Generalized Estimating Equations

Ashley I Naimi

Spring 2024

Contents

1 Introduction

Generalized estimating equations are a technique for handling correlated outcome data that were introduced in 1986. The original paper introducing these methods was published in *Biometrika* by Kung Yi Liang and Scott Zeger (Liang and Zeger, 1986). They proposed an "extension" to generalized linear models that enables handling correlated outcomes.

But why was there a need to "extend" GLMs in the first place? To illustrate, suppose we had a dataset of 400 observations with one continuous outcome and one 50:50 randomized binary exposure. For example, let's look at the first ten observations of a dataset from a double-blind placebo controlled trial to see if a chemical chelating agent can successfully lower blood lead concentrations in children aged 12 to 36 months, with a total of 400 observations:

```
lead_data <- read_table("https://content.sph.harvard.edu/fitzmaur/ala/tlc.txt",
    col_names = F, skip = 29)
names(lead_data) <- c("ID", "treatment", "L0", "L1", "L4", "L6")

lead_data <- gather(lead_data, week, lead_value, L0:L6, factor_key = TRUE) %>%
    mutate(week = as.numeric(gsub("L", "", week)), treatment = as.numeric(treatment ==
        "A")) %>%
    arrange(runif(400)) %>%
    select(treatment, lead_value)

lead_data %>%
    print(n = 10)
```

```
## # A tibble: 400 x 2
##    treatment lead_value
##        <dbl>      <dbl>
##  1         0       20.5
##  2         0       27.9
##  3         0       17.4
##  4         1       31.2
##  5         0       29.2
##  6         1       11.9
```

```
##  7         0       26.2
##  8         1        6.5
##  9         1        8.6
## 10         1       20.2
## # i 390 more rows
```

Suppose, without knowing any more than this, we want to use GLMs to estimate the effect of treatment assignment on the outcome. We might fit a GLM that looks something like this:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, and use maximum likelihood estimation to quantify the parameters (β_0, β_1). The R code for this would look something like:

```
mod1 <- glm(lead_value ~ treatment, data = lead_data, family = gaussian("identity"))

summary(mod1)$coefficients
```

```
##             Estimate Std. Error   t value      Pr(>|t|)
## (Intercept)  24.6620  0.5318026 46.374352 1.451291e-162
## treatment    -5.5775  0.7520824 -7.416075  7.338470e-13
```

which suggests that, relative to placebo, the chemical chelating agent reduces blood lead concentrations by -5.58 $\mu g/dL$, with a standard error of this estimate of 0.75 $\mu g/dL$. With this point estimate and standard error, we could construct a whole set of inferential statistics, such as 95% confidence intervals and p-values, each of which have specific and mathematically rigorous interpretations.

For example, we might want to interpret the p-value for the treatment effect coefficient from the model and state that, if the null hypothesis were, in fact, true, the probability of observing a result as extreme as -5.58 $\mu g/dL$ or more extreme is less than 0.0001.

However, in order for this interpretation to work, the errors ϵ_i (or, equivalently, the outcomes Y_i conditional on X_i) must be **independent** across all individuals i. If this independence assumption doesn't hold, the math proving that maximum likelihood estimation "works" breaks down, as does the interpre-

tation of our inferential statistics. As a consequence, we can no longer trust the results of an empirical analysis in the ways that we'd like too.

