Subject: Re: Efficiently perform histogram reverse indices like procedure on a string array?
Posted by ben.bighair on Thu, 26 Jul 2012 01:34:22 GMT
View Forum Message <> Reply to Message

On Wednesday, July 25, 2012 7:39:33 PM UTC-4, Bogdanovist wrote:
> I have an array of a data structure, one tag of which is a string identifier indicating which location the data belongs to. There are many thousands of data points, but only about a dozen or so unique locations.
>
> I make frequent use of the HISTOGRAM function with the reverse_indices in order to carve up data by some identifier, most commonly the time. In this case, I want to divide out the data by site efficiently. I can&#39;t use HISTOGRAM on strings, so I need some other approach. There are plenty of ways this can be done, but I&#39;d like some views on the better and most efficient ways to do it.
>
> Take an example, say we have a simple string array
>
>     foo=[&#39;a&#39;,&#39;b&#39;,&#39;c& #39;,&#39;b&#39;,&#39;b&#39;,&#39;a& #39;,&#39;a&#39;,&#39;c&#39;]
>
> To determine the list of unique strings we could do
>
>    sfoo = foo[sort(foo)]
>    print,sfoo[uniq(sfoo)]
>
> We can then repeatedly use WHERE to find the indices in the data array(s) corresponding to each site.
>
> Is there a quicker/better way to do this? Repeatedly calling WHERE seems inefficient (certainly HISTOGRAM is way faster when it is usable)

Hi,

You can convert your strings to unique numbers - it's a bit awkward - and then you may find the the spacing between populated bins makes the whole thing drag when you do the histogram. But here goes...

```
foo = ['az','bs','cd','ba','ba','az','aa','c']   ; tricky strings
boo = strtrim(fix(byte(foo)),2)  ; note the 'fix' in there
soo = strjoin(boo)
noo = long(soo)
```

There you go - numbers as unique as the strings you started with.

Cheers,
Ben

## Subject: Re: Efficiently perform histogram reverse indices like procedure on a string array?
Posted by Jeremy Bailin on Thu, 26 Jul 2012 02:17:16 GMT
View Forum Message <> Reply to Message

On 7/25/12 9:09 PM, Bogdanovist wrote:
> I have an array of a data structure, one tag of which is a string identifier indicating which location the data belongs to. There are many thousands of data points, but only about a dozen or so unique locations.
>
> I make frequent use of the HISTOGRAM function with the reverse_indices in order to carve up data by some identifier, most commonly the time. In this case, I want to divide out the data by site efficiently. I can't use HISTOGRAM on strings, so I need some other approach. There are plenty of ways this can be done, but I'd like some views on the better and most efficient ways to do it.
>
> Take an example, say we have a simple string array
>
>     foo=['a','b','c','b','b','a','a','c']
>
> To determine the list of unique strings we could do
>
>     sfoo = foo[sort(foo)]
>     print,sfoo[uniq(sfoo)]
>
> We can then repeatedly use WHERE to find the indices in the data array(s) corresponding to each site.
>
> Is there a quicker/better way to do this? Repeatedly calling WHERE seems inefficient (certainly HISTOGRAM is way faster when it is usable)

Use VALUE_LOCATE to find where in the list of unique indices the
elements belong to, and use that index as a number that you can run
HISTOGRAM on.

(raise your hand everyone who saw that coming...)

-Jeremy.

## Subject: Re: Efficiently perform histogram reverse indices like procedure on a string array?
Posted by ben.bighair on Thu, 26 Jul 2012 17:30:37 GMT
View Forum Message <> Reply to Message

On Wednesday, July 25, 2012 10:17:16 PM UTC-4, Jeremy Bailin wrote:
> On 7/25/12 9:09 PM, Bogdanovist wrote:
> &gt; I have an array of a data structure, one tag of which is a string identifier indicating which location the data belongs to. There are many thousands of data points, but only about a dozen or

