

Calling Fortran Subroutines in Splus on a Unix Machine

Most students will use R on their PC machines for computing, but Splus is still available on our Unix machines. This is a handout that was used back when Splus on Unix was the standard platform for our department. The instructions are still relevant.

We will learn how to use Fortran (or C) subroutines in Splus. I don't know C but specific help on running C subroutines can be found by typing **help(.C)**. Let's create a Fortran subroutine first. This particular subroutine computes the sample mean and standard deviation using an efficient algorithm. Open a new file called "cp1.f" and enter the following Fortran commands (be sure not to enter commands before column 7 except for the numbered command line which should start in column 3):

```
      subroutine mands(n,x,dm,s)
      implicit double precision(a-h,o-z)
      dimension x(n)
      dm=x(1)
      ds=0.d0
      do 100 i=1,n
      dmold=dm
      dm=dm+(x(i)-dm)/i
      s=s+(x(i)-dmold)*(x(i)-dm)
100  continue
      s=sqrt(s/(n-1))
      return
      end
```

If you don't have a Makefile (a file containing commands that "make" executable or object code), open a file called "Makefile" and enter the following command (make sure to press the Tab key after the ":"):

```
mands:  mands.o
```

Now type **make mands** to create the object code **mands.o**. If we had used Makefile to create executable code, it would have failed because no main program had been created. We could have included more than one subroutine in **mands.f**. We now need to dynamically load the object code. Enter Splus and type the command:

```
dyn.load("/dir/mands.o")
```

where **dir** refers to the directory path for **mands.o**. If you will be using this

object code frequently, place this command in your **.First** function.

Generate some random data and call the Fortran routine:

```
x<-rnorm(100)
mean.f<-0.0
stan.f<-0.0
mands.out<-.Fortran("mands",length(x),as.double(x),as.double(mean.f),
  as.double(stan.f))}
```

The first argument is the name of the subroutine. The calling arguments should match the Fortran declarations exactly; this is the only time Splus will explicitly refer to double precision variables as a type. All arguments must be initialized. Fortran will return a list (to be stored in `mands.out`) that you can then examine. Within the department, we have encountered some difficulty when dynamically loading complex Fortran subroutines.