Subject: Re: BEGINNER ASKS FOR HELP!!!
Posted by davidf on Sat, 31 Jan 1998 08:00:00 GMT
View Forum Message <> Reply to Message

"Edoardo Marcora" <marcora@colorado.edu> writes:

> I need to create a RANDOM square matrix that contains only three numbers 0,
> 1, and (-1). There are two constraints though. I would like to tell the
> program before it creates this matrix what the (at least approximate) RATIO
> of 1 / (-1) should be and HOW MANY (at least approximately) 1 and (-1)
> (total) are on each row of the matrix.
>
> For example, the input by the user would be:
>
> NUMBER OF ELEMENTS PER ROW e.g., 6
>
> RATIO OF 1 / (-1) e.g., 0.5 (at the level of the matrix not of the row)
>
> NUMBER OF 1 AND (-1) (total) IN EACH ROW e.g., 3 (for example 0 0 0 1 1 -1)
>
> I do not need the ratio and number of |1| per row to be exact, also an
> approximation it's good.

Uh, Lord knows I'm no statistician, but doesn't this smell
just a *wee* bit suspicious to you?

Put here you go. You can tell this little routine the ratio
of 1s to -1s, and the density of the 1/-1 distribution. For
example if I want the ratio of 1/-1 to be 0.5 and I want
40 percent (approximately) of the locations in the 10 by 10
array to be filled, I call the routine like this:

```
  IDL> Random_Fill, 0.5, 0.40
```

The routine fills random locations in the array.

I can tell you that (on average) about 4 of the elements
in any particular row will be filled. (It will actually
be a little less, because my 1 and -1 locations can overlap.)
But I leave it up to you to iron out the problems. This is just
food for thought.

Cheers,

David

```
 ********************************************************** *****
PRO Random_Fill, ratio, density
```

```
   ; Ratio = pluses/minuses. Should be a float.
   ; Density = Percent of total squares occupied (e.g., 0.20)

On_Error, 1

   ; Check parameter values.
   ; Assign defaults if necessary.

CASE N_Params() OF
   0: BEGIN
      ratio = 1.0
      density = .40
      END
   1: BEGIN
      density = .40
      END
   ELSE:
ENDCASE

IF density LT 0.0 OR density GT 0.99 THEN $
   Message,'Illegal density value.'
ratio = Float(ratio)
density = Fix(density * 100)

   ; How many 1s and -1s ?

minus = Fix(density/(ratio + 1))
plus = density - minus

   ; Find random positions for 1s and -1s.

arrayPlus = Floor(RandomU(seed, plus) * 100)
arrayMinus = Floor(RandomU(seed, minus) * 100)
array = IntArr(100)

   ; Fill array with 1s.

array(arrayPlus) = 1

   ; Fill array with -1s. If the cell is already full,
   ; flip a coin to see if a +1 or -1 should be entered.

FOR j=0,N_Elements(arrayMinus)-1 DO BEGIN
   IF array(arrayMinus[j]) EQ 1 THEN BEGIN
      flip = Randomu(seed, 1)
      IF flip[0] LT 0.5 THEN array(arrayMinus[j]) = -1
   ENDIF ELSE array(arrayMinus[j]) = -1
```

```
ENDFOR

   ; Reformat to a 10-by-10 array.

array = Reform(array, 10, 10)
Print, array
Print, 'Number of 1s: ', plus
Print, 'Number of -1s:', minus
a = Where(array EQ 1 OR array EQ -1, count)
Print, 'Number of non-zero values in array: ', count
END
```

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: BEGINNER ASKS FOR HELP!!!
Posted by Dr. G. Scott Lett on Sat, 31 Jan 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Edoardo "Dado" Marcora wrote:

> Dear Sirs,
>
> I've just started using IDL 5 and I would like to solve this problem I've
> encountered.
>
> I need to create a RANDOM square matrix that contains only three numbers 0,
> 1, and (-1). There are two constraints though. I would like to tell the
> program before it creates this matrix what the (at least approximate) RATIO
> of 1 / (-1) should be and HOW MANY (at least approximately) 1 and (-1)
> (total) are on each row of the matrix.
>
> For example, the input by the user would be:
>
> NUMBER OF ELEMENTS PER ROW e.g., 6
>
> RATIO OF 1 / (-1) e.g., 0.5 (at the level of the matrix not of the row)
>
> NUMBER OF 1 AND (-1) (total) IN EACH ROW e.g., 3 (for example 0 0 0 1 1 -1)
>
> I do not need the ratio and number of |1| per row to be exact, also an
> approximation it's good.
>