Unfortunately for us, this is a problem with our lead data. Although we have 400 observations in this dataset, they come from a total of 100 children. The blood lead concentrations were measured four times on each child, at week 0, 1, 4, and 6:

```
lead_data0 <- read_table("https://content.sph.harvard.edu/fitzmaur/ala/tlc.txt",
    col_names = F, skip = 29)
names(lead_data0) <- c("ID", "treatment", "L0", "L1", "L4", "L6")

lead_data <- gather(lead_data0, week, lead_value, L0:L6, factor_key = TRUE) %>%
    mutate(week = as.numeric(gsub("L", "", week)), treatment = as.numeric(treatment ==
        "A")) %>%
    arrange(ID, week)

lead_data %>%
    print(n = 10)
```

```
## # A tibble: 400 x 4
##       ID treatment  week lead_value
##    <dbl>     <dbl> <dbl>      <dbl>
##  1     1         0     0       30.8
##  2     1         0     1       26.9
##  3     1         0     4       25.8
##  4     1         0     6       23.8
##  5     2         1     0       26.5
##  6     2         1     1       14.8
##  7     2         1     4       19.5
##  8     2         1     6       21
##  9     3         1     0       25.8
## 10     3         1     1       23
## # i 390 more rows
```

```
mean(lead_data$treatment)
```

```
## [1] 0.5
```

```
lead_data %>%
    group_by(treatment) %>%
    summarize(meanLead = mean(lead_value), sdBMI = sd(lead_value),
        numTx = length(lead_value))
```

```
## # A tibble: 2 x 4
##   treatment meanLead sdBMI numTx
##       <dbl>    <dbl> <dbl> <int>
## 1         0     24.7  5.53   200
## 2         1     19.1  9.09   200
```

This suggests that within each individual child, the measurements are likely correlated. We can explore this informally using variance-covariance matrices, correlation matrices, or intracluster correlation coefficients:

```
lead_data0 <- read_table("https://content.sph.harvard.edu/fitzmaur/ala/tlc.txt",
    col_names = F, skip = 29)
names(lead_data0) <- c("ID", "treatment", "L0", "L1", "L4", "L6")

lead_data0 <- lead_data0 %>%
    select(L0, L1, L4, L6)

cov(lead_data0)
```

```
##          L0       L1       L4       L6
## L0 24.98905 18.16066 18.92146 21.78220
## L1 18.16066 75.22487 59.24104 37.48691
## L4 18.92146 59.24104 65.38539 36.54236
## L6 21.78220 37.48691 36.54236 60.15897
```

```
cor(lead_data0)
```

```
##           L0        L1        L4        L6
## L0 1.0000000 0.4188669 0.4681009 0.5617933
## L1 0.4188669 1.0000000 0.8446982 0.5572482
## L4 0.4681009 0.8446982 1.0000000 0.5826476
## L6 0.5617933 0.5572482 0.5826476 1.0000000
```

```
## compute intracluster correlation coefficient
lead_summary <- summary(aov(lead_value ~ as.factor(ID), data = lead_data))  # type II SS

icc <- lead_summary[[1]][1, 2]/sum(lead_summary[[1]][, 2])

icc
```

```
## [1] 0.5892469
```

2 Generalized Estimating Equations

All these metrics suggest that there is an important degree of dependence between observations measured within each child. So the question becomes, can we somehow adjust our generalized linear model to account for this lack of independence, and recover the interpretation of the p-values, confidence intervals, and standard errors of interest? This was the focus of the paper by Liang and Zeger (Liang and Zeger, 1986). Their "extension" did just that: adjusted the GLM by incorporating information on the correlation structure within individual (in this case) children. This extension generalized the GLM using a theory of estimating equations, and the new method was hence named generalized estimating equations.

The main distinction between deploying GEEs versus GLMs is that, in the former, we have to consider the structure of the correlation within units in our data. Several correlation structures exist, and include things like the indepencence, the exchangeable, the unstructured, or the variations of autoregression correlation matrices. In R, these methods can be deployed using the `geepack` library.

Data should be arranged in long form to use the GEE functions in R:

```
lead_data0 <- read_table("https://content.sph.harvard.edu/fitzmaur/ala/tlc.txt",
    col_names = F, skip = 29)
names(lead_data0) <- c("ID", "treatment", "L0", "L1", "L4", "L6")

lead_data <- gather(lead_data0, week, lead_value, L0:L6, factor_key = TRUE) %>%
    mutate(week = as.numeric(gsub("L", "", week)), treatment = as.numeric(treatment ==
        "A")) %>%
    arrange(ID, week)

lead_data %>%
    print(n = 10)
```

```
## # A tibble: 400 x 4
##       ID treatment  week lead_value
##    <dbl>     <dbl> <dbl>      <dbl>
##  1     1         0     0       30.8
##  2     1         0     1       26.9
##  3     1         0     4       25.8
##  4     1         0     6       23.8
##  5     2         1     0       26.5
##  6     2         1     1       14.8
##  7     2         1     4       19.5
##  8     2         1     6       21
##  9     3         1     0       25.8
## 10     3         1     1       23
## # i 390 more rows
```

