Subject: Re: Mode and variation of cells in multiple grids (3-D problem)
Posted by MarioIncandenza on Tue, 26 Jan 2010 00:49:26 GMT
View Forum Message <> Reply to Message

On Jan 25, 12:47 pm, Ed Hyer <ejh...@gmail.com> wrote:
> 1) The number of different values in the matched grids;
> 2) The mode, that is, the frequency of the most common value.

So, what I ended up with was something like this:

allgrids=[[[grid1]],[[grid2]],etc.]
matchcount=allgrids *0b; initialize
for i1=0,ngrids-1 do for i2=0,ngrids-1 do matchcount[*,*,i1] +=
(allgrids[*,*,i1] eq allgrids[*,*,i2])
magnitude_of_mode = max(matchcount,dim=3,imode)
value_of_mode = allgrids[imode]

So 2 for-loops still needed, but across the shortest dimension.

--Edward H.

Subject: Re: Mode and variation of cells in multiple grids (3-D problem)
Posted by Jeremy Bailin on Tue, 26 Jan 2010 21:56:10 GMT
View Forum Message <> Reply to Message

On Jan 25, 7:49 pm, Ed Hyer <ejh...@gmail.com> wrote:
> On Jan 25, 12:47 pm, Ed Hyer <ejh...@gmail.com> wrote:
>
>> 1) The number of different values in the matched grids;
>> 2) The mode, that is, the frequency of the most common value.
>
> So, what I ended up with was something like this:
>
> allgrids=[[[grid1]],[[grid2]],etc.]
> matchcount=allgrids *0b; initialize
> for i1=0,ngrids-1 do for i2=0,ngrids-1 do matchcount[*,*,i1] +=
> (allgrids[*,*,i1] eq allgrids[*,*,i2])
> magnitude_of_mode = max(matchcount,dim=3,imode)
> value_of_mode = allgrids[imode]
>
> So 2 for-loops still needed, but across the shortest dimension.
>
> --Edward H.

Here's a pure IDL no-loop solution that uses... <drum roll>... the
combination of VALUE_LOCATE and HISTOGRAM! (tell me you didn't see
that coming).

```
; some sample data
grid1 = [[1., 0, -10, 1.],[-5, -10, 9, 8],[0, 1, 2, 3]]
grid2 = [[1., 9, 8, 2],[-5, 0, 2, 3],[0, 1, 2, 3]]
grid3 = [[-10, 8, 9, 2],[-5, -10, 1, 3],[0, 1, 2, 3]]
grid4 = [[0, 9, 8, 2],[-10, 2, 2, 2],[0, 1, 2, 3]]
grid5 = [[1., -2, -20, 2],[-5, 2, 2, 8],[0, 1, 2, 3]]

allgrids = [[[grid1]],[[grid2]],[[grid3]],[[grid4]],[[grid5]]]

gridsize=size(allgrids,/dimen)
npix = gridsize[0]*gridsize[1]
uniqvals = allgrids[uniq(allgrids,sort(allgrids))]
nuniq = n_elements(uniqvals)
; map values into their index in uniqvals so we only need to
; deal with integers
mappedgrids = value_locate(uniqvals, allgrids)
; now we can use histogram!
; first give each pixel its own increment so that the histogram for
; the pixels run from 0..nuniq-1, nuniq..2*nuniq-1, etc.
pixincrement = ((lindgen(gridsize[0]) # replicate(1,gridsize[1])) + $
  (lindgen(gridsize[1]) ## replicate(1,gridsize[0]) * gridsize[0])) *
nuniq
; now histogram everything
gridhist = reform(histogram(mappedgrids + rebin
(pixincrement,gridsize), $
  bin=1, min=0, max=npix*nuniq-1), nuniq, gridsize[0], gridsize[1])

; get the mode and value out of the histogram, for each pixel
magnitude_of_mode = max(gridhist, dim=1, imode)
value_of_mode = uniqvals[imode - pixincrement]


IDL> print, magnitude_of_mode
         3         2         2         4
         4         2         3         2
         5         5         5         5
IDL> print, value_of_mode
     1.00000      9.00000      8.00000      2.00000
    -5.00000     -10.0000      2.00000      3.00000
     0.00000      1.00000      2.00000      3.00000


-Jeremy.
```

Subject: Re: Mode and variation of cells in multiple grids (3-D problem)