so unique locations.
> &gt;
> &gt; I make frequent use of the HISTOGRAM function with the reverse_indices in order to carve up data by some identifier, most commonly the time. In this case, I want to divide out the data by site efficiently. I can&#39;t use HISTOGRAM on strings, so I need some other approach. There are plenty of ways this can be done, but I&#39;d like some views on the better and most efficient ways to do it.
> &gt;
> &gt; Take an example, say we have a simple string array
> &gt;
> &gt;     foo=[&#39;a&#39;,&#39;b&#39;,&#39;c& #39;,&#39;b&#39;,&#39;b&#39;,&#39;a& #39;,&#39;a&#39;,&#39;c&#39;]
> &gt;
> &gt; To determine the list of unique strings we could do
> &gt;
> &gt;     sfoo = foo[sort(foo)]
> &gt;     print,sfoo[uniq(sfoo)]
> &gt;
> &gt; We can then repeatedly use WHERE to find the indices in the data array(s) corresponding to each site.
> &gt;
> &gt; Is there a quicker/better way to do this? Repeatedly calling WHERE seems inefficient (certainly HISTOGRAM is way faster when it is usable)
>
> Use VALUE_LOCATE to find where in the list of unique indices the
> elements belong to, and use that index as a number that you can run
> HISTOGRAM on.
>
> (raise your hand everyone who saw that coming...)
>
> -Jeremy.

Not me.  I had no idea VALUE_LOCATE works on strings.  Now that is cool!

---

## Subject: Re: Efficiently perform histogram reverse indices like procedure on a string array?

Posted by Jeremy Bailin on Thu, 26 Jul 2012 17:41:08 GMT

View Forum Message <> Reply to Message

>> Use VALUE_LOCATE to find where in the list of unique indices the
>> elements belong to, and use that index as a number that you can run
>> HISTOGRAM on.
>>
>> (raise your hand everyone who saw that coming...)
>>
>> -Jeremy.
>

>  Not me.  I had no idea VALUE_LOCATE works on strings.  Now that is cool!

Yup, it works on anything that can be sorted.

-Jeremy.

---

Subject: Re: Efficiently perform histogram reverse indices like procedure on a string array?
Posted by Craig Markwardt on Thu, 26 Jul 2012 21:33:51 GMT
View Forum Message <> Reply to Message

On Wednesday, July 25, 2012 7:39:33 PM UTC-4, Bogdanovist wrote:
>  I have an array of a data structure, one tag of which is a string identifier indicating which location the data belongs to. There are many thousands of data points, but only about a dozen or so unique locations.
>
>  I make frequent use of the HISTOGRAM function with the reverse_indices in order to carve up data by some identifier, most commonly the time. In this case, I want to divide out the data by site efficiently. I can&#39;t use HISTOGRAM on strings, so I need some other approach. There are plenty of ways this can be done, but I&#39;d like some views on the better and most efficient ways to do it.
>
>  Take an example, say we have a simple string array
>
>     foo=[&#39;a&#39;,&#39;b&#39;,&#39;c& #39;,&#39;b&#39;,&#39;b&#39;,&#39;a& #39;,&#39;a&#39;,&#39;c&#39;]
>
>  To determine the list of unique strings we could do
>
>    sfoo = foo[sort(foo)]
>    print,sfoo[uniq(sfoo)]
>
>  We can then repeatedly use WHERE to find the indices in the data array(s) corresponding to each site.
>
>  Is there a quicker/better way to do this? Repeatedly calling WHERE seems inefficient (certainly HISTOGRAM is way faster when it is usable)

I prefer to do it slightly differently than your other suggestions.

I locate the breakpoints between different runs of strings like this,

```
 ibreaks = where(sfoo[1:*] NE sfoo, ct)
```

This gives the interior breakpoints.  In your case, ibreaks = [2,5], which is the point where 'a' changes to 'b', and 'b' changes to 'c'.  Usually I add this little bit of extra post-processing,

```
 if ct EQ 0 then begin
    ibreaks = [0, n_elements(sfoo)]
 endif else begin
    ibreaks = [0, ibreaks+1, n_elements(sfoo)]
 endelse
```

You need that little extra 'if' statement to handle the case where you have only one unique string, so there are no breaks at all.

The start of the ith run is indexed by ibreaks[i], and the end of the ith run is indexed by ibreaks[i+1]-1, where i goes from 0 through n_elements(sfoo)-1.

I.e. the ith run is given by sfoo[ibreaks[i]:ibreaks[i+1]-1].  Of course you can index back into the original array once you've done this.

Craig