directories. The case we have been looking at is a test file normally in the root of the object directory tree. You can begin exploring the tree by clicking on the "Directory menu/ Exit" item and hitting the "Select object from directory" item. You'll get a listing looking something like Figure 4a. You can move around in the tree by copying a subdirectory name into the object box and clicking on "Change object directory." Try this with the "bloom/" subdirectory. You get a listing such as Figure 4b. If you now copy one of the object names into the object box you can look at the information it contains in the same way as before. (Note: you can select items in the listings area by double-clicking the left button on them or by holding down the left button and drawing through them. You can copy them to the object box by moving the mouse into the object box and clicking the middle button.)





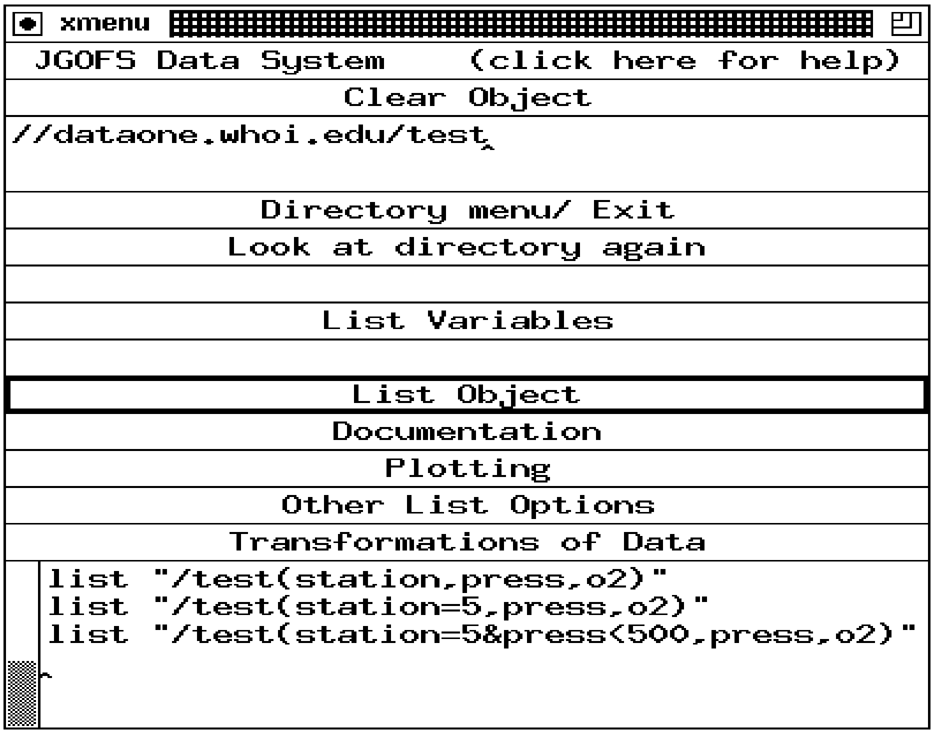
Remote Objects

Data on remote systems is accessed in exactly the same way. Many of the objects listed in the dictionary (e.g., `bloom/bot`) are not stored on your machine. To name a remote object directly, you precede the method name by the internet name of the machine. For example, enter the object name

```
//dataone.whoi.edu/test
```

and hit “List Object” (figure 5). Restrictions are added in the same way, e.g.

```
//dataone.whoi.edu/test(station=5&press<500,press,o2)
```



Constructed Objects

For some methods, one or more of the parameters are the names of objects, rather than files or variable restrictions. These methods act like functions applied to other functions. We call the resulting object a "constructed object." For example, the method `dyn` computes potential temperature, density, specific volume anomaly, and dynamic height from a data object containing press, temp, and sal. Modify the object name to read

```
dyn(/test)
```

and list. The method `dyn` adds new columns and computes their values as the data flows through it.

