Subject: Re: Avoiding multiple FOR loops
Posted by Gray on Sat, 19 Mar 2011 11:06:43 GMT
View Forum Message <> Reply to Message

On Mar 19, 12:35 am, Jeremy Bailin <astroco...@gmail.com> wrote:
> The problem, I think, is the regress statement. Everything else can be completely vectorized, but you're going to have to loop through all the combinations and run the regression on each. Still, this may save time:
>
> 1. Generate the list of combinations. E.g.
>
> nper = indgen(nvar) + maxn - nvar
> vars = lindgen(nper)
> vars_ai = array_indices(vars, vars) ; all permutations with duplicates
> goodcombip = vars_ai[0:nvar-2,*] lt vars_ai[1:*,*]
> goodcombi = where(total(goodcombip, 1, /int) eq nvar-1, ngoodcombi)
> vars_ai = vars_ai[*, goodcombi] ; all unique combinations
>
> (note: I'm sure there's a better way to do this, but it's not going to be the limiting step so this is probably good enough)
>
> 2. Collect the data
>
> x = data[vars_ai, *]
>
> 3. Loop through the regressions and store yfit.
>
> ndata = (size(x, /dimen))[1]
> yfits = fltarr(ndata, ngoodcombi]
> for i=0l, ngoodcombi-1 do begin
>   coef = regress(x[i*nvar:(i+1)*nvar-1, *], y, const=const, yfit=yfit)
>   yfits[*,i] = yfit
> endfor
>
> 4. Calculate rms.
>
> rms = sqrt(total((rebin(y,ndata,ngoodcombi,/sample) - yfits)^2, 1) / ndata)
>
> Note: totally untested and totally untimed, so I don't know if this is any faster. My gut feeling is that having one for loop that has a minimal amount of code in it is going to be faster than 6 nested for loops that have a few more lines in the meat, but I don't know by how much and it certainly won't be nearly as fast as if regress were vectorized.
>
> -Jeremy.

Any way you can use mpfit?