Delayed Data Packages

The g.data package

by David E. Brahm

Data storage in R and S-Plus

The biggest shock for me in transitioning from S-Plus to R was the different data storage model. In S-Plus, each position in the search path corresponds to a directory ("chapter") on your disk, and every object is written immediately to disk upon creation, as a file of the same name as the object (under Unix). In R, positions on the search path are "environments", and objects live—and die—in memory unless you `save` them explicitly. The purpose of the `g.data` package is to provide data storage in R in a way that combines the best features of both models, including:

- Storing objects in individual files, without multiple `save`'s
- Viewing the contents of an attached directory without pulling them all into memory
- Loading objects into memory as they're needed, without multiple `load`'s.

For example, I store twenty large (date $\times$ stock) matrices `m1`, ..., `m20` in a "delayed data package" (DDP) called `hist`. They belong together, because they cover the same date and stock ranges, so I can calculate e.g. `mnew <- m1 * m2`. But in a given session, I may only care about two of the twenty matrices, so I don't want to load all twenty into memory. I also don't want to have to type a `load` command for every matrix I need. When I'm done, I may want to save `mnew` into the DDP for future use. And then I want to detach that DDP and attach another (with different dates and stocks), to perform the same operations there.

Here's how that looks with the `g.data` package (assuming the DDP already exists):

```
> require(g.data)
> g.data.attach("/rdata/hist")
> assign("mnew", m1*m2, 2)
> g.data.save()
> detach(2)
```

More examples

Here's an example that creates a database from scratch.

```
> g.data.attach("/tmp/newdir", warn=FALSE)
> assign("x1", matrix(1, 1000, 1000), 2)
> assign("x2", matrix(2, 1000, 1000), 2)
> g.data.save()
> detach(2)
```

In the next example, the first timed command takes a while, because `x1` is being loaded. The second is quick, because now `x1` is in memory.

```
> g.data.attach("/tmp/newdir")
> system.time(print(dim(x1)))
[1] 1000 1000
[1] 1.70 0.01 1.90 0.00 0.00
> system.time(print(dim(x1)))
[1] 1000 1000
[1] 0 0 0 0 0
```

Suppose you now type

```
> assign("x3", x2*10, 2)
```

and go look at the contents of '/tmp/newdir'. Objects `x1` and `x2` are written there (under subdirectory 'data', with names 'x1.RData' and 'x2.RData'), but `x3` is not. 'x3.RData' only gets written when you type `g.data.save()`. Unlike S-Plus, you have the option *not* to save the changes you've made, just by not calling `g.data.save`.

With no arguments, `g.data.save` (re-)writes all objects in position 2 to disk. Both `g.data.attach` and `g.data.save` take an optional argument `pos` (defaulting to `pos=2`), so you can work with multiple DDP's in different positions on the search path. `g.data.save` also takes an argument `obj` to (re-)write only specified objects, an argument `rm.obj` to remove objects, and an argument `dir` which can be used e.g. in this context:

```
> y <- list(a=1, b=11:15, c=21:29)
> attach(y, pos=4)
> g.data.save(dir="/tmp/mylist", pos=4)
```

Finally, the command `g.data.get` allows you to retrieve individual objects stored in a DDP without attaching the DDP:

```
> bcopy <- g.data.get("b", "/tmp/mylist")
> bcopy
[1] 11 12 13 14 15
```

Under the hood

`g.data.save` creates (or writes to) a DDP directory with subdirectories 'R' and 'data', so it looks to R much like a typical package. `g.data.attach` is really just a snippet of `library`. The data files in subdirectory 'data' are created with `save`, as you'd expect. The code under subdirectory 'R' consists of lines like:

```
x1 <- delay(g.data.load("x1", "newdir"))
```

so when you first `g.data.attach` the directory, the object loaded into memory as `x1` is a "promise object" (which is very small). When you actually use `x1` (e.g., to query its dimensions), the promise is fulfilled, and `g.data.load` does two things:

1. It loads the actual large object, and
2. It returns that object for the query.

Henceforth, the object in memory as `x1` is the real (large) object.

David E. Brahm
Geode Capital Management
brahm@alum.mit.edu

geepack: Yet Another Package for Generalized Estimating Equations

Modeling Both Mean and Association of Multivariate Responses

by Jun Yan

Introduction

geepack is designed to provide an inferential basis for both the association structure and the mean structure in multivariate analysis, using the Generalized Estimating Equations (GEE) approach.