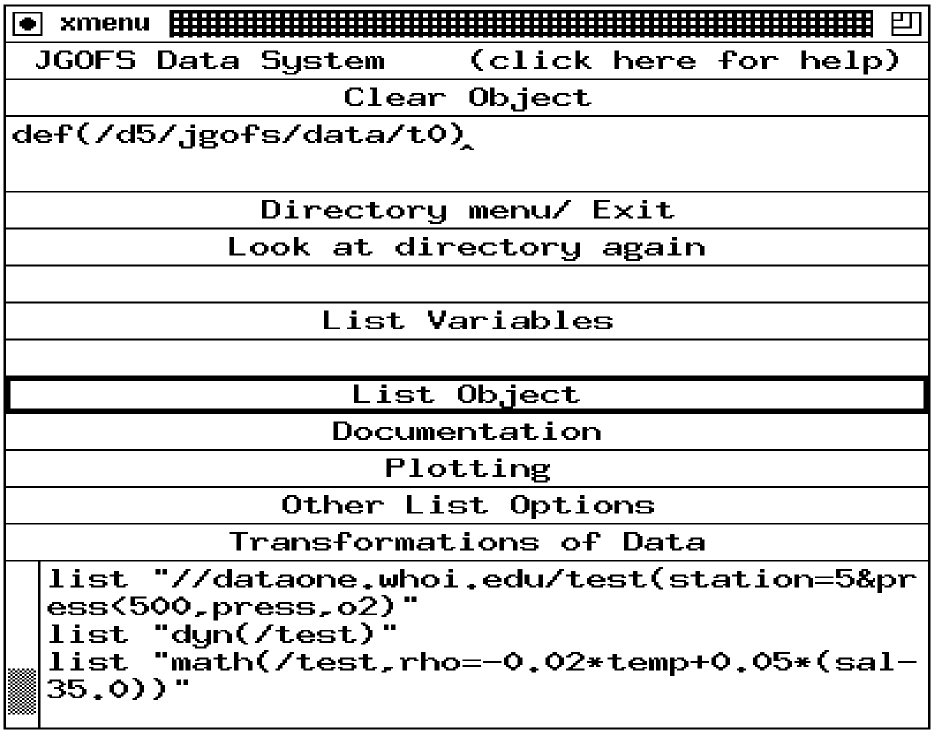
Another example is the `math` object, which allows you to make transformations of the data. As an example, try

```
math(/test,rho= −0.02*temp+0.05*(sal-35.0))
```

which computes a density–like quantity. The math object handles the usual operators (=, +, −, *, /,^ , (,), sin, cos, ln, log (base ten), exp, int).

Object names — methods and parameters

We can use a more general object naming convention to list objects which are not in the dictionary; the lookup in the dictionary actually results in the name (e.g., `/test`) being translated into this more general form (in this case, `def(/d5/jgofs/data/t0)`; however, on your system, the `/d5` will reflect the actual path to the home directory of the jgofs user). The `def` names the executable program — the "method" — which interprets the data. The first parameter, in this case, `/d5/jgofs/data/t0` points to the particular file that `def` is supposed to use. Other parameters are used to place restrictions on the object. To see how this form works, clear the object name and type into the object box `def(/d5/jgofs/data/t0)` except substituting the appropriate path for the `/d5`. See figure 6. List this object; you see that it is the same as `/test`. The equivalent names for the examples of row and column restriction above are




```
def(/d5/jgofs/data/t0,station=5)
def(/d5/jgofs/data/t0,station=5|station=7)
def(/d5/jgofs/data/t0,(station=5|station=7)&press<500)
def(/d5/jgofs/data/t0,station,lat,lon)
def(/d5/jgofs/data/t0,station,press,o2)
def(/d5/jgofs/data/t0,station=5,press,o2)
def(/d5/jgofs/data/t0,station=5&press<500,press,o2)
```

