Subject: Re: find missing elements in an array
Posted by havok2063 on Mon, 07 Jan 2013 15:37:44 GMT
View Forum Message <> Reply to Message

On Thursday, December 27, 2012 1:37:48 PM UTC-5, Brian Cherinka wrote:
> I have a peak-finding routine that finds, well...peaks, in a larger array. I know how many peaks there are suppose to be, and I know the way they should be arranged, e.g.
>
>
>
> 1 set of 5 peaks, a gap, then a set of 10 peaks, a gap, then a set of 7 peaks etc....
>
>
>
> and I know how many sets of peaks there are as well. But sometimes these peaks could be missing and I'm trying to find a way to locate them based on nothing more than the input data, and the expected values.
>
>
>
> so with the above example, I know there should be 22 peaks total in an array of data of say 30 elements, grouped in 3 sets of [5,10,7] peaks. I don't know where the peaks are apriori in the large array of 30 elements or where they should be.
>
>
>
> If I find 20 peaks, I know there are 2 missing but I would like to find out which peaks they are, in what sets, and where in the data array of 30 elements they are. Any ideas?
>
>
>
> I have a current method in place involving using the separations between found peaks but it is bulky, and inconsistent. I am looking for alternative methods or help in making the method I do have implemented more efficient. Below I will describe my currently attempted method.
>
>
>
> If all peaks were found then the separations (element+1 - element) should be something like
>
>
>
> seps = [5,4,5,6,15,11,10,11,12,12,12,11,11,10,14,4,4,5,6,4,5,4] 22 elements
>
>
>
> and I know there should be 2 gaps that separate the 3 bundles ; the 15 and the 14 indicate gap locations.
>

>
>
> if some are missing it might look like
>
>
>
> seps = [5,4,5,6,15,11,10,11,16,12,11,11,10,12,4,4,10,4,5,4] 20 elements ; here a missing peak is in the 2nd and 3rd sets.
>
>
>
> I do have an array of expected separation between peaks in the sets, so [5,11,4] would be the expected (or lets say median) separations in the above example. So I basically loop over each set and look for regions in the seps array where the value is greater than the expected value; basically looking for the gaps. If all the peaks are there I only find 2 gaps. In the above example of 2 missing peaks, I find 4 gaps and can figure out that there are 2 missing peaks and where they are and in which sets. I've had to refine it (and make it more complicated) where the separation decreases from 1 set to the next, as in the above set 2 to set 3 transition. But it's still inconsistent. It mostly works but if the separation values are different by 1 or 2, then my code isn't very robust. The problems come in then with using equalities (LE,GE) vs not (LT,GT) and the fact the the actual separations can vary up or down from the expected value.
>
>
>
>
>
> My loop here is something like this, with, for the above example, nbundle=3, medsep=[5,11,4] is the expected separation, ntot = [5,15,22] is the cumulative total of the number of peaks in each set =[5,10,7]
>
>
>
> test=-1
>
> tmp=where(seps gt medsep[0]+2)
>
> test=[test,tmp[where(tmp lt ntot[0])]]
>
> for bid=1,nbundle-2 do begin
>
> nt=n_elements(test)
>
> if bid eq 1 then begin
>
> ;refinement to handle the set 2 to 3 transition with the decrease which I've had to modify slightly and is somewhat inconsistent
>
> tmp=where(sep[test[nt-1]+1:n_elements(sep)-1] le medsep[bid+1])+test[nt-1]+1

>
>   tmp = tmp[where(tmp+(indgen(n_elements(tmp))) le ntot[bid]+1)]-2
>
>   if n_elements(tmp) gt 1 then tmp=tmp[n_elements(tmp)-1] ; in place in case it finds a few elements that meet the condition
>
>   test=[test,tmp]
>
>   endif else begin
>
>   ;original implemenation
>
>   tmp=where(sep[test[nt-1]+1:n_elements(sep)-1] ge medsep[bid]+2)+test[nt-1]+1
>
>   test=[test,tmp[where(tmp+(indgen(n_elements(tmp))+1) le ntot[bid])]]
>
>   endelse
>
> endfor
>
> gaps = test[where(test gt 0)]
>
> ngaps = n_elements(gaps)
>
>
>
> It actually gets more complicated than this because I first look for missing peaks within sets, and then I look for missing peaks at the edges of sets which involves very similar code, well the same code to find gaps, and then different code to ascertain whether it's a missing edge or not. If I could simplify these two stages down into one, that would be great as well.
>
>
>
> I'm looking for something that will be robust because this needs to be automated and be able to handle any input slightly varying inputs.
>
>
>
> I think this about covers it. :)
>
>
>
> Can value_locate solve my problem? Any help would be appreciated. Thanks. :)

Does anyone have some thoughts on this? I could use some help on this problem. This is becoming quite buggy and finicky and I've already added too many bandaids.

