Subject: Re: percentile with dimension keyword
Posted by ben.bighair on Tue, 19 Jul 2011 13:42:27 GMT
View Forum Message <> Reply to Message

On 7/18/11 9:31 PM, JP wrote:
> Dear IDLers,
>
> The MEDIAN (i.e. 50th percentile) function has a dimension keyword.
> I need to calculate other percentiles (e.g. 5th, 95th) in a large 3D array and don't want to loop.
> Does anyone know of a function that could do that in a similar way to MEDIAN?

Hi,

Since the median operates in a rank like way, I wonder if you could simple sort the original and determine the quantiles you want by relative order. Perhaps like this?

```
x = randomn(s, 3,4,5)  ; some array
ix = sort(x)  ; it's ascending order indices
sx = x[ix]  ; the original sorted
n = size(ix,/N_ELEMENTS)  ; the number of items
p = [0.05, 0.5, 0.95]  ; your quantiles  (5th, 50th, and 95th)
ip = n * p  ; quantiles as indices into ix
v = sx[ip] ; the quantiles as values

print, "sorted original ", sx
print, "the median ", median(x)
print, "the quantiles 5th 50th and 95th ", v
```

I might have fuzzy thinking on the correct way to handle the indices, but I think it is a good start. The next step is to wrap the above in a function that takes the array, the quantile you specify and the dimension to operate on as arguments.

Now that I think of it, have you searched for a function called percentile.pro? It might be worth looking for that as a starting point.

Cheers,
Ben

Subject: Re: percentile with dimension keyword
Posted by Kim on Tue, 19 Jul 2011 14:34:05 GMT

On Jul 19, 9:42 am, Ben Tupper <ben.bigh...@gmail.com> wrote:
>
> Now that I think of it, have you searched for a function called
> percentile.pro? It might be worth looking for that as a starting point.
>
> Cheers,
> Ben

; PERCENTILE: This Function returns the Values of data Corresponding
to input Percentiles
; NOTE: The Values returned are linear interpolates of the
sorted data
; EXAMPLE: DATA=FINDGEN(1001) & PRINT,
PERCENTILE(DATA,PERCENT=[0,0.01,1,10,50,98,99.9,100.0])

```
FUNCTION PERCENTILE,DATA,PERCENT=PERCENT,MISSING=MISSING,ERROR=error

  IF N_ELEMENTS(PERCENT) EQ 0 THEN PERCENT = FINDGEN(101)
  INDEX  = INTERPOL([0.0,100.0],N_ELEMENTS(DATA))  ; Normalize INDEX
FROM 0 to 100 percent
  VALUES = INTERPOL(DATA(SORT(DATA)) ,INDEX , PERCENT)
  IF N_ELEMENTS(VALUES) EQ 1 THEN RETURN, VALUES(0) ELSE RETURN,
VALUES

END
```

Subject: Re: percentile with dimension keyword
Posted by JP on Wed, 20 Jul 2011 04:53:48 GMT

thanks,

What I am after is a function that could be used in an array with 3 dimensions.
like:

```
array = Lindgen(1000,1000,100)
median_array = MEDIAN(array, dimension=3)
```

the result will be a 2d array of 1000x1000

something like that but for any percentile (the example abobe would give the 50th percentile)

thanks

JP

Subject: Re: percentile with dimension keyword
Posted by ben.bighair on Wed, 20 Jul 2011 14:03:04 GMT
View Forum Message <> Reply to Message

On 7/20/11 12:53 AM, JP wrote:
> thanks,
>
> What I am after is a function that could be used in an array with 3 dimensions.
> like:
>
> array = Lindgen(1000,1000,100)
> median_array = MEDIAN(array, dimension=3)
>
> the result will be a 2d array of 1000x1000
>
> something like that but for any percentile (the example abobe would give the 50th percentile)
>
> thanks
>
> JP

Hi again,

I think you could use the PERCENTILE function (provided by Kim) or some variant of it with Craig Markwardt's CMAPPLY function. CMAPPLY accepts a user defined function name as the operation and the dimension over which to apply the operation. You can find it here...

http://www.physics.wisc.edu/~craigm/idl/down/cmapply.pro

Something along the lines of (untested) ...

r = CMAPPLY("USER:PERCENTILE", data, 3, functargs = {PERCENT: 95})

Cheers,
Ben

Subject: Re: percentile with dimension keyword
Posted by JP on Wed, 20 Jul 2011 15:11:44 GMT
View Forum Message <> Reply to Message

thanks Ben, I think that will work.

cheers

JP

Subject: Re: percentile with dimension keyword
Posted by JDS on Wed, 20 Jul 2011 20:00:37 GMT
View Forum Message <> Reply to Message

SORT_ND, plus Ben's suggestion. Sorting for selection is overkill, but that's what's fast. Reminds me of my favorite numerical recipes quote: "Selection is sorting's austere sister."

JD
