Notes on extending yags working correlation structures

yags 3.16 includes a `"user"` `corstruct` option. When this option is encountered, yags will use the routines `alpfun_user` and `wcorinv_user` to estimate α and $R^{-1}(\alpha)$ on the basis of current Pearson residuals and other structural information. $R^{-1}(\hat{\alpha})$ is plugged in to the next iteration of GEE estimation of regression parameters; these are then used to recompute Pearson residuals for iteration to convergence.

A user-defined working correlation structure routine must be a C++ program with the following structure; `matrix` is a C++ class provided by the MC++ library included with the package.

⟨*user-defined wcor routine*⟩≡

```
⟨C++ headers; optional addtl decls⟩

matrix alpfun_user( ⟨alpfun arglist⟩ )
 ⟨body of estimating function evaluator for alpha⟩

matrix wcorinv_user( ⟨wcorinv arglist⟩ )
 ⟨body of working correlation inverse routine⟩

⟨additional code⟩
```

If the bodies or additional code require additional C++ routines not known to MC++, those routines will have to be correctly declared within the C++ file.

Details on the alpfun evaluator routine

At the present version (3.16) alpfun handling is very limited. The program must return a matrix representing a q-vector of current estimates of α based on information in the first 6 arguments. In future versions the user may pass an *estimating function* whose zero will be sought by grid/secant search methods that use the last 2 arguments:

⟨*alpfun arglist*⟩≡

```
        matrix PRin,  /* pearson residuals */
        matrix ID,    /* cluster discriminator */
        matrix TIMin,  /* coordinate matrix */
        double phi,    /* scale parameter */
        int p,         /* regression dimension */
        matrix alpin,  /* initial value of alpha */
        double atol,   /* convergence criterion */
        int amaxit    /* maximum number of iterations */
```

A simple example of an `alpfun` that works for a fixed known value of α is:

⟨*body of estimating function evaluator for alpha*⟩≡

```
 {return alpin;}  /* trivial function that returns the
                     initializing value */
```

Details on the wcorinv routine

The `wcorinv_user` function actually returns the inverse for the i^{th} cluster. Thus heterogeneous correlation models are supported; stratum information may be passed through the **`tim`** argument. This will typically be a 1-column matrix of observation times but can be any matrix with n_i rows.

⟨*wcorinv arglist*⟩≡

```
        matrix alp, /* current q-vector alpha hat */
        int ni,     /* size of this cluster */
        matrix tim  /* structural information on coordinates
                       of this cluster's elements */
```

The following fragment shows several simple ways of using the `wcorinv_user` functionality (two are commented out but tested). First, the unnecessary step of inverting an identity matrix is demonstrated to be sure that the independence working model results are recovered. Then a fixed matrix structure (exchangeable) with variable parameter is demonstrated. Finally a damped exponential structure is demonstrated. The latter two examples require additional code.

⟨*body of working correlation inverse routine*⟩≡

```
   {
    /* return sweep(ident(ni)); */
    /* return sweep(fixed_exch( alp, ni )); */
    return sweep(dec( alp, ni, tim ));
   }
```

⟨*additional code*⟩≡

```
matrix fixed_exch( matrix alp, int ni )
  {
  matrix x= newmat(ni, ni);
  for (int i = 0; i < ni ; i++ )
    {
    set_el(x,i,i) = 1.0;
    for (int j = i+1; j < ni; j++ )
       {
       set_el(x,i,j) = alp.el(0,0);
       set_el(x,j,i) = alp.el(0,0);
       }
    }
  return(x);
  }

matrix dec(matrix alp, int ni, matrix tim)
 {
// damped exponential correlation
 matrix out = newmat(ni,ni);
 for (int i = 0; i < ni; i++ )
  {
  set_el(out,i,i) = 1.0;
  for (int j = i+1; j < ni; j++ )
    {
    double d = fabs( tim.el(j,0) - tim.el(i,0) );
    set_el(out,i,j) = pow( alp.el(0,0), pow( d, alp.el(1,0) ) );
    set_el(out,j,i) = out.el(i,j);
    }
  }
 return out;
 }
```

⟨*C++ headers; optional addtl decls*⟩≡

```
#include "MC++.h"
#include "MC++class.h"

matrix dec( matrix, int, matrix );
matrix fixed_exch( matrix, int );
```

⟨*demo_wcor_user.cc*⟩≡

  ⟨*user-defined wcor routine*⟩

1 Administration

⟨*EMakefile*⟩≡

```
TOPIC = extyags
EMakefile: $(TOPIC).nw
        notangle -t8 -R"EMakefile" $(TOPIC).nw > EMakefile

html %.html: $(TOPIC).nw
        noweave -filter l2h -latex+html -index -html $(TOPIC).nw |htmltoc> $(TOPIC).htm

tex:
        noweave $(TOPIC).nw > $(TOPIC).tex

edit:
        vi $(TOPIC).nw

tt.cc:
        notangle -t8 -R"demo\_wcor\_user.cc" $(TOPIC).nw > tt.cc
```