

On the usage of the `geepack`

Søren Højsgaard and Ulrich Halekoh

`geepack` version 1.3-1 as of 2019-12-12

Contents

1	Introduction	1
2	Citing <code>geepack</code>	1
3	Simulating a dataset	2
4	Using the <code>waves</code> argument	3
5	Using a fixed correlation matrix and the <code>zcor</code> argument	5
6	When do GEE's work best?	6

1 Introduction

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.

2 Citing `geepack`

The primary reference for the `geepack` package is

Halekoh, U., Højsgaard, S., Yan, J. (2006) *The R Package `geepack` for Generalized Estimating Equations (2006)* Journal of Statistical Software
<https://www.jstatsoft.org/article/view/v015i02>

```

> 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.

To see these entries in BibTeX format, use 'print(<citation>,
bibtex=TRUE)', 'toBibtex(.)', or set
'options(citation.bibtex.max=999)'.

```

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

3 Simulating a dataset

To illustrate the usage of the `waves` argument and the `zcor` argument together with a fixed working correlation matrix for the `geeglm()` 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	1.50601090	2.038988
2	1	2	3.10709767	7.084939
3	1	3	3.39502382	7.227487
4	1	4	3.42457877	6.626946
5	1	5	3.24963360	4.659393
6	2	1	-0.02111803	0.668086
7	2	2	0.61245078	3.903763
8	2	3	1.67391591	6.228827
9	2	4	2.95798747	6.703759
10	2	5	4.92283604	10.663722
11	3	1	0.98271082	3.518166
12	3	2	2.58970219	7.912803

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
    1.191122     2.052764

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

Scale Link:              identity
Estimated Scale Parameters: [1] 2.140089

Correlation: Structure = ar1   Link = identity
Estimated Correlation Parameters:
      alpha
0.4533905

Number of clusters: 6   Maximum cluster size: 5

```

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

4 Using the waves argument

If observations 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
3     1         3 3.395024 7.227487
5     1         5 3.249634 4.659393
4     1         4 3.424579 6.626946
1     1         1 1.506011 2.038988
2     1         2 3.107098 7.084939
10    2         5 4.922836 10.663722

```

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
  1.507896    1.988578

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

Scale Link:              identity
Estimated Scale Parameters: [1] 2.145164

Correlation: Structure = ar1   Link = identity
Estimated Correlation Parameters:
alpha
0.5500409

Number of clusters:  6   Maximum cluster size: 5

```

Likewise if clusters do not appear contiguously 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
  1.248357    2.071333

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

Scale Link:              identity
Estimated Scale Parameters: [1] 2.125603

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

> 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
  1.191122      2.052764

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

Scale Link:          identity
Estimated Scale Parameters: [1] 2.140089

Correlation: Structure = ar1   Link = identity
Estimated Correlation Parameters:
  alpha
0.4533905

Number of clusters: 6   Maximum cluster size: 5

```

5 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 0.250 0.125 0.125 0.125 0.125 0.125 0.500 0.125 0.125
[13] 0.125 0.125 0.500 0.125 0.125 0.250 0.250 0.500 0.125 0.125 0.125 0.125
[25] 0.125 0.500 0.125 0.250 0.500 0.250 0.500 0.125 0.125 0.125 0.125 0.250
[37] 0.250 0.125 0.125 0.500 0.125 0.125 0.250 0.500 0.125 0.500 0.125 0.125
[49] 0.125 0.250 0.250 0.250 0.125 0.500 0.500 0.125 0.125 0.125 0.125 0.125

```

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
  1.097249    2.093038

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

Scale Link:              identity
Estimated Scale Parameters: [1] 2.133702

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

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

6 When do GEE's work best?

GEEs work best when you have relatively many relatively small clusters in your data.