
Subject: Re: Applying scalar functions to arrays
Posted by [Martin Schultz](#) on Fri, 05 Dec 1997 08:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

This is a multi-part message in MIME format.

-----167E2781446B

Content-Type: text/plain; charset=iso-8859-1

Content-Transfer-Encoding: 8bit

>
> How can you apply an IDL scalar function to each element in an
> array, without using for loops?
>
> Suppose we have the following array:
>
> IDL> a=[0.05,0.01,0.001]
>
> [...]
> But how about chi-square quantiles...?
> IDL> print,chisqr_cvf(a,17)
> % Expression must be a scalar in this context: <BYTE Array[3]>.
>
> The manual instructs you to avoid for loops when operating on
> arrays, but doesn't tell you how! Can anyone help?
>
> Niels
>

I don't think there is any way to avoid a loop in this case. I just
tried to compose a wrapper routine that would allow to call *any*
function in a loop - the result is attached below. Use it as
 print,loop("chisqr_cvf",a,17)
and you will see:

 27.5871 33.4087 40.7903

Note for serious IDL addicts: I tried to add the _EXTRA=e keyword
option, but in this case chisqr_cvf would again complain that the number
of parameters wasn't right, although no keywords were provided. That's
sad :-)

Hope this helps somewhat,
Martin.

Dr. Martin Schultz

Department for Earth&Planetary Sciences, Harvard University
186 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318

fax : (617)-495-4551

e-mail: mgs@io.harvard.edu

IDL-homepage: <http://www-as.harvard.edu/people/staff/mgs/idl/>

-----167E2781446B

Content-Type: text/plain; charset=us-ascii; name="loop.pro"

Content-Transfer-Encoding: 7bit

Content-Disposition: inline; filename="loop.pro"

```
;----- --
;+
; NAME:
;   LOOP
;
; PURPOSE:
;   This routine provides a wrapper for function calls that accept
;   only scalars so that they can operate on arrays.
;
; CATEGORY:
;   serious stuff
;
; CALLING SEQUENCE:
;   result = LOOP(name,arg,p1,p2,p3,p4)
;
; INPUTS:
;   NAME --> the name of the function (string)
;
;   ARG --> the argument (array)
;
;   P1 .. P4 --> optional function parameters
;
; KEYWORD PARAMETERS:
;   unfortunately none. Would be nice if _EXTRA would work.
;
; OUTPUTS:
;   a result *vector* with the same number of elements as arg.
;
; SUBROUTINES:
;
; REQUIREMENTS:
;
; NOTES:
```

```

;
;
; EXAMPLE:
;   a=[0.05,0.01,0.001]
;   print,loop("chisqr_cvf",a,17)
;
;   IDL prints:
;   27.5871    33.4087    40.7903
;
; MODIFICATION HISTORY:
;   mgs, 05 Dec 1997: VERSION 1.00
;
;
;-
; Copyright (C) 1997, Martin Schultz, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to mgs@io.harvard.edu
; with subject "IDL routine loop"
;-----

```

```
function loop,name,arg,p1,p2,p3,p4
```

```
    on_error,2 ; return to caller
```

```
    result = fltarr(n_elements(arg))
```

```

; print,n_elements(p1),n_elements(p2),n_elements(p3),n_elements(p4)
; call function with number of parameters supplied
; not very elegant but safe.
; (would probably look nicer with EXECUTE, but isn't that slower ?)

```

```

if (n_elements(p4) gt 0) then begin
    for i=0,n_elements(arg)-1 do $
        result(i) = call_function(name,arg(i),p1,p2,p3,p4)
    return,result
endif

```

```

if (n_elements(p3) gt 0) then begin
    for i=0,n_elements(arg)-1 do $
        result(i) = call_function(name,arg(i),p1,p2,p3)
    return,result
endif

```

```
if (n_elements(p2) gt 0) then begin
  for i=0,n_elements(arg)-1 do $
    result(i) = call_function(name,arg(i),p1,p2)
  return,result
endif
```

```
if (n_elements(p1) gt 0) then begin
  for i=0,n_elements(arg)-1 do $
    result(i) = call_function(name,arg(i),p1)
  return,result
endif
```

```
for i=0,n_elements(arg)-1 do $
  result(i) = call_function(name,arg(i))
return,result
```

```
end
```

```
-----167E2781446B--
```
