Subject: Re: Array juggling help needed
Posted by David Fanning on Fri, 23 Sep 2005 13:40:59 GMT
View Forum Message <> Reply to Message

Haje Korth writes:

> I need to expand and an array non-uniformly based on its content. I am
> trying to to the following:
>
> input array: [1,5,4,1]
> output array should be: [1,0.2,0.2,0.2,0.2,0.2,0.25,0.25,0.25,0.25,1]
>
> Each elements of the input array is basically tuned into
> fltarr(inputarray[i])/inputarray[i] and the subarray concatenated. Is there
> a way to do this in one step, without using "for" loops and array
> concatenation? If not, I can work around this, but knowing for sure that
> this doesn't work would at least allow me to stop thinking about this
> problem. :-)
>
> To me this looks kind of like a "REPLICATE" for vectors function?

We've got to get more people reading that Histogram Tutorial.
Does anyone have a picture of a sexy young woman in a
"Histogram" T-shirt they want to share?

This is the "index chunking" problem discussed in the tutorial
and last week in this newsgroup:

```
IDL> n = [1, 5, 4, 1]
IDL> d = 1./n
IDL> print, d
      1.00000     0.200000     0.250000      1.00000
IDL> h=histogram(total(n,/CUMULATIVE)-1,/BINSIZE,$
       MIN=0,REVERSE_INDICES=ri)
IDL> I=ri[0:n_elements(h)-1]-ri[0]
IDL> print, d[I]
1.00 0.20 0.20 0.20 0.20 0.20 0.25 0.25 0.25 0.25
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Array juggling help needed
Posted by Haje Korth on Fri, 23 Sep 2005 13:58:13 GMT
View Forum Message <> Reply to Message

David,
thanks for your help. Like you, I read the histogram tutorial regularly.
Unlike you, my brain seems to be too small to comprehend it and make use of
it. :-)

Cheers,
Haje

"David Fanning" <davidf@dfanning.com> wrote in message
news:MPG.1d9db3c5a36ee9b0989a99@news.frii.com...
> Haje Korth writes:
>
>> I need to expand and an array non-uniformly based on its content. I am
>> trying to to the following:
>>
>> input array: [1,5,4,1]
>> output array should be: [1,0.2,0.2,0.2,0.2,0.2,0.25,0.25,0.25,0.25,1]
>>
>> Each elements of the input array is basically tuned into
>> fltarr(inputarray[i])/inputarray[i] and the subarray concatenated. Is
>> there
>> a way to do this in one step, without using "for" loops and array
>> concatenation? If not, I can work around this, but knowing for sure that
>> this doesn't work would at least allow me to stop thinking about this
>> problem. :-)
>>
>> To me this looks kind of like a "REPLICATE" for vectors function?
>
> We've got to get more people reading that Histogram Tutorial.
> Does anyone have a picture of a sexy young woman in a
> "Histogram" T-shirt they want to share?
>
> This is the "index chunking" problem discussed in the tutorial
> and last week in this newsgroup:
>
> IDL> n = [1, 5, 4, 1]
> IDL> d = 1./n
> IDL> print, d
>      1.00000     0.200000     0.250000      1.00000
> IDL> h=histogram(total(n,/CUMULATIVE)-1,/BINSIZE,$
>       MIN=0,REVERSE_INDICES=ri)
> IDL> I=ri[0:n_elements(h)-1]-ri[0]
> IDL> print, d[I]
> 1.00 0.20 0.20 0.20 0.20 0.20 0.25 0.25 0.25 0.25

> 
> Cheers,
> 
> David
> 
> 
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/

---