On Jan 26, 4:56 pm, Jeremy Bailin <astroco...@gmail.com> wrote:
> On Jan 25, 7:49 pm, Ed Hyer <ejh...@gmail.com> wrote:
>
>
>
>
>
>
>> On Jan 25, 12:47 pm, Ed Hyer <ejh...@gmail.com> wrote:
>
>>> 1) The number of different values in the matched grids;
>>> 2) The mode, that is, the frequency of the most common value.
>
>> So, what I ended up with was something like this:
>
>> allgrids=[[[grid1]],[[grid2]],etc.]
>> matchcount=allgrids *0b; initialize
>> for i1=0,ngrids-1 do for i2=0,ngrids-1 do matchcount[*,*,i1] +=
>> (allgrids[*,*,i1] eq allgrids[*,*,i2])
>> magnitude_of_mode = max(matchcount,dim=3,imode)
>> value_of_mode = allgrids[imode]
>
>> So 2 for-loops still needed, but across the shortest dimension.
>
>> --Edward H.
>
> Here's a pure IDL no-loop solution that uses... <drum roll>... the
> combination of VALUE_LOCATE and HISTOGRAM! (tell me you didn't see
> that coming).
>
> ; some sample data
> grid1 = [[1., 0, -10, 1.],[-5, -10, 9, 8],[0, 1, 2, 3]]
> grid2 = [[1., 9, 8, 2],[-5, 0, 2, 3],[0, 1, 2, 3]]
> grid3 = [[-10, 8, 9, 2],[-5, -10, 1, 3],[0, 1, 2, 3]]
> grid4 = [[0, 9, 8, 2],[-10, 2, 2, 2],[0, 1, 2, 3]]
> grid5 = [[1., -2, -20, 2],[-5, 2, 2, 8],[0, 1, 2, 3]]
>
> allgrids = [[[grid1]],[[grid2]],[[grid3]],[[grid4]],[[grid5]]]
>
> gridsize=size(allgrids,/dimen)
> npix = gridsize[0]*gridsize[1]
> uniqvals = allgrids[uniq(allgrids,sort(allgrids))]
> nuniq = n_elements(uniqvals)
> ; map values into their index in uniqvals so we only need to
> ; deal with integers
> mappedgrids = value_locate(uniqvals, allgrids)
> ; now we can use histogram!

> ; first give each pixel its own increment so that the histogram for
> ; the pixels run from 0..nuniq-1, nuniq..2*nuniq-1, etc.
> pixincrement = ((lindgen(gridsize[0]) # replicate(1,gridsize[1])) + $
> (lindgen(gridsize[1]) ## replicate(1,gridsize[0]) * gridsize[0])) *
> nuniq
> ; now histogram everything
> gridhist = reform(histogram(mappedgrids + rebin
> (pixincrement,gridsize), $
> bin=1, min=0, max=npix*nuniq-1), nuniq, gridsize[0], gridsize[1])
>
> ; get the mode and value out of the histogram, for each pixel
> magnitude_of_mode = max(gridhist, dim=1, imode)
> value_of_mode = uniqvals[imode - pixincrement]
>
> IDL> print, magnitude_of_mode
> 3 2 2 4
> 4 2 3 2
> 5 5 5 5
> IDL> print, value_of_mode
> 1.00000 9.00000 8.00000 2.00000
> -5.00000 -10.0000 2.00000 3.00000
> 0.00000 1.00000 2.00000 3.00000
>
> -Jeremy.

Oh, and I almost forgot your first requirement: the number of
different values at each pixel:

variation_of_cells = total(gridhist gt 0, 1, /int)

IDL> print, variation_of_cells
3 4 4 2
2 3 3 3
1 1 1 1

-Jeremy.

Subject: Re: Mode and variation of cells in multiple grids (3-D problem)
Posted by MarioIncandenza on Thu, 28 Jan 2010 15:47:10 GMT
View Forum Message <> Reply to Message

On Jan 26, 8:37 pm, Jeremy Bailin <astroco...@gmail.com> wrote:
>> ; map values into their index in uniqvals so we only need to
>> ; deal with integers
>> mappedgrids = value_locate(uniqvals, allgrids)
>> ; now we can use histogram!
>> ; first give each pixel its own increment so that the histogram for

>> ; the pixels run from 0..nuniq-1, nuniq..2*nuniq-1, etc.
>> pixincrement = ((lindgen(gridsize[0]) # replicate(1,gridsize[1])) + $
>>   (lindgen(gridsize[1]) ## replicate(1,gridsize[0]) * gridsize[0])) *
>> nuniq

:slow clap:

Nicely done!

Subject: Re: Mode and variation of cells in multiple grids (3-D problem)
Posted by Jeremy Bailin on Fri, 29 Jan 2010 12:58:08 GMT
View Forum Message <> Reply to Message

On Jan 28, 10:47 am, Ed Hyer <ejh...@gmail.com> wrote:
> On Jan 26, 8:37 pm, Jeremy Bailin <astroco...@gmail.com> wrote:
>
>>> ; map values into their index in uniqvals so we only need to
>>> ; deal with integers
>>> mappedgrids = value_locate(uniqvals, allgrids)
>>> ; now we can use histogram!
>>> ; first give each pixel its own increment so that the histogram for
>>> ; the pixels run from 0..nuniq-1, nuniq..2*nuniq-1, etc.
>>> pixincrement = ((lindgen(gridsize[0]) # replicate(1,gridsize[1])) + $
>>>   (lindgen(gridsize[1]) ## replicate(1,gridsize[0]) * gridsize[0])) *
>>> nuniq
>
> :slow clap:
>
> Nicely done!

Thanks. :-)=

You might run into memory problems if nuniq is much greater than the number of grids, since the histogram ends up being nuniq x nx x ny but only at most ngrid x nx x ny of the elements can be non-zero. I've been thinking about ways around that, but it gets complicated!

-Jeremy.