Subject: Re: find missing elements in an array
Posted by Craig Markwardt on Mon, 07 Jan 2013 20:02:22 GMT
View Forum Message <> Reply to Message

On Monday, January 7, 2013 10:37:44 AM UTC-5, Brian Cherinka wrote:
> On Thursday, December 27, 2012 1:37:48 PM UTC-5, Brian Cherinka wrote:
>
>> I have a peak-finding routine that finds, well...peaks, in a larger array. I know how many peaks there are suppose to be, and I know the way they should be arranged, e.g.
>
>>
>
>>
>
>>
>
>> 1 set of 5 peaks, a gap, then a set of 10 peaks, a gap, then a set of 7 peaks etc....
>
>>
>
>>
>
>>
>

>> and I know how many sets of peaks there are as well. But sometimes these peaks could be missing and I'm trying to find a way to locate them based on nothing more than the input data, and the expected values.
>
>>
>
>>
>
>>
>

>> so with the above example, I know there should be 22 peaks total in an array of data of say 30 elements, grouped in 3 sets of [5,10,7] peaks. I don't know where the peaks are apriori in the large array of 30 elements or where they should be.
>
>>
>
>>
>
>>
>

>> If I find 20 peaks, I know there are 2 missing but I would like to find out which peaks they are, in what sets, and where in the data array of 30 elements they are. Any ideas?
>
>>
>

>>
>
>>
>
>> I have a current method in place involving using the separations between found peaks but it is bulky, and inconsistent. I am looking for alternative methods or help in making the method I do have implemented more efficient. Below I will describe my currently attempted method.
>
>>
>
>>
>
>>
>
>> If all peaks were found then the separations (element+1 - element) should be something like
>
>>
>
>>
>
>>
>
>> seps = [5,4,5,6,15,11,10,11,12,12,12,11,11,10,14,4,4,5,6,4,5,4] 22 elements
>
>>
>
>>
>
>>
>
>> and I know there should be 2 gaps that separate the 3 bundles ; the 15 and the 14 indicate gap locations.
>
>>
>
>>
>
>>
>
>> if some are missing it might look like
>
>>
>
>>
>
>>
>
>> seps = [5,4,5,6,15,11,10,11,16,12,11,11,10,12,4,4,10,4,5,4] 20 elements ; here a missing

peak is in the 2nd and 3rd sets.
>
>>
>
>>
>
>>
>
>> I do have an array of expected separation between peaks in the sets, so [5,11,4] would be the expected (or lets say median) separations in the above example. So I basically loop over each set and look for regions in the seps array where the value is greater than the expected value; basically looking for the gaps. If all the peaks are there I only find 2 gaps. In the above example of 2 missing peaks, I find 4 gaps and can figure out that there are 2 missing peaks and where they are and in which sets. I've had to refine it (and make it more complicated) where the separation decreases from 1 set to the next, as in the above set 2 to set 3 transition. But it's still inconsistent. It mostly works but if the separation values are different by 1 or 2, then my code isn't very robust. The problems come in then with using equalities (LE,GE) vs not (LT,GT) and the fact the the actual separations can vary up or down from the expected value.
>
>>
>
>>
>
>>
>
>>
>
>>
>
>> My loop here is something like this, with, for the above example, nbundle=3, medsep=[5,11,4] is the expected separation, ntot = [5,15,22] is the cumulative total of the number of peaks in each set =[5,10,7]
>
>>
>
>>
>
>>
>
>> test=-1
>
>>
>
>> tmp=where(seps gt medsep[0]+2)
>
>>
>
>> test=[test,tmp[where(tmp lt ntot[0])]]

```
>
>>
>
>>  for bid=1,nbundle-2 do begin
>
>>
>
>>   nt=n_elements(test)
>
>>
>
>>   if bid eq 1 then begin
>
>>
>
>>             ;refinement to handle the set 2 to 3 transition with the decrease which I've had to
modify slightly and is somewhat inconsistent
>
>>
>
>>      tmp=where(sep[test[nt-1]+1:n_elements(sep)-1] le medsep[bid+1])+test[nt-1]+1
>
>>
>
>>      tmp = tmp[where(tmp+(indgen(n_elements(tmp))) le ntot[bid]+1)]-2
>
>>
>
>>      if n_elements(tmp) gt 1 then tmp=tmp[n_elements(tmp)-1] ; in place in case it finds a few
elements that meet the condition
>
>>
>
>>      test=[test,tmp]
>
>>
>
>>   endif else begin
>
>>
>
>>            ;original implemenation
>
>>
>
>>        tmp=where(sep[test[nt-1]+1:n_elements(sep)-1] ge medsep[bid]+2)+test[nt-1]+1
>
>>
```

