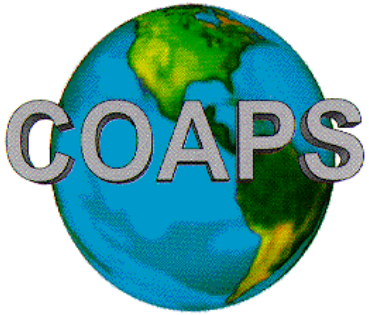

MET3220C & MET6480
Computational Statistics

Programming – week #2

Dr. Mark Bourassa

TA: David Rock, LOV305A, 644-8577
Office Hours: Tue. 1-2PM & Wed. 2:30-3:30
drock@met.fsu.edu

First Programming Assignment

- Goals: 1) Write your own program (with a lot of guidance)
  2) Learn how to use (call) a function
  3) Play with a random number generator
  4) Test the accuracy of the law of large numbers

- Recall the Law of Large Numbers
  - $\Pr\{|a / n - \Pr\{E\}| < \varepsilon\} \rightarrow 0$ as $n \rightarrow \infty$,
  - Where *E* is an event, *a* is the number of occurrences, *n* is the number of opportunities, and ε is a arbitrarily small number.

- The word arbitrarily seems OK in theory, but let's quantify it.
  - We will see just how big ε is, and
  - We will see if it is uniform
    - given a uniform random number generator)

Assignment #1: Enter, Compile, and Run a Simple Program

- 1) Change directories to that directory
  - cd MET3320
- 2) Open a new file called AS2_your_last_name.f90, where 'your_last_name' is replaced by your last name.
  - emacs AS2_your_last_name.f90
- 3) Enter the following program into that file.
- 4) Attempt to compile the program
  - f90 AS2_your_last_name.f90 –o AS2_your_last_name
  - Debug until it compiles.
- 5) Run the program
  - AS1_your_last_name
- 6) Comment the steps in the code – Explain what it is doing. This is the bulk of your grade. Answer the questions that are asked under purpose. The location of your answer should be just after the comments near the top of the code.
- 7) Email the working source code (AS2_your_last_name.f90) to the TA (drock@met.fsu.edu).
- Due date: **Monday**, Jan. 23, before 5:00PM (local time)

The Program As Seen In emacs

- Variables, including arrays are declared.
- Note that three arrays have two dimensions.
- One of these dimensions is the probability that is being tested, and the other is the sample size

emacs@huey.met.fsu.edu

Buffers Files Tools Edit Search Mule F90 Help

```
PROGRAM as2_your_last_name


! Purpose: test the law of large numbers.
!   1) What kind of estimate are we going to get for various size data sets?
!   2) Is the accuracy uniform?
! We wil test these ideas with a random number generator. It produces
! pseudorandom numbers, uniformly, between 0.0 and 1.0. We will test how well
! the law of large numbers works for probabilities of 0.1, 0.2, 0.3,...1.0.


IMPLICIT NONE

INTEGER i, j, i, i_repeats, k, n_repeats
INTEGER, dimension(4) :: sample_size = (/ 10, 100, 1000, 10000 /)
REAL :: rand_out, test_prob
REAL, dimension(10,5) :: mean_error, error, est_prob

n_repeats = 100
mean_error = 0.0

DO i = 1, 4

  DO i_repeats = 1, n_repeats

    est_prob = 0.0

    DO j = 1, sample_size( i )

      call random_number( rand_out )
      DO k = 1, 10
        test_prob = 0.1 * REAL( k )
        if ( rand_out .LT. test_prob ) est_prob(k,i) = est_prob(k,i) + 1
      ENDDO
```

The Program As Seen In emacs

- Assign the number of times that and the Law of Large Numbers will be tested for each probability
- Initialize every value in an array to zero.

```
IMPLICIT NONE

INTEGER i, j, i, i_repeats, k, n_repeats
INTEGER, dimension(4) :: sample_size = (/ 10, 100, 1000, 10000 /)
REAL :: rand_out, test_prob
REAL, dimension(10,5) :: mean_error, error, est_prob

n_repeats = 100
mean_error = 0.0

DO i = 1, 4

  DO i_repeats = 1, n_repeats

    est_prob = 0.0

    DO j = 1, sample_size( i )

      call random_number( rand_out )
      DO k = 1, 10
        test_prob = 0.1 * REAL( k )
        if ( rand_out .LT. test_prob ) est_prob(k,i) = est_prob(k,i) + 1
      ENDDO

    ENDDO

    DO k = 1, 10
      est_prob(k,i) = est_prob(k,i) / sample_size(i)
      error(k,i) = ABS( est_prob(k,i) - 0.1 * REAL( k ) )
    ENDDO

    mean_error(k,i) = mean_error(k,i) + error(k,i)

  ENDDO
  mean_error(k,i) = mean_error(k,i) / REAL( n_repeats )
```

