Longitudinal Models

Lecture 12

Nicholas Christian

BIOST 2094 Spring 2011

Outline

1. GEE Models
2. Mixed Models
3. Frailty Models

Generalized Estimating Equations

- Population-average or marginal model, provides a regression approach for generalized linear models when the responses are not independent (correlated/clustered data)
- Goal is to make inferences about the population, accounting for the within-subject correlation
- The packages `gee` and `geepack` are used for GEE models in `R`
- The major difference between `gee` and `geepack` is that `geepack` contains an ANOVA method that allows us to compare models and perform Wald tests.

Generalized Estimating Equations

- Basic Syntax for `geeglm()` from the `geepack` package; has a syntax very similar to `glm()`

```
geeglm(formula, family=gaussian, data, id, zcor=NULL, constr,
                                        std.err="san.se")
```

`formula`	Symbolic description of the model to be fitted
`family`	Description of the error distribution and link function
`data`	Optional dataframe
`id`	Vector that identifies the clusters
`zcor`	Enter a user defined correlation structure
`constr`	Working correlation structure: `"independence"`, `"exchangeable"`, `"ar1"`, `"unstructured"`, `"userdefined"`
`std.err`	Type of standard error to be calculated. Default `"san.se"` is the robust (sandwich) estimate; use `"jack"` for approximate jackknife variance estimate

Correlation Structure

- Independence,
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
- Exchangeable,
$$\begin{pmatrix} 1 & \rho & \rho \\ \rho & 1 & \rho \\ \rho & \rho & 1 \end{pmatrix}$$
- Autoregressive order 1,
$$\begin{pmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{pmatrix}$$
- Unstructured,
$$\begin{pmatrix} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{pmatrix}$$
- GEE model will give valid results with a misspecified correlation structure when the sandwich variance estimator is used

Inference

- For a `geeglm` object returned by `geeglm()`, the functions `drop1()`, `confint()` and `step()` do not apply; however `anova()` does apply.
- The function `esticon()` in the `doBy` package computes and test linear functions of the regression parameters for `lm`, `glm` and `geeglm` objects
- Basic syntax,

```
esticon(obj, cm, beta0, joint.test=FALSE)
```

`obj`	Model object
`cm`	Matrix specifying linear functions of the regression parameters (one linear function per row and one column for each parameter)
`beta0`	Vector of numbers
`joint.test`	If `TRUE` joint Wald test of the hypothesis `Lbeta=beta0` is made, default is one test for each row, `(Lbeta).i=beta0.i`

esticon()

- Let $\hat{\beta} = (\hat{\beta}_1, \ldots, \hat{\beta}_p)$ denote the estimated parameters. Also let $k = (k_1, \ldots, k_p)$ denote a vector of constants; one row of the matrix for the `cm` argument. Then $c = k^T\beta = k_1\beta_1 + \ldots + k_p\beta_p$.
- `esticon()` calculates the linear combinations of the parameter estimates c, the standard error and the confidence interval
- Specify a value for `beta0` to test $H_0 : c =$ `beta0`
- If `joint.test=TRUE` then all of the linear combinations are tested jointly

Example - GEE

```
# Install and load package geepack
install.packages("geepack")
library(geepack)

# ohio dataset from geepack - Health effect of air pollution
# Children followed for four years, wheeze status recorded annually
data(ohio) # Load the dataset
head(ohio)
str(ohio)

# Response is binary - fit a logistic GEE model
# Treat time (age) as continuous
fit.exch <- geeglm(resp~age+smoke, family=binomial(link="logit"),
     data=ohio, id=id, corstr = "exchangeable", std.err="san.se")
fit.unstr <- geeglm(resp~age+smoke, family=binomial(link="logit"),
     data=ohio, id=id, corstr = "unstructured", std.err="san.se")
summary(fit.exch)
summary(fit.unstr)
```