>
>>   test=[test,tmp[where(tmp+(indgen(n_elements(tmp))+1) le ntot[bid])]]
>
>>
>
>>   endelse
>
>>
>
>> endfor
>
>>
>
>> gaps = test[where(test gt 0)]
>
>>
>
>> ngaps = n_elements(gaps)
>
>>
>
>>
>
>>
>
>> It actually gets more complicated than this because I first look for missing peaks within sets, and then I look for missing peaks at the edges of sets which involves very similar code, well the same code to find gaps, and then different code to ascertain whether it's a missing edge or not. If I could simplify these two stages down into one, that would be great as well.
>
>>
>
>>
>
>>
>
>> I'm looking for something that will be robust because this needs to be automated and be able to handle any input slightly varying inputs.
>
>>
>
>>
>
>>
>
>> I think this about covers it. :)
>
>>

>
>>
>
>>
>
>> Can value_locate solve my problem? Any help would be appreciated. Thanks. :)
>
>
>
> Does anyone have some thoughts on this? I could use some help on this problem. This is becoming quite buggy and finicky and I've already added too many bandaids.

I wondered if you could try a cross-correlation technique. You know where your peaks should be, so you can create a template mask. You can slide this template across your data to find matches, a la C_CORRELATE(). A missing peak wouldn't contribute to the correlation amplitude, but it shouldn't subtract either. I don't know how bar-code readers work, but I suspect it is this way.

Best wishes,
Craig

Subject: Re: find missing elements in an array
Posted by havok2063 on Wed, 09 Jan 2013 20:56:47 GMT
View Forum Message <> Reply to Message

On Monday, January 7, 2013 3:02:22 PM UTC-5, Craig Markwardt wrote:
> On Monday, January 7, 2013 10:37:44 AM UTC-5, Brian Cherinka wrote:
>
>> On Thursday, December 27, 2012 1:37:48 PM UTC-5, Brian Cherinka wrote:
>
>>
>
>>> I have a peak-finding routine that finds, well...peaks, in a larger array. I know how many peaks there are suppose to be, and I know the way they should be arranged, e.g.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>

>
>>> 1 set of 5 peaks, a gap, then a set of 10 peaks, a gap, then a set of 7 peaks etc....
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> and I know how many sets of peaks there are as well. But sometimes these peaks could be missing and I'm trying to find a way to locate them based on nothing more than the input data, and the expected values.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> so with the above example, I know there should be 22 peaks total in an array of data of say 30 elements, grouped in 3 sets of [5,10,7] peaks. I don't know where the peaks are apriori in the large array of 30 elements or where they should be.
>
>>
>
>>>
>
>>
>
>>>
>
>>

>
>>>
>
>>
>
>>> If I find 20 peaks, I know there are 2 missing but I would like to find out which peaks they are, in what sets, and where in the data array of 30 elements they are. Any ideas?
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> I have a current method in place involving using the separations between found peaks but it is bulky, and inconsistent. I am looking for alternative methods or help in making the method I do have implemented more efficient. Below I will describe my currently attempted method.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> If all peaks were found then the separations (element+1 - element) should be something like
>
>>
>
>>>
>
>>
>

>>>
>
>>
>
>>>
>
>>
>
>>> seps = [5,4,5,6,15,11,10,11,12,12,12,11,11,10,14,4,4,5,6,4,5,4] 22 elements
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> and I know there should be 2 gaps that separate the 3 bundles ; the 15 and the 14 indicate gap locations.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> if some are missing it might look like
>
>>
>
>>>
>
>>