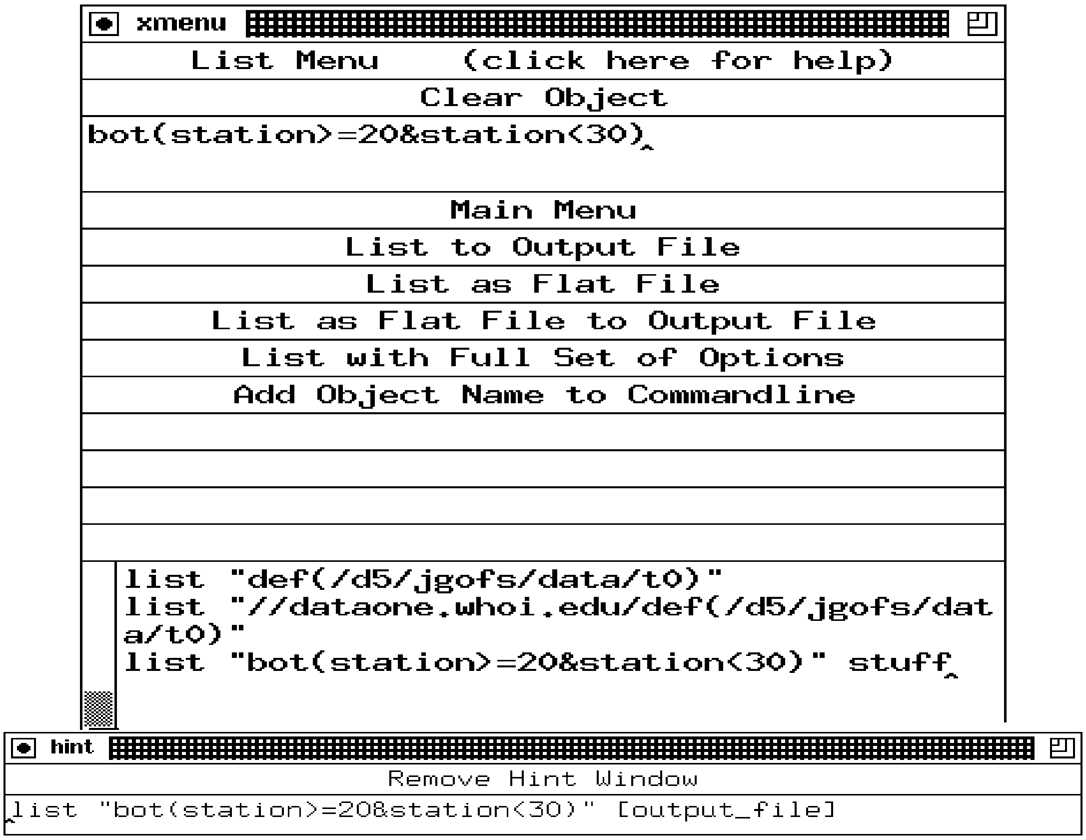
As another example, let's create a file of output from one object and then use that file, rather than the original object, for further processing. Look at the directory again, and copy the "bot" object into the object box. Edit it to read

```
bot(station>=20&station<30)
```

Hit the "Other List Options" item and then select "List to Output File" item. At this point, the line

```
list "bot(station>=20&station<30)"
```

appears in the *command history* area, along with a box showing the form the command takes. (You may have to click on the scroll bar left of the command area to see the last line.) Move the mouse to the *command history* area and type a space followed by the output filename, e.g. `stuff`. Then hit CR (figure 7). Now you have an ordinary file with the output of "list" in it; you can view it by typing



```
cat stuff
```

in the command history window. To examine it with the data system, we use the method `lm` which handles the output from "list." Thus, if we return to the "Main Menu," clear the object name, and type

```
lm(stuff,station,cast,press,theta,o2)
```

and then hit "List Object" we see the reduced data object (figure 8). This illustrates the ability of the jgofs data system to handle many different data formats; all that is required is a suitable "method" which interprets the data.