Example - GEE

```
# Treat time (age) as categorical
fit <- geeglm(resp~factor(age)+smoke, family=binomial(link="logit"),
     data=ohio, id=id, corstr = "exchangeable", std.err="san.se")
summary(fit)

# Test the effect of smoke using anova()
fit1 <- geeglm(resp~factor(age)+smoke, family=binomial(link="logit"),
     data=ohio, id=id, corstr = "exchangeable", std.err="san.se")
fit2 <- geeglm(resp~factor(age), family=binomial(link="logit"),
     data=ohio, id=id, corstr = "exchangeable", std.err="san.se")
anova(fit1, fit2)

# Individual Wald test and confidence interval for each parameter
est <- esticon(fit, diag(5))

# Odds ratio and confidence intervals
OR.CI <- exp(cbind(est$Estimate, est$Lower.CI, est$Upper.CI))
rownames(OR.CI) <- names(coef(fit))
colnames(OR.CI) <- c("OR", "Lower OR", "Upper OR")
```

Example - GEE

```
# Odds ratio of wheezing for a 9-year old with a mother who smoked
# during the first year of the study compared to an 8-year old with a
# mother who did not smoke during the first year of the study
# That is estimate, [smoke+factor(age)0] - [factor(age)-1]
esticon(fit, c(0,-1,1,0,1))
exp(.Last.value$Estimate)
# 9-year old with mother who smoked is at greater risk of wheezing

# Jointly test effects using esticon()
fit <- geeglm(resp~factor(age)*smoke, family=binomial(link="logit"),
     data=ohio, id=id, corstr = "exchangeable", std.err="san.se")
summary(fit)

L = cbind(matrix(0, nrow=3, ncol=5), diag(3))
esticon(fit, L, joint.test=TRUE)

# Could also use anova()
fit1 <- geeglm(resp~factor(age)*smoke, family=binomial(link="logit"),
     data=ohio, id=id, corstr = "exchangeable", std.err="san.se")
fit2 <- geeglm(resp~factor(age)+smoke, family=binomial(link="logit"),
     data=ohio, id=id, corstr = "exchangeable", std.err="san.se")
anova(fit1, fit2)
```

Mixed Models

- Subject-specific or cluster-specific model of correlated/clustered data
- Basic premise is that there is natural heterogeneity across individuals in the study population that is the result of unobserved covariates; random effects account for the unobserved covariates.
- The `lme4` package contains functions for fitting linear mixed models, generalized linear mixed models and nonlinear mixed models
- The `lme4` package uses S4 classes and methods.
  - Information in S4 classes is organized into slots. Each slot is named and requires a specified class.
  - Use the `@` to extract information from a slot.
  - To get the names of the slots use, `getSlots("class name")`
- For more information about fitting mixed models in `R` using `lme4` see the available vignettes, `vignette(package="lme4")`

lmer()

- The `lmer()` function in the `lme4` package is used to fit linear and generalized linear models.
- Basic syntax,

```
lmer(formula, data, family=NULL, REML=TRUE)
```

`formula`	Symbolic description of the model to be fitted
`data`	Optional dataframe
`family`	Description of the error distribution and link function, if `NULL` a linear mixed model is fitted
`REML`	Logical, if `TRUE` estimate using REML (provides a consistent estimate of the variance components); if `FALSE` estimate using ML

- `lmer()` returns a `mer` object; see `?"mer-class"` for an explanation of the slots of `mer`

Formula `lmer()`

- A random-effects term in `lmer()` is specified by a linear model term and a grouping factor separated by '|'; i.e. a random effect is a linear model term conditional on the level of the grouping factor.
- The entire random-effects expression should be enclosed in parentheses since the precedence of '|' as an operator is lower than most other operators used in linear model formulas
- For example,
  - Random intercept,

    ```
    lmer(Reaction ~ Days + (1 | Subject), data=sleepstudy)
    ```

  - Random intercept and slope,

    ```
    lmer(Reaction ~ Days + (Days | Subject), data=sleepstudy)
    ```

- See the vignettes for how to fit nested random effects, `vignette(package="lme4")`

Inference

- Functions used for inference and prediction,

`summary()`	Summarize model results
`anova()`	Sequential tests of fixed effects and model comparison
`VarCorr()`	Extract variance components
`ranef()`	Predict random effects
`residuals()`	Extract residuals

- See `?"mer-class"` for a complete list of available methods

Example - Mixed Models

```
# Install and load lme4 package
install.packages("lme4")
libary(lme4)

# Sleep data
# Average reaction time per day for subjects in a sleep deprivation study
