Subject: totalling data with LONG lists of indices
Posted by Jeremy Bailin on Tue, 22 Apr 2008 18:40:54 GMT
View Forum Message <> Reply to Message

In JD's magnificent histogram tutorial, there is a description of how to total data using a separate list of indices (with possible repeats),
along with a vague hint that "For large histograms, there are even more
efficient ways to do this with very short or no loops (e.g. using a histogram of the histogram)." I have exactly that situation. Because about 90% of my indices are not repeated, I have achieved a decent speed-up using the following code for the "single" cases ("indices", "data", and "hist" are the index list, data list, and final result respectively, and hist is already pre-dimensioned).

```
indhist = histogram(indices, omin=om, reverse_indices=indri)
dupehist = histogram(indhist, min=1, reverse_indices=duperi)
; unique cases, so we can use them to index the LHS:
if dupehist[0] gt 0 then begin
  just1 = duperi[duperi[0]:duperi[1]-1]
  hist[just1+om] += data[indri[indri[just1]]]
endif
```

And going with the brute-force for loop for the rest:

```
; loop through the rest
if n_elements(dupehist) gt 1 then begin
  multiples = duperi[duperi[1]:*]
  for j=0l,n_elements(multiples)-1 do begin
    elements = indri[indri[multiples[j]]:indri[multiples[j]+1]-1]
    hist[multiples[j]+om] += total(data[elements])
  endfor
endif
```

However, the loop is still going over hundreds of thousands of entries and I can't help but suspect that another histogram and some fancy footing with the i-vector would get rid of it. Does anyone have any suggestions? Thanks.

Subject: Re: totalling data with LONG lists of indices
Posted by Conor on Thu, 24 Apr 2008 12:36:08 GMT
View Forum Message <> Reply to Message

On Apr 22, 2:40 pm, Jeremy Bailin <astroco...@gmail.com> wrote:
> In JD's magnificent histogram tutorial, there is a description of how

> to total data using a separate list of indices (with possible
> repeats),
> along with a vague hint that "For large histograms, there are even
> more
> efficient ways to do this with very short or no loops (e.g. using a
> histogram of the histogram)." I have exactly that situation. Because
> about 90% of my indices are not repeated, I have achieved a decent
> speed-up using the following code for the "single" cases ("indices",
> "data", and "hist" are the index list, data list, and final result
> respectively, and hist is already pre-dimensioned).
>
> indhist = histogram(indices, omin=om, reverse_indices=indri)
> dupehist = histogram(indhist, min=1, reverse_indices=duperi)
> ; unique cases, so we can use them to index the LHS:
> if dupehist[0] gt 0 then begin
> just1 = duperi[duperi[0]:duperi[1]-1]
> hist[just1+om] += data[indri[indri[just1]]]
> endif
>
> And going with the brute-force for loop for the rest:
>
> ; loop through the rest
> if n_elements(dupehist) gt 1 then begin
> multiples = duperi[duperi[1]:*]
> for j=0l,n_elements(multiples)-1 do begin
> elements = indri[indri[multiples[j]]:indri[multiples[j]+1]-1]
> hist[multiples[j]+om] += total(data[elements])
> endfor
> endif
>
> However, the loop is still going over hundreds of thousands of entries
> and I can't help but suspect that another histogram and some fancy
> footing with the i-vector would get rid of it. Does anyone have any
> suggestions? Thanks.

I had to do something like this and found this page to be very handy:

http://www.dfanning.com/code_tips/drizzling.html

Subject: Re: totalling data with LONG lists of indices
Posted by Jeremy Bailin on Thu, 24 Apr 2008 18:40:47 GMT
View Forum Message <> Reply to Message

>> In JD's magnificent histogram tutorial, there is a description of how
>> to total data using a separate list of indices (with possible
>> repeats),
>> along with a vague hint that "For large histograms, there are even

>> more
>> efficient ways to do this with very short or no loops (e.g. using a
>> histogram of the histogram)." I have exactly that situation. Because
>> about 90% of my indices are not repeated, I have achieved a decent
>> speed-up using the following code for the "single" cases ("indices",
>> "data", and "hist" are the index list, data list, and final result
>> respectively, and hist is already pre-dimensioned).
>
>> indhist = histogram(indices, omin=om, reverse_indices=indri)
>> dupehist = histogram(indhist, min=1, reverse_indices=duperi)
>> ; unique cases, so we can use them to index the LHS:
>> if dupehist[0] gt 0 then begin
>>   just1 = duperi[duperi[0]:duperi[1]-1]
>>   hist[just1+om] += data[indri[indri[just1]]]
>> endif
>
>> And going with the brute-force for loop for the rest:
>
>> ; loop through the rest
>> if n_elements(dupehist) gt 1 then begin
>>   multiples = duperi[duperi[1]:*]
>>   for j=0l,n_elements(multiples)-1 do begin
>>   elements = indri[indri[multiples[j]]:indri[multiples[j]+1]-1]
>>   hist[multiples[j]+om] += total(data[elements])
>>   endfor
>> endif
>
>> However, the loop is still going over hundreds of thousands of entries
>> and I can't help but suspect that another histogram and some fancy
>> footing with the i-vector would get rid of it. Does anyone have any
>> suggestions? Thanks.
>
> I had to do something like this and found this page to be very handy:
>
> http://www.dfanning.com/code_tips/drizzling.html

Ah, yes, that would do it! I replaced my second for loop with JD's in
the dual histogram implementation, and it's now suitably fast. :-)=
Thanks!

David: if you could link the Drizzling page into the Histogram page at
that "histogram of a histogram" hint, that would be awesome...

-Jeremy.
