Subject: Re: calculating long term statistics on ALBEDO data
Posted by Klaus Scipal on Tue, 05 Apr 2005 14:02:39 GMT
View Forum Message <> Reply to Message

I just noticed that calculating the variance is a bit more tricky than
calculating the mean:

1. Calculate the mean

mean=total(array, 3)

2. Substract the mean from each sample, take the square and sum it up. Maybe
there is a faster way, but the only way which comes to my mind is
replicating the mean array and then do the substraction

dev=total((array-replicate(mean, 360))^2),3)

3. devide dev by n-1

variance=total(array, 3)/(total(finite(array),3)-1)

and of course if you have to deal with missing data don't forget the /NaN
keyword in the call to total

Klaus

"Klaus Scipal" <ks@ipf.tuwien.ac.at> wrote in message
news:425297e0$0$8024$3b214f66@tunews.univie.ac.at...
> Hi Allard
>
> Calculating the mean can simply be done on the entire array using the
> total function
>
> mean=total(array, 3)/360.
>
> This however assumes that there are no missing values in your array. In
> case of missing values your divisor should not be 360 but something
> smaller. You can deal with missing values by setting all the missing
> values to !values.nan and applying following formula:
>
> mean=total(array, 3)/total(finite(array),3)
>
> the total & finite function in the divisor gives you the number of "valid"
> samples for each point.
>
> For the standard deviation the code should look similiar!
>

> cheers
>
> Klaus
>
>
>
> "wita" <allard.dewit@wur.nl> wrote in message
> news:1112701185.279975.281940@o13g2000cwo.googlegroups.com.. .
>> Dear all,
>>
>> I have a dataset of Meteosat derived albedo values over the period
>> 1994-2003 with a temporal frequency of 10 days (36 observations per
>> year). Now I want to calculate some long term
>> statistics on this data such as long term mean and st. deviation per
>> pixel and per 10-days.
>> I am using the ENVI tiling mechanism to read in data as
>> interleaved-by-line. The data has
>> dimensions 1300x825x360 and I am getting chunks of data of size
>> 1300x360 with each call
>> for a new tile.
>>
>> The code I am using to calculate the statistics is this:
>> ----------
>> ;Main processing loop
>> FOR i=0L, num_tiles-1 DO BEGIN
>> envi_report_stat,base, i, num_tiles
>> data = envi_get_tile(tile_id1, i)
>> mask = envi_get_tile(tile_id2, i)
>> ;Only execute at first iterations to get the data dimensions
>> IF i EQ 0 THEN BEGIN
>> ds = SIZE(data, /DIMENSIONS)
>> means = FLTARR(ds[0],36)
>> stdev = means
>> decades = LINDGEN(ds[1]) MOD 36
>> index = WHERE(decades EQ 0)
>> tmpmeans = FLTARR(36)
>> tmpstdevs = FLTARR(36)
>> ENDIF
>> ;Loop over X direction
>> FOR j=0, ds[0]-1 DO BEGIN
>> ;If mask = Land surface then loop over Z dimension
>> IF mask[j] EQ 1 THEN BEGIN
>> tmpdata = REFORM(data[j,*])
>> FOR k=0, 35 DO BEGIN
>> tmpindex = index+k
>> tmpmeans[k] = MEAN(tmpdata[tmpindex], /NAN)
>> tmpstdev[k] = STDDEV(tmpdata[tmpindex], /NAN)
>> ENDFOR

```
>>      ENDIF ELSE BEGIN
>>        tmpmeans = FLTARR(36)
>>        tmpstdev = FLTARR(36)
>>      ENDELSE
>>      means[j,*] = tmpmeans
>>      stdev[j,*] = tmpstdev
>>    ENDFOR
>>    WRITEU, unit1, means
>>    WRITEU, unit2, stdev
>>  ENDFOR
>> ----------------------------------------
>> This is not particularly fast because of the three nested FOR loops.
>> Note that I am using a mask image to determine what is land/sea and I
>> only execute the inner loop over land. I've been trying to speed this
>> up but no success so far. One idea was to use the HIST_ND() function
>> to assemble all data into histograms in order to calculate statistics:
>> ----------------------------------------
>>    IF i EQ 0 THEN BEGIN
>>      ds = SIZE(data, /DIMENSIONS)
>>      decades  = LINDGEN(ds[1]) MOD 36L
>>      decades  = REFORM(decades,1,ds[1])
>>      decades2d = REBIN(decades, ds[0], ds[1])
>>      decades2d = REFORM(decades2d, 1, ds[0], ds[1])
>>      pixels2d  = REBIN(LINDGEN(ds[0]),ds[0], ds[1])
>>      pixels2d  = REFORM(pixels2d, 1, ds[0], ds[1])
>>    ENDIF
>>
>> ;   Add extra data dimension
>>    data = REFORM(data, 1, ds[0], ds[1])
>> ;   Concatenate everything to 1 array and reform it in order to have
>> NxP points
>> ;   with the ALBEDO data on n=0, the image pixel nr in N=1 and the
>> decade in N=2
>>    tmp  = [data, decades2d, pixels2d]
>>    tmp  = REFORM(tmp, 3, ds[0]*ds[1])
>>    r = HIST_ND(tmp, 1)
>> ----------------------------------------
>>
>> But this doesn't solve anything because r becomes a
>> 1300x36x<nrofalbedoclasses> array
>> and I still need to loop 1300x36 times.
>>
>> Has anyone some idea how to vectorise this particular problem?
>>
>> Thanks in advance,
>>
>> Allard
>>
```

>
>