These data can then be analyzed using the `geeglm` functions in the `geepack` library:

```
# install.packages('geepack')
library(geepack)

## use GEE
```

```r
mod1_ind <- geeglm(lead_value ~ treatment, id = factor(ID), data = lead_data,
    corstr = "independence")
summary(mod1_ind)  #$coefficients
```

```
##
## Call:
## geeglm(formula = lead_value ~ treatment, data = lead_data, id = factor(ID),
##     corstr = "independence")
##
##  Coefficients:
##             Estimate Std.err    Wald Pr(>|W|)
## (Intercept)  24.6620  0.7122 1198.94  < 2e-16 ***
## treatment    -5.5775  1.0949   25.95  3.5e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = independence
## Estimated Scale Parameters:
##
##             Estimate Std.err
## (Intercept)    56.28   7.401
## Number of clusters:   100  Maximum cluster size: 4
```

```r
mod1_exch <- geeglm(lead_value ~ treatment, id = factor(ID),
    data = lead_data, corstr = "exchangeable")
summary(mod1_exch)  #$coefficients
```

```
##
## Call:
## geeglm(formula = lead_value ~ treatment, data = lead_data, id = factor(ID),
##     corstr = "exchangeable")
##
##  Coefficients:
##             Estimate Std.err Wald Pr(>|W|)
## (Intercept)   24.662   0.712 1199  < 2e-16 ***
```

```
## treatment     -5.577   1.095   26  3.5e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = exchangeable
## Estimated Scale Parameters:
##
##             Estimate Std.err
## (Intercept)     56.3     7.4
##   Link = identity
##
## Estimated Correlation Parameters:
##       Estimate Std.err
## alpha    0.377  0.0683
## Number of clusters:   100  Maximum cluster size: 4
```

```
mod1_unstr <- geeglm(lead_value ~ treatment, id = factor(ID),
    data = lead_data, corstr = "unstructured")
summary(mod1_unstr)  #$coefficients
```

```
##
## Call:
## geeglm(formula = lead_value ~ treatment, data = lead_data, id = factor(ID),
##     corstr = "unstructured")
##
##  Coefficients:
##             Estimate Std.err   Wald Pr(>|W|)
## (Intercept)   25.379   0.698 1321.3  < 2e-16 ***
## treatment     -5.288   1.028   26.4  2.7e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation structure = unstructured
## Estimated Scale Parameters:
##
```

```
##             Estimate Std.err
## (Intercept)       57    7.35
##   Link = identity
##
## Estimated Correlation Parameters:
##           Estimate Std.err
## alpha.1:2  -0.0487  0.1062
## alpha.1:3   0.0695  0.1058
## alpha.1:4   0.4058  0.0880
## alpha.2:3   0.8822  0.0933
## alpha.2:4   0.4821  0.1094
## alpha.3:4   0.5190  0.1202
## Number of clusters:   100  Maximum cluster size: 4
```

3 Conclusion

As a tool for discussion and consideration, here is a table showing the treatment effect estimates and standard errors from each of the models we've fit above:

```
res_tab <- rbind(summary(mod1)$coefficients[2, c(1, 2)], summary(mod1_ind)$coefficients[2,
    c(1, 2)], summary(mod1_exch)$coefficients[2, c(1, 2)], summary(mod1_unstr)$coefficients[2,
    c(1, 2)])

row.names(res_tab) <- c("GLM", "GEE, Independence", "GEE, Exchangeable",
    "GEE, Unstructured")

kable(res_tab)
```

	Estimate	Std.err
GLM	-5.58	0.752
GEE, Independence	-5.58	1.095
GEE, Exchangeable	-5.58	1.095
GEE, Unstructured	-5.29	1.028

References

Kung-Yee Liang Liang and Scott L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22, 1986.