Consider a sample of K independent clusters $y_i^T = (y_{i1}, \cdots, y_{in_i})$, $i = 1, \cdots, K$, of n_i-variate responses. In a generalized linear model setup, the variance of y_{it}, V_{it}, can be factored as

$$\text{var}(y_{it}) = \phi_{it} v(\mu_{it}),$$

where ϕ_{it} is the scale parameter, v is the variance function $v(\mu_{it})$, where $\mu_{it} = E(y_{it})$. To model the association, we decompose $\text{cov}(y_i)$ into two parts, the variance and the correlation,

$$\text{cov}(y_i) = V^{1/2} R V^{1/2},$$

where V is the diagonal matrix of V_{it}, and R is the correlation matrix of y_i.

Let X_{1i}, X_{2i} and X_{3i} be the covariate matrices for the mean, the scale, and the correlation of the response y_i, with dimensions $n_i \times p$, $n_i \times r$, and $n_i(n_i - 1)/2 \times q$, respectively. The models are

$$g_1(\mu_i) = X_{1i}\beta, \quad (1)$$
$$g_2(\phi_i) = X_{2i}\gamma, \quad (2)$$
$$g_3(\rho_i) = X_{3i}\alpha, \quad (3)$$

where $g_i, i = 1, 2, 3$, are known link functions, μ_i is a $n_i \times 1$ vector containing $E(y_i|X_{1i})$, ϕ_i is a $n_i \times 1$ vector containing $\text{var}(y_i|X_{2i})/v_{it}$, where $v_{it} = v(\mu_{it})$ is the variance function, and ρ_i is a $n_i(n_i - 1)/2 \times 1$ vector containing $\text{cor}(y_{is}, y_{it}|X_{3i})$. β, γ, and α are the mean, the scale, and the correlation parameters of dimension $p \times 1$, $r \times 1$, and $q \times 1$, respectively.

The mean link has been well studied. The scale link is often taken to be log, while it is natural to let the correlation link be "logistic" (i.e., Fisher's z transformation), in which case the inverse link function is the hyperbolic tangent, that is,

$$\rho_{its} = \text{cor}(y_{is}, y_{it}|X_{3i}) = \frac{\exp(X_{3i(s,t)}\alpha) - 1}{\exp(X_{3i(s,t)}\alpha) + 1}, \quad (4)$$

where $X_{3i(s,t)}$ is the row in matrix X_{3i} corresponding to the correlation of y_{is} and y_{it}. These links ensure that the scale is positive and that the correlation is in $(-1, 1)$. The scale model is useful in situations where parameters are needed for covariate effects either on over- or under-dispersion or on heteroscedasticity.

A convenient set of estimating equations for the three-link model is

$$U_1(\beta, \gamma, \alpha) = \sum_{i=1}^{K} D_{1i}^T V_{1i}^{-1}(y_i - \mu_i) = 0 \quad (5)$$
$$U_2(\beta, \gamma, \alpha) = \sum_{i=1}^{K} D_{2i}^T V_{2i}^{-1}(s_i - \phi_i) = 0 \quad (6)$$
$$U_3(\beta, \gamma, \alpha) = \sum_{i=1}^{K} D_{3i}^T V_{3i}^{-1}(z_i - \rho_i) = 0 \quad (7)$$

where s_i is the $n_i \times 1$ vector of $s_{it} = (y_{it} - \mu_{it})^2/v_{it}$, z_i is the $n_i(n_i - 1)/2 \times 1$ vector of $z_{its} = (y_{it} - \mu_{it})(y_{is} - \mu_{is})/\sqrt{\phi_{it} v_{it} \phi_{is} v_{is}}$, $D_{1i} = \partial\mu_i/\partial\beta^T$, $D_{2i} = \partial\phi_i/\partial\gamma^T$, $D_{3i} = \partial\rho_i/\partial\alpha^T$, and V_{1i}, V_{2i} and V_{3i} are the conditional working covariance matrices of y_i, s_i, and z_i.

The matrix V_{1i} generally contains scale parameters γ and correlation parameters α. The matrices V_{2i} and V_{3i} may contain other estimated quantities which characterize the third and fourth order moments. In practice, in order to avoid specification of higher order moments, estimation of higher order nuisance parameters, and convergence problems, V_{2i} may be chosen to be a diagonal matrix whose diagonal elements are $2\phi_{it}$, following the independence Gaussian working matrix in Prentice and Zhao (1991), and V_{3i} may be an identity matrix (Ziegler et al., 1998, p.129), at the cost of potential efficiency