> Thanx for your help,
>
> Dado

Dear Dado,
This isn't the entire function you want, but it should help you write one.

```
A = randomu( seed, 6, 6 )
high = float( 3 ) /6
low = high * 0.5
neg = where( A lt low, countLow )
pos = where( A ge low and A lt high, countHigh )
B = intarr( 6, 6 )
if countLow gt 0 then B[ neg ] = -1
if countHigh gt 0 then B[ pos ] = 1
```

--
==========
Dr. G. Scott Lett
slett@holisticmath.com
http://holisticmath.com/
==========

Subject: Re: BEGINNER ASKS FOR HELP!!!
Posted by steinhh on Mon, 02 Feb 1998 08:00:00 GMT
View Forum Message <> Reply to Message

In article <6at4ri$21e@lace.colorado.edu> "Edoardo \"Dado\" Marcora"
<marcora@colorado.edu> writes:

> For example, the input by the user would be:
>
> NUMBER OF ELEMENTS PER ROW e.g., 6
>
> RATIO OF 1 / (-1) e.g., 0.5 (at the level of the matrix not of the row)
>
> NUMBER OF 1 AND (-1) (total) IN EACH ROW e.g., 3 (for example 0 0 0 1 1 -1)
>
> I do not need the ratio and number of |1| per row to be exact, also an
> approximation it's good.

One way of solving this is to make a row with the specified

number of ones/zeroes/minus ones, e.g., [0,0,1,1,-1], and
then *ordering* those elements in a random way, which is
easily done by sorting a parallell, random array.

I.e.,

```
ELEMENTS = [0,0,0,1,1,-1]
N = N_ELEMENTS(ELEMENTS)
FOR ROW_NO=0,N_ROWS DO BEGIN
  DATA(*, ROW_NO) = ELEMENTS[SORT(RANDOMU(SEED,N))]
END
```

This will result in a fixed number of elements of each kind,
but randomly ordered.

If you'd like the number of ones/zeroes/minus ones to vary,
with probabilities according to the specifications,
another approach is better. First, we need to calculate
the probabilities of each "outcome".

Given N (number of elements per row), R (ratio of the number
of ones to minus ones), and NONZ (number of nonzero
elements), the probability of a single element being nonzero
will be:

```
P_NONZ = NONZ/FLOAT(N)  ; Avoid integer arithmetic
```

The probability of an element being zero will (of course) be:

```
P_ZERO = 1-P_NONZ
```

The probabilities of an element being either plus or minus
one of course add up to P_NONZ: P_PLUS + P_MINUS = P_NONZ.
We also have P_PLUS/P_MINUS = R, which gives:

```
P_PLUS = P_NONZ * R/(1.0+R)
P_MINUS = P_NONZ * 1.0/(1.0+R)
```

Now, if I generate a number that is uniformly distributed
between 0 (inclusive) and 1, the probability of that number
being between zero and P_ZERO is exactly P_ZERO. The
probability of the nuber being between P_ZERO and
P_ZERO+P_PLUS is exactly P_PLUS, and the probability of the
number being between P_ZERO+P_PLUS and 1 is P_MINUS (since
P_ZERO, P_PLUS and P_MINUS add up to one).

Thus,

```
  ARRAY = RANDOMU(SEED,N,N_ROWS)
  ZMASK = ROW LT P_ZERO
  PMASK = ROW GE P_ZERO AND ROW LT P_ZERO+P_PLUS
  MMASK = ROW GE P_ZERO+P_PLUS
  ROW = 0*ZMASK + 1*PMASK + (-1)*MMASK
```

..should do it, I think. Of course that "0*ZMASK" part is not necessary, it's only there for clarity.

Disclaimers about typos etc. apply, of course...

Stein Vidar
