On the usage of the geepack

Søren Højsgaard and Ulrich Halekoh

geepack version 1.2-0 as of 2014-09-13

1 Introduction

The primary reference for the geepack package is the Halekoh, U., Højsgaard, S., Yan, J. (2006) – paper in Journal of Statistical Software, see

```
> library(geepack)
> citation("geepack")

To cite geepack in publications use:

  Højsgaard, S., Halekoh, U. & Yan J. (2006) The R Package geepack for Generalized
  Estimating Equations Journal of Statistical Software, 15, 2, pp1--11

  Yan, J. & Fine, J.P. (2004) Estimating Equations for Association Structures
  Statistics in Medicine, 23, pp859--880.

  Yan, J (2002) geepack: Yet Another Package for Generalized Estimating Equations
  R-News, 2/3, pp12-14.
```

If you use geepack in your own work, please do cite the above reference.

This note contains a few extra examples. We illustrate the usage of a the `waves` argument and the `zcor` argument together with a fixed working correlation matrix for the `geeglm()` function. To illustrate these features we simulate some data suitable for a regression model.

```
> library(geepack)
> timeorder <- rep(1:5, 6)
> tvar      <- timeorder + rnorm(length(timeorder))
> idvar <- rep(1:6, each=5)
> uuu   <- rep(rnorm(6), each=5)
> yvar  <- 1 + 2*tvar + uuu + rnorm(length(tvar))
> simdat <- data.frame(idvar, timeorder, tvar, yvar)
> head(simdat,12)

   idvar timeorder        tvar      yvar
1      1         1  2.78691314  5.422171
2      1         2  2.49785048  4.829722
3      1         3  1.03338284  3.312259
4      1         4  4.70135590  9.053832
5      1         5  4.52720859 10.447779
6      2         1 -0.06782371 -1.087285
7      2         2  1.78202509  4.745779
8      2         3  1.97399555  4.668961
9      2         4  3.27110877  7.355274
10     2         5  4.37496073  9.726676
11     3         1 -0.68669331 -1.342365
12     3         2  2.83778704  5.875711
```

Notice that clusters of data appear together in `simdat` and that observations are ordered (according to `timeorder`) within clusters.

We can fit a model with an AR(1) error structure as

```
> mod1 <- geeglm(yvar~tvar, id=idvar, data=simdat, corstr="ar1")
> mod1

Call:
geeglm(formula = yvar ~ tvar, data = simdat, id = idvar, corstr = "ar1")

Coefficients:
(Intercept)        tvar
  0.4399188   2.1106751

Degrees of Freedom: 30 Total (i.e. Null);  28 Residual

Scale Link:                  identity
Estimated Scale Parameters:  [1] 1.136851

Correlation:  Structure = ar1    Link = identity
Estimated Correlation Parameters:
    alpha
0.5346614

Number of clusters:   6   Maximum cluster size: 5
```

This works because observations are ordered according to time within each subject in the dataset.

2 Using the `waves` argument

If observatios were not ordered according to cluster and time within cluster we would get the wrong result:

```
> set.seed(123)
> ## library(doBy)
> simdatPerm <- simdat[sample(nrow(simdat)),]
> ## simdatPerm <- orderBy(~idvar, simdatPerm)
> simdatPerm <- simdatPerm[order(simdatPerm$idvar),]
> head(simdatPerm)

  idvar timeorder     tvar      yvar
2     1         2 2.497850  4.829722
4     1         4 4.701356  9.053832
1     1         1 2.786913  5.422171
3     1         3 1.033383  3.312259
5     1         5 4.527209 10.447779
9     2         4 3.271109  7.355274
```

Notice that in `simdatPerm` data is ordered according to subject but the time ordering within subject is random.

Fitting the model as before gives

```
> mod2 <- geeglm(yvar~tvar, id=idvar, data=simdatPerm, corstr="ar1")
> mod2

Call:
geeglm(formula = yvar ~ tvar, data = simdatPerm, id = idvar,
    corstr = "ar1")

Coefficients:
(Intercept)        tvar
  0.1969892   2.2155856

Degrees of Freedom: 30 Total (i.e. Null);  28 Residual

Scale Link:                   identity
Estimated Scale Parameters:  [1] 1.123093

Correlation:  Structure = ar1    Link = identity
Estimated Correlation Parameters:
  alpha
0.51013

Number of clusters:   6   Maximum cluster size: 5
```