The Program As Seen In emacs

- The command CALL runs a subroutine.
- In this case, the argument (the variable in brackets) is output by the subroutine.
- It is a random number with a value between 0.0 and 1.0

```
IMPLICIT NONE

INTEGER i, j, i, i_repeats, k, n_repeats
INTEGER, dimension(4) :: sample_size = (/ 10, 100, 1000, 10000 /)
REAL :: rand_out, test_prob
REAL, dimension(10,5) :: mean_error, error, est_prob

n_repeats = 100
mean_error = 0.0

DO i = 1, 4

  DO i_repeats = 1, n_repeats

    est_prob = 0.0

    DO j = 1, sample_size( i )

      call random_number( rand_out )
      DO k = 1, 10
        test_prob = 0.1 * REAL( k )
        if ( rand_out .LT. test_prob ) est_prob(k,i) = est_prob(k,i) + 1
      ENDDO

    ENDDO

    DO k = 1, 10
      est_prob(k,i) = est_prob(k,i) / sample_size(i)
      error(k,i) = ABS( est_prob(k,i) - 0.1 * REAL( k ) )
    ENDDO

    mean_error(k,i) = mean_error(k,i) + error(k,i)

  ENDDO
  mean_error(k,i) = mean_error(k,i) / REAL( n_repeats )
```

The Program As Seen In emacs

- The ABS command takes the absolute value of the argument.

```
IMPLICIT NONE

INTEGER i, j, i, i_repeats, k, n_repeats
INTEGER, dimension(4) :: sample_size = (/ 10, 100, 1000, 10000 /)
REAL :: rand_out, test_prob
REAL, dimension(10,5) :: mean_error, error, est_prob

n_repeats = 100
mean_error = 0.0

DO i = 1, 4

  DO i_repeats = 1, n_repeats

    est_prob = 0.0

    DO j = 1, sample_size( i )

      call random_number( rand_out )
      DO k = 1, 10
        test_prob = 0.1 * REAL( k )
        if ( rand_out .LT. test_prob ) est_prob(k,i) = est_prob(k,i) + 1
      ENDDO

    ENDDO

    DO k = 1, 10
      est_prob(k,i) = est_prob(k,i) / sample_size(i)
      error(k,i) = ABS( est_prob(k,i) - 0.1 * REAL( k ) )
    ENDDO

    mean_error(k,i) = mean_error(k,i) + error(k,i)

  ENDDO
  mean_error(k,i) = mean_error(k,i) / REAL( n_repeats )
```

The Program As Seen In emacs

- The WRITE command allows the output to be formatted.
- The first argument describes where the output goes. 6 means to the screen.
- The characters in quotes indicates the format.

```
   DO i_repeats = 1, n_repeats

     est_prob = 0.0

     DO j = 1, sample_size( i )

       call random_number( rand_out )
       DO k = 1, 10
         test_prob = 0.1 * REAL( k )
         if ( rand_out .LT. test_prob ) est_prob(k,i) = est_prob(k,i) + 1
       ENDDO

     ENDDO

     DO k = 1, 10
       est_prob(k,i) = est_prob(k,i) / sample_size(i)
       error(k,i) = ABS( est_prob(k,i) - 0.1 * REAL( k ) )
     ENDDO

     mean_error(k,i) = mean_error(k,i) + error(k,i)

   ENDDO
   mean_error(k,i) = mean_error(k,i) / REAL( n_repeats )

   write(6,'(10(F6.4,1X))') (error(k,i),k=1,10)
   print*, ' '


ENDDO

END
 
--:--  as2.f90          (F90)--L55--All----------------------------------
menu-bar edit copy
```

Formatting

- Consider F6.4
  - F means a floating point value (real number)
  - The 6 means 6 digits of output, and the 4 means that 4 digits will be after the decimal
- Consider 1X
  - X means write a blank space,
  - The number before the X is the number of blank space
- Consider 10(F6.4,1X)
  - 10( ) means do the stuff in brackets 10 time
  - In this case, a real value and a space.