

---

Subject: Re: Need Some Advice on Seperating Out Some Data  
Posted by [rdellsy](#) on Fri, 11 Aug 2006 20:44:22 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Sorry for the double post.

Okay, I'm in the home stretch here. I'm just trying to figure out how to use a WHERE command or a histogram to select only the data that lies below a given line specified in  $y=mx+b$  form.

Thanks for all ya'll's help.

- Rob

rdellsy@gmail.com wrote:

```
> Interesting. I found that if I take the 4th power of the y variable, I
> get a few decent clusters. I was planning on doing a linear fit of
> those good clusters, and then using that to find the trailing data. Add
> in a small histogram to find the lower x bound of what I want and it
> should work. Your idea also sounds promissing. I'm getting close.
> Hopefully, I'll have something that'll work by Tuesday, and I'll let
> you all tear it to pieces...err find more efficient ways of doing it.
> =)
> Thanks,
> - Rob
```

```
>
>
```

```
> JD Smith wrote:
```

```
>> On Fri, 11 Aug 2006 10:22:35 -0700, kuyper wrote:
```

```
>>
```

```
>>> rdellsy@gmail.com wrote:
```

```
>>>> I'm working on doing a cluster tree and getting say the lower-right
>>>> cluster and the one or two nearest neighbors (sp?). I may still be
>>>> loosing some data though. Another possibilty would be compressing the
>>>> data, say, by half, and see if that helps.
>>>> Thanks,
>>>> Rob
```

```
>>>
```

```
>>> IDL> help,data
```

```
>>> DATA          FLOAT    = Array[2, 681]
```

```
>>>
```

```
>>> If all of the dimensions of your data have the same physical meaning,
>>> then
```

```
>>> you don't need to do anything to your data. However, I got the
>>> following
```

```
>>> results:
```

```
>>>
```

```
>>> IDL> print,stddev(data[0,*]),stddev(data[1,*])
```

```
>>>      2748.5689      1.7135388
```

```
>>>
```

```
>>> Which implies to me that your x and y coordinates probably have
```

```

>>> drastically
>>> different meanings, so they need to be scaled to have a meaningful
>>> distance
>>> measurement. The simplest way is to base the scale factors on the
>>> standard deviations:
>>>
>>> IDL> scaled = data
>>> IDL> scaled[0,*] /= stddev(data[0,*])
>>> IDL> scaled[1,*] /= stddev(data[1,*])
>>>
>>> I recommend, since you're analyzing many different but comparable
>>> datasets, to use a single scaling factor on each axis for all the
>>> datasets; otherwise it will be difficult to compare your results
>>> between one dataset and another.
>>>
>>> IDL> pairdistance = DISTANCE_MEASURE(scaled)
>>> IDL> clusters =
>>> CLUSTER_TREE(pairdistance,linkdistance,LINKAGE=0,data=scaled )
>>>
>>> I'm surprised by the fact that I haven't been able to locate an IDL
>>> function or procedure for taking the output from CLUSTER_TREE and using
>>> it to determine cluster membership at the point
>>> when there are N clusters left, so I wrote my own:
>>>
>>> FUNCTION cluster_member, clusters
>>>   dims = SIZE(clusters, /DIMENSIONS)
>>>   num = dims[1] + 1
>>>   membership = INTARR(num, num-1)
>>>   work = indgen(num)
>>>   FOR i=0, num-2 DO BEGIN
>>>     newclust = WHERE (work eq clusters[0,i] OR work EQ
>>> clusters[1,i])
>>>     work[newclust] = num+i
>>>     membership[0,i] = work
>>>   ENDFOR
>>>
>>>   RETURN, membership
>>> END
>>>
>>> There's probably a more efficient way of handling that loop.
>>
>> Very cool! I'll have to remember this one. If you only care about n
>> remaining clusters, you can simplify somewhat to:
>>
>> function cluster_member, clusters,n
>>   dims = SIZE(clusters, /DIMENSIONS)
>>   num = dims[1] + 1L
>>   n>=1

```

```
>> work = lindgen(num)
>> for i=0L, num-1L-n do $
>>     work[where(work eq clusters[0,i] OR work eq clusters[1,i])]= num+i
>> return, work
>> end
>>
>> JD
```

---