Subject: Re: Array juggling help needed
Posted by news.qwest.net on Fri, 23 Sep 2005 14:28:14 GMT
View Forum Message <> Reply to Message

"Haje Korth" <haje.korth@nospam.jhuapl.edu> wrote in message
news:dh11ll$slo$1@aplcore.jhuapl.edu...
> David,
> thanks for your help. Like you, I read the histogram tutorial regularly.
> Unlike you, my brain seems to be too small to comprehend it and make use
> of it. :-)
> 
> Cheers,
> Haje

Well, you are ahead of me Haje. Not only did I not know
the answer, I don't even know what the question means :(

Cheers,
bob

---

Subject: Re: Array juggling help needed
Posted by David Fanning on Fri, 23 Sep 2005 14:34:42 GMT
View Forum Message <> Reply to Message

news.qwest.net writes:

> Well, you are ahead of me Haje. Not only did I not know
> the answer, I don't even know what the question means :(

Coyote is off looking for beautiful women now. We'll soon
have something for the *rest* of you on the Histogram
Tutorial page! :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

---

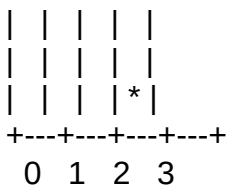
Subject: Re: Array juggling help needed
Posted by JD Smith on Fri, 23 Sep 2005 18:07:26 GMT
View Forum Message <> Reply to Message

On Fri, 23 Sep 2005 07:40:59 -0600, David Fanning wrote:

> This is the "index chunking" problem discussed in the tutorial
> and last week in this newsgroup:
>
> IDL> n = [1, 5, 4, 1]
> IDL> d = 1./n
> IDL> print, d
>  1.00000  0.200000  0.250000  1.00000
> IDL> h=histogram(total(n,/CUMULATIVE)-1,/BINSIZE,$
>  MIN=0,REVERSE_INDICES=ri)
> IDL> I=ri[0:n_elements(h)-1]-ri[0]
> IDL> print, d[I]
> 1.00 0.20 0.20 0.20 0.20 0.20 0.25 0.25 0.25 0.25

Maybe I should provide a little more explanation for what really is an underhanded trick. The trick requires that you understand how REVERSE_INDICES are setup. For a given bin in a histogram, the "i-vector" (or first half) part of REVERSE_INDICES (RI) will contain a series of index pairs into the "o-vector" (or second half) of RI. These two vectors could have been kept separate, saving countless person-hours of head scratching, but they were instead glued together into one ungainly beast. In any case, there will be a pair of indices in the i-vector for each bin in the histogram. This pair tells you the range of elements of REVERSE_INDICES which contain the original indices of the data which fell in that bin. If the bin is empty, the pair will be the same number (i.e. spanning no elements). This is the crux of our trick. We don't care about the data, or the histogram itself, of the indices in the data, just the i-vector.

Here's how it works. Let's imagine we have a histogram which has lots of empty bins, like this:



Remember, when a bin is empty, the corresponding pair of i-vector indices is the same number. The i-vector for this might look like: [5,5,5,5,6], which is to say:

Bin RI Range
==========
0 5:5 (i.e. empty)
1 5:5
2 5:5
3 5:5
6 5:6 (i.e. one inside)

For those first three empty bins, there were 4 5's in a row. Why is it 5? Because the histogram has 4 elements. So, we were able to get 4 5's in a row, simply by creating a histogram with an integer 3, like this:

```
IDL> h=histogram([3],MIN=0,/BINSIZE,REVERSE_INDICES=ri)
```

Subtracting off ri[0], and we have 4 0's in a row. Getting closer. How can we arrange to sparsely sprinkle a single drop in all the correct histogram buckets, such that the spacing between them will create the right sort of i-vector? By using total(/CUMULATIVE).

```
IDL> n=[1,5,4,1]
IDL> print,total(n,/CUMULATIVE)
     1.00000      6.00000      10.0000      11.0000
```

Now we must subtract 1 because it always takes 1 more index for an equivalent set of pairs in the i-vector. You can see that this is now the perfect "sparse sprinkling" to get just what we want:

```
IDL> print,histogram(total(n,/CUMULATIVE)-1,MIN=0)
         1         0         0         0         0         1
         0         0         0         1         1
```

Notice the spacing between the 1's... 1,5,4,1. We're almost there. Now the first part of RI will contain the chunked indices we need:

```
IDL>  h=histogram(total(n,/CUMULATIVE)-1,MIN=0,REVERSE_INDICES=ri)
IDL> print,ri[0:n_elements(h)-1]
```

```
      12       13       13       13       13       13
      14       14       14       14       15
```

Simply subtract off the offset:

```
IDL> print,ri[0:n_elements(h)-1]-ri[0]
       0        1        1        1        1        1
       2        2        2        2        3
```

And there you have it. This is not intuitive. It's simply a convenient trick to leverage the speed of HISTOGRAM to do something its designers probably never intended you to do.

JD

---

## Subject: Re: Array juggling help needed
Posted by David Fanning on Fri, 23 Sep 2005 21:54:17 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> Maybe I should provide a little more explanation for what really is an
> underhanded trick.

Since this has come up twice in the past 10 days, and given JD's usual lucid commentary, I thought I should write this up for the general riff-raft, uh, public. But could someone fill me in on *why* this problem comes up? I'm trying to imagine a problem this would solve and my mind is coming up blank. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

---

## Subject: Re: Array juggling help needed
Posted by Haje Korth on Mon, 26 Sep 2005 12:19:47 GMT
View Forum Message <> Reply to Message

JD,
thanks for sheeding further light onto the darkness of my histogram

knowledge!

Haje

"JD Smith" <jdsmith@as.arizona.edu> wrote in message
news:pan.2005.09.23.18.07.17.267156@as.arizona.edu...
> On Fri, 23 Sep 2005 07:40:59 -0600, David Fanning wrote:
>
>
>> This is the "index chunking" problem discussed in the tutorial
>> and last week in this newsgroup:
>>
>> IDL> n = [1, 5, 4, 1]
>> IDL> d = 1./n
>> IDL> print, d
>>       1.00000     0.200000     0.250000      1.00000
>> IDL> h=histogram(total(n,/CUMULATIVE)-1,/BINSIZE,$
>>        MIN=0,REVERSE_INDICES=ri)
>> IDL> I=ri[0:n_elements(h)-1]-ri[0]
>> IDL> print, d[I]
>> 1.00 0.20 0.20 0.20 0.20 0.20 0.25 0.25 0.25 0.25
>
> Maybe I should provide a little more explanation for what really is an
> underhanded trick. The trick requires that you understand how
> REVERSE_INDICES are setup. For a given bin in a histogram, the
> "i-vector" (or first half) part of REVERSE_INDICES (RI) will contain a
> series of index pairs into the "o-vector" (or second half) of RI.
> These two vectors could have been kept separate, saving countless
> person-hours of head scratching, but they were instead glued together
> into one ungainly beast. In any case, there will be a pair of indices
> in the i-vector for each bin in the histogram. This pair tells you
> the range of elements of REVERSE_INDICES which contain the original
> indices of the data which fell in that bin. If the bin is empty, the
> pair will be the same number (i.e. spanning no elements). This is the
> crux of our trick. We don't care about the data, or the histogram
> itself, of the indices in the data, just the i-vector.
>
> Here's how it works. Let's imagine we have a histogram which has lots
> of empty bins, like this:
>
>
> |  |  |  |  |
> |  |  |  |  |
> |  |  |  | * |
> +---+---+---+---+
>  0  1  2  3
>

> Remember, when a bin is empty, the corresponding pair of i-vector
> indices is the same number. The i-vector for this might look like:
> [5,5,5,5,6], which is to say:
>
>
> Bin RI Range
> ============
> 0 5:5 (i.e. empty)
> 1 5:5
> 2 5:5
> 3 5:5
> 6 5:6 (i.e. one inside)
>
> For those first three empty bins, there were 4 5's in a row. Why is
> it 5? Because the histogram has 4 elements. So, we were able to get
> 4 5's in a row, simply by creating a histogram with an integer 3, like
> this:
>
> IDL> h=histogram([3],MIN=0,/BINSIZE,REVERSE_INDICES=ri)
>
> Subtracting off ri[0], and we have 4 0's in a row. Getting closer.
> How can we arrange to sparsely sprinkle a single drop in all the
> correct histogram buckets, such that the spacing between them will
> create the right sort of i-vector? By using total(/CUMULATIVE).
>
> IDL> n=[1,5,4,1]
> IDL> print,total(n,/CUMULATIVE)
> 1.00000 6.00000 10.0000 11.0000
>
> Now we must subtract 1 because it always takes 1 more index for an
> equivalent set of pairs in the i-vector. You can see that this is now
> the perfect "sparse sprinkling" to get just what we want:
>
> IDL> print,histogram(total(n,/CUMULATIVE)-1,MIN=0)
> 1 0 0 0 0 1
> 0 0 0 1 1
>
> Notice the spacing between the 1's... 1,5,4,1. We're almost there.
> Now the first part of RI will contain the chunked indices we need:
>
> IDL> h=histogram(total(n,/CUMULATIVE)-1,MIN=0,REVERSE_INDICES=ri)
> IDL> print,ri[0:n_elements(h)-1]
> 12 13 13 13 13 13
> 14 14 14 14 15
>
> Simply subtract off the offset:
>
> IDL> print,ri[0:n_elements(h)-1]-ri[0]

>  0  1  1  1  1  1
>  2  2  2  2  3
>
> And there you have it.  This is not intuitive.  It's simply a
> convenient trick to leverage the speed of HISTOGRAM to do something
> its designers probably never intended you to do.
>
> JD
>

---