>
>>>
>
>>
>
>>>
>
>>
>
>>> seps = [5,4,5,6,15,11,10,11,16,12,11,11,10,12,4,4,10,4,5,4] 20 elements ; here a missing peak is in the 2nd and 3rd sets.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> I do have an array of expected separation between peaks in the sets, so [5,11,4] would be the expected (or lets say median) separations in the above example. So I basically loop over each set and look for regions in the seps array where the value is greater than the expected value; basically looking for the gaps. If all the peaks are there I only find 2 gaps. In the above example of 2 missing peaks, I find 4 gaps and can figure out that there are 2 missing peaks and where they are and in which sets. I've had to refine it (and make it more complicated) where the separation decreases from 1 set to the next, as in the above set 2 to set 3 transition. But it's still inconsistent. It mostly works but if the separation values are different by 1 or 2, then my code isn't very robust. The problems come in then with using equalities (LE,GE) vs not (LT,GT) and the fact the the actual separations can vary up or down from the expected value.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>

>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> My loop here is something like this, with, for the above example, nbundle=3, medsep=[5,11,4] is the expected separation, ntot = [5,15,22] is the cumulative total of the number of peaks in each set =[5,10,7]
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> test=-1
>
>>
>
>>>
>
>>
>
>>> tmp=where(seps gt medsep[0]+2)
>
>>
>
>>>
>
>>
>
>>> test=[test,tmp[where(tmp lt ntot[0])]]
>
>>

>
>>>
>
>>
>
>>> for bid=1,nbundle-2 do begin
>
>>
>
>>>
>
>>
>
>>>   nt=n_elements(test)
>
>>
>
>>>
>
>>
>
>>>   if bid eq 1 then begin
>
>>
>
>>>
>
>>
>
>>>   ;refinement to handle the set 2 to 3 transition with the decrease which I've had to modify slightly and is somewhat inconsistent
>
>>
>
>>>
>
>>
>
>>>   tmp=where(sep[test[nt-1]+1:n_elements(sep)-1] le medsep[bid+1])+test[nt-1]+1
>
>>
>
>>>
>
>>
>
>>>   tmp = tmp[where(tmp+(indgen(n_elements(tmp))) le ntot[bid]+1)]-2
>

>>
>
>>>
>
>>
>
>>>   if n_elements(tmp) gt 1 then tmp=tmp[n_elements(tmp)-1] ; in place in case it finds a few elements that meet the condition
>
>>
>
>>>
>
>>
>
>>>   test=[test,tmp]
>
>>
>
>>>
>
>>
>
>>>   endif else begin
>
>>
>
>>>
>
>>
>
>>>   ;original implemenation
>
>>
>
>>>
>
>>
>
>>>   tmp=where(sep[test[nt-1]+1:n_elements(sep)-1] ge medsep[bid]+2)+test[nt-1]+1
>
>>
>
>>>
>
>>
>
>>>   test=[test,tmp[where(tmp+(indgen(n_elements(tmp))+1) le ntot[bid])]]

>
>>
>
>>>
>
>>
>
>>> endelse
>
>>
>
>>>
>
>>
>
>>> endfor
>
>>
>
>>>
>
>>
>
>>> gaps = test[where(test gt 0)]
>
>>
>
>>>
>
>>
>
>>> ngaps = n_elements(gaps)
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> It actually gets more complicated than this because I first look for missing peaks within sets,

and then I look for missing peaks at the edges of sets which involves very similar code, well the same code to find gaps, and then different code to ascertain whether it's a missing edge or not. If I could simplify these two stages down into one, that would be great as well.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> I'm looking for something that will be robust because this needs to be automated and be able to handle any input slightly varying inputs.
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>> I think this about covers it. :)
>
>>
>
>>>
>
>>
>
>>>
>
>>
>
>>>

>
>>
>
>>> Can value_locate solve my problem? Any help would be appreciated. Thanks. :)
>
>>
>
>>
>
>>
>
>> Does anyone have some thoughts on this? I could use some help on this problem. This is becoming quite buggy and finicky and I've already added too many bandaids.
>
>
>
> I wondered if you could try a cross-correlation technique. You know where your peaks should be, so you can create a template mask. You can slide this template across your data to find matches, a la C_CORRELATE(). A missing peak wouldn't contribute to the correlation amplitude, but it shouldn't subtract either. I don't know how bar-code readers work, but I suspect it is this way.
>
>
>
> Best wishes,
>
> Craig

Well I don't actually know beforehand where the peaks are supposed to be. I just know how many peaks there should be and how they are arranged (in so many groups of so many peaks with this expected spacing). My first step is to find the supposed peaks and their locations then I do this complicated business of looking for peaks missing if it doesn't find how many it's suppose to. I guess I could build a model based off the first pass of finding supposed peaks.
