Subject: Re: mean() function
Posted by Foldy Lajos on Mon, 16 Jan 2006 10:27:34 GMT
View Forum Message <> Reply to Message

Hi,

one small correction: instead of n_elements(...) you should use
n_elements(...)-1.

This approach has an other problem: the elements at the beginning of the
array are divided ~n_elements() times, which can introduce large rounding
errors.

Example:

```
IDL> print, !version
print, !version
{ x86 linux unix linux 6.2 Jun 20 2005      32      64}

IDL> a=fltarr(10000000l)
IDL> a[0]=1.0e7
IDL> print, alt_mean(a)
   1.02190
```

vs.

```
IDL> print, mean(a)
% Compiled module: MEAN.
% Compiled module: MOMENT.
   1.00000
```

there is no golden way :-)))

regards,
lajos

On Mon, 16 Jan 2006, Maarten wrote:

```
>  I think that this comes close. I ignores infinite numbers on request.
>  Is it fast: no. But implementing the thing is C should be near trivial
>  if you have dealt with that before (I haven't, at least not in IDL).
>
>  function alt_mean, D, nan=nan
>   compile_opt defint32, strictarr, logical_predicate, strictarrsubs
>
>   M = 0.0
```

```
>
>   if keyword_set(nan) then begin
>    idx = where(finite(D), cnt)
>    if cnt gt 0 then begin
>     for  ii=0,n_elements(idx) do $
>      M += (D[idx[ii]] - M)/(ii+1)
>    endif else begin
>     M = !values.d_nan
>    endelse
>   endif else begin
>    for  ii=0,n_elements(D) do $
>     M += (D[ii] - M)/(ii+1)
>   endelse
>
>   return, M
>  end
>
>
```