Likewise if clusters do not appear contigously in data we also get the wrong result (the clusters are not recognized):

```
> ## simdatPerm2 <- orderBy(~timeorder, data=simdat)
> simdatPerm2 <- simdat[order(simdat$timeorder),]
> geeglm(yvar~tvar, id=idvar, data=simdatPerm2, corstr="ar1")

Call:
geeglm(formula = yvar ~ tvar, data = simdatPerm2, id = idvar,
    corstr = "ar1")

Coefficients:
(Intercept)        tvar
  0.2589139   2.1828479

Degrees of Freedom: 30 Total (i.e. Null);  28 Residual

Scale Link:                   identity
Estimated Scale Parameters:  [1] 1.118158

Correlation:  Structure = ar1    Link = identity
Estimated Correlation Parameters:
alpha
    0

Number of clusters:   30   Maximum cluster size: 1
```

To obtain the right result we must give the `waves` argument:

```
> wav <- simdatPerm$timeorder
> wav

 [1] 2 4 1 3 5 4 5 2 1 3 2 3 4 5 1 5 4 2 1 3 3 4 5 1 2 2 5 4 1 3

> mod3 <- geeglm(yvar~tvar, id=idvar, data=simdatPerm, corstr="ar1", waves=wav)
> mod3

Call:
geeglm(formula = yvar ~ tvar, data = simdatPerm, id = idvar,
    waves = wav, corstr = "ar1")

Coefficients:
(Intercept)        tvar
  0.4399188   2.1106751

Degrees of Freedom: 30 Total (i.e. Null);  28 Residual

Scale Link:                  identity
Estimated Scale Parameters:  [1] 1.136851

Correlation:  Structure = ar1    Link = identity
Estimated Correlation Parameters:
    alpha
0.5346614

Number of clusters:   6   Maximum cluster size: 5
```

3 Using a fixed correlation matrix and the `zcor` argument

Suppose we want to use a fixed working correlation matrix:

```
> cor.fixed <- matrix(c(1    , 0.5  , 0.25,  0.125, 0.125,
+                       0.5  , 1    , 0.25,  0.125, 0.125,
+                       0.25 , 0.25 , 1   ,  0.5  , 0.125,
+                       0.125, 0.125, 0.5  , 1    , 0.125,
+                       0.125, 0.125, 0.125, 0.125, 1     ), 5, 5)
> cor.fixed

      [,1]  [,2]  [,3]  [,4]  [,5]
[1,] 1.000 0.500 0.250 0.125 0.125
[2,] 0.500 1.000 0.250 0.125 0.125
[3,] 0.250 0.250 1.000 0.500 0.125
[4,] 0.125 0.125 0.500 1.000 0.125
[5,] 0.125 0.125 0.125 0.125 1.000
```

Such a working correlation matrix has to be passed to `geeglm()` as a vector in the `zcor` argument. This vector can be created using the `fixed2Zcor()` function:

```
> zcor <- fixed2Zcor(cor.fixed, id=simdatPerm$idvar, waves=simdatPerm$timeorder)
> zcor

 [1] 0.125 0.500 0.250 0.125 0.125 0.500 0.125 0.250 0.125 0.125 0.125 0.125 0.125 0.500 0.125
[16] 0.125 0.125 0.500 0.250 0.250 0.250 0.125 0.125 0.500 0.500 0.125 0.250 0.125 0.125 0.125
[31] 0.125 0.125 0.125 0.125 0.125 0.125 0.500 0.500 0.250 0.250 0.500 0.125 0.250 0.250 0.125
[46] 0.125 0.125 0.125 0.125 0.500 0.125 0.125 0.500 0.250 0.125 0.125 0.125 0.125 0.500 0.250
```

Notice that `zcor` contains correlations between measurements within the same cluster. Hence if a cluster contains only one observation, then there will be generated no entry in `zcor` for that cluster. Now we can fit the model with:

```
> mod4 <- geeglm(yvar~tvar, id=idvar, data=simdatPerm, corstr="fixed", zcor=zcor)
> mod4

Call:
geeglm(formula = yvar ~ tvar, data = simdatPerm, id = idvar,
    zcor = zcor, corstr = "fixed")

Coefficients:
(Intercept)        tvar
  0.4692344   2.1148665

Degrees of Freedom: 30 Total (i.e. Null);  28 Residual

Scale Link:                   identity
Estimated Scale Parameters:  [1] 1.133532

Correlation:  Structure = fixed    Link = identity
Estimated Correlation Parameters:
alpha:1
      1

Number of clusters:   6   Maximum cluster size: 5
```