Subject: Re: cool way to determine durations in time series
Posted by news.qwest.net on Fri, 20 Jan 2006 19:01:35 GMT
View Forum Message <> Reply to Message

"Thomas Pfaff" <yael@gmx.net> wrote in message
news:43com4F1mh6c1U1@individual.net...
> Hello everyone,
>
> I'm doing some IDL-abuse in hydrology, so my question might seem a bit
> odd, maybe.
>
> A task that is occurring quite regularly is to determine the duration of
> certain events. For example "What is the longest contiguous duration of
> stream flow below/above a certain discharge"
>
> Assuming I have an equidistant time series (e.g. one value each day)
> this basically reduces to the question of how can I transform an array
> like this
>
> series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]
>
> into something like this
>
> durations = [2,1,3,2]
>
> which is I want to count all contiguous fields of '1's in an array.
>
> Somehow my brain wants to use HISTOGRAM for this, but I just can't see
> how to do it.
> At the moment I'm helping myself by using CONVOL(to highlight the edges)
> and WHERE(to get the differences between two adjacent edge indices) but
> as the data gets more, this becomes extremely tedious as well as memory
> consuming (see the example below). Besides, CONVOL wouldn't work if a
> series started or ended with '1's as it can't correctly apply the kernel
> to those elements.
>
> Any ideas? Anyone who has seen this problem in one of his/her maths
> textbooks? Hints to literature are also highly appreciated.
>
> Thanks in advance,
>
>
> Thomas

Hi Thomas,
here is a starting point for you.
Assuming series contains integers, this piece of code gives you the

durations of
each repeated value EXCEPT for the very end of the series. Should be easy
enough to
patcht that up. Also, should be easy enough to remove the series of zeros,
and explicitly
make it do a specific number (like 1 in your example).
Also, one can apply this to your "above/below" by first applying "series =
data > threshold"
which is what I am assuming you are doing above to get your "series".

Anyways, here it is (and it is just a starting point suggestion, you will
have to
modify it a bit)

```
series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]
d = series - shift(series,1)
w =where(d ne 0)
newdur = w - shift(w,1)
newdur[0] = w[0]
print,newdur

>   newdur = 2 4 1 1 3 2
```

Note, it is missing that last "2" points, so you have to patch that.

Cheers,
bob

Subject: Re: cool way to determine durations in time series
Posted by btt on Fri, 20 Jan 2006 21:50:06 GMT

View Forum Message <> Reply to Message

Thomas Pfaff wrote:
> Hello everyone,
>
> I'm doing some IDL-abuse in hydrology, so my question might seem a bit
> odd, maybe.
>
> A task that is occurring quite regularly is to determine the duration of
> certain events. For example "What is the longest contiguous duration of
> stream flow below/above a certain discharge"
>
> Assuming I have an equidistant time series (e.g. one value each day)
> this basically reduces to the question of how can I transform an array
> like this
>
> series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]

>
> into something like this
>
> durations = [2,1,3,2]
>
> which is I want to count all contiguous fields of '1's in an array.
>
> Somehow my brain wants to use HISTOGRAM for this, but I just can't see
> how to do it.
> At the moment I'm helping myself by using CONVOL(to highlight the edges)
> and WHERE(to get the differences between two adjacent edge indices) but
> as the data gets more, this becomes extremely tedious as well as memory
> consuming (see the example below). Besides, CONVOL wouldn't work if a
> series started or ended with '1's as it can't correctly apply the kernel
> to those elements.
>

I think I would use a combination of LABEL_REGION and HISTOGRAM.

```
****START
series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]

nSeries= n_elements(Series)

buffered = [0,series,0]
dummy = FIX(LABEL_REGION(buffered))
label = dummy[1:nSeries]

H = HISTOGRAM(label, MIN = 1S)

print, series
print, label
print, H
*****END
```

Note the you must pad series with "background" values at the endpoints.

Cheers,
ben

Subject: Re: cool way to determine durations in time series
Posted by Dick Jackson on Sat, 21 Jan 2006 05:11:49 GMT
View Forum Message <> Reply to Message

Hi all,

"Ben Tupper" <btupper@bigelow.org> wrote in message
news:43d48fF1n6d7qU1@individual.net...
> Thomas Pfaff wrote:

>> [...] how can I transform an array
>> like this
>>
>> series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]
>>
>> into something like this
>>
>> durations = [2,1,3,2]
>>
>> which is I want to count all contiguous fields of '1's in an array.

>
> I think I would use a combination of LABEL_REGION and HISTOGRAM.
>
>
> ****START
> series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]
>
> nSeries= n_elements(Series)
>
> buffered = [0,series,0]
> dummy = FIX(LABEL_REGION(buffered))
> label = dummy[1:nSeries]
>
> H = HISTOGRAM(label, MIN = 1S)
>
> print, series
> print, label
> print, H
> *****END
>
>
> Note the you must pad series with "background" values at the endpoints.

I have to say, that looks pretty cool, as requested! I've approached this
another way, playing from Ben's 'buffered' array, finding where we have
transitions from 0->1 or 1->0:

```
;   Find where all transitions occur
whereChange = Where(buffered[1:*] NE buffered, nChange)

;   Change array to [2, m]
whereChange = Reform(whereChange, 2, nChange/2, /Overwrite)
```

```
;   Measure distance from odd transitions to even transitions
durations = Reform(whereChange[1, *] - whereChange[0, *])
```

In case anyone would want to know the time or memory efficiency of these (I was curious), I tried to optimize them as much as possible and put them through their paces with long series:

=====

```
PRO TimeSeriesDurations, n

;series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]

IF N_Elements(n) EQ 0 THEN n = 1E6
series = RandomU(seed, n) GT 0.5

nSeries= n_elements(Series)

buffered = [0,series,0]

; DJ method:

m0 = Memory(/Current)
t0 = SysTime(/Seconds)

whereChange = Where(buffered[1:*] NE buffered, nChange)
IF nChange EQ 0 THEN Return  ; No 1's in series
whereChange = Reform(whereChange, 2, nChange/2, /Overwrite)

durations = Reform(whereChange[1, *] - whereChange[0, *])

Print, 'DJ time: ', SysTime(/Seconds)-t0
Print, 'DJ memory: ', Memory(/Current)-m0

; BT method:

m0 = Memory(/Current)
t0 = SysTime(/Seconds)

;   Need to use ULong for longer series:
;dummy = LABEL_REGION(buffered, /ULong)
;label = dummy[1:nSeries]
;   Compressed to this for efficiency:
label = (LABEL_REGION(buffered, /ULong))[1:nSeries]

H = HISTOGRAM(label, MIN = 1S)
```

```
Print, 'BT time: ', SysTime(/Seconds)-t0
Print, 'BT memory: ', Memory(/Current)-m0

Print, 'Differing results: ', Total(durations NE H)

END
```

=====

Running this gives:

```
IDL> timeseriesdurations,1E6
DJ time:      0.030999899
DJ memory:      3003060
BT time:      0.062000036
BT memory:      5001132
Differing results:      0.000000

IDL> TimeSeriesDurations,1E7
DJ time:       0.39000010
DJ memory:     30011760
BT time:       0.51600003
BT memory:     50004032
Differing results:      0.000000
```

Any other methods out there? Hope this helps!

--
Cheers,
--
-Dick

Dick Jackson   /   dick@d-jackson.com
D-Jackson Software Consulting /   http://www.d-jackson.com
Calgary, Alberta, Canada   / +1-403-242-7398 / Fax: 241-7392

Subject: Re: cool way to determine durations in time series
Posted by Thomas Pfaff on Mon, 23 Jan 2006 10:22:33 GMT
View Forum Message <> Reply to Message

Hi all,

Thanks for this most instructive help.
Good to have you people around.

Thomas

Dick Jackson schrieb:
> Hi all,
>
> "Ben Tupper" <btupper@bigelow.org> wrote in message
> news:43d48fF1n6d7qU1@individual.net...
>
>> Thomas Pfaff wrote:
>
>
>>> [...] how can I transform an array
>>> like this
>>>
>>> series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]
>>>
>>> into something like this
>>>
>>> durations = [2,1,3,2]
>>>
>>> which is I want to count all contiguous fields of '1's in an array.
>
>
>> I think I would use a combination of LABEL_REGION and HISTOGRAM.
>>
>>
>> ****START
>> series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]
>>
>> nSeries= n_elements(Series)
>>
>> buffered = [0,series,0]
>> dummy = FIX(LABEL_REGION(buffered))
>> label = dummy[1:nSeries]
>>
>> H = HISTOGRAM(label, MIN = 1S)
>>
>> print, series
>> print, label
>> print, H
>> *****END
>>
>>
>> Note the you must pad series with "background" values at the endpoints.
>
>
> I have to say, that looks pretty cool, as requested! I've approached this
> another way, playing from Ben's 'buffered' array, finding where we have
> transitions from 0->1 or 1->0:

```
>
> ;   Find where all transitions occur
> whereChange = Where(buffered[1:*] NE buffered, nChange)
>
> ;   Change array to [2, m]
> whereChange = Reform(whereChange, 2, nChange/2, /Overwrite)
>
> ;   Measure distance from odd transitions to even transitions
> durations = Reform(whereChange[1, *] - whereChange[0, *])
>
> In case anyone would want to know the time or memory efficiency of these (I
> was curious), I tried to optimize them as much as possible and put them
> through their paces with long series:
>
> =====
>
> PRO TimeSeriesDurations, n
>
> ;series = [1,1,0,0,0,0,1,0,1,1,1,0,0,1,1]
>
> IF N_Elements(n) EQ 0 THEN n = 1E6
> series = RandomU(seed, n) GT 0.5
>
> nSeries= n_elements(Series)
>
> buffered = [0,series,0]
>
> ; DJ method:
>
> m0 = Memory(/Current)
> t0 = SysTime(/Seconds)
>
> whereChange = Where(buffered[1:*] NE buffered, nChange)
> IF nChange EQ 0 THEN Return  ; No 1's in series
> whereChange = Reform(whereChange, 2, nChange/2, /Overwrite)
>
> durations = Reform(whereChange[1, *] - whereChange[0, *])
>
> Print, 'DJ time: ', SysTime(/Seconds)-t0
> Print, 'DJ memory: ', Memory(/Current)-m0
>
> ; BT method:
>
> m0 = Memory(/Current)
> t0 = SysTime(/Seconds)
>
> ;   Need to use ULong for longer series:
> ;dummy = LABEL_REGION(buffered, /ULong)
>  ;label = dummy[1:nSeries]
>  ;   Compressed to this for efficiency:
>  label = (LABEL_REGION(buffered, /ULong))[1:nSeries]
>
>  H = HISTOGRAM(label, MIN = 1S)
>
>  Print, 'BT time: ', SysTime(/Seconds)-t0
>  Print, 'BT memory: ', Memory(/Current)-m0
>
>  Print, 'Differing results: ', Total(durations NE H)
>
>  END
>
>  =====
>
>  Running this gives:
>
>  IDL> timeseriesdurations,1E6
>  DJ time:      0.030999899
>  DJ memory:      3003060
>  BT time:      0.062000036
>  BT memory:      5001132
>  Differing results:      0.000000
>
>  IDL> TimeSeriesDurations,1E7
>  DJ time:       0.39000010
>  DJ memory:     30011760
>  BT time:       0.51600003
>  BT memory:     50004032
>  Differing results:      0.000000
>
>  Any other methods out there? Hope this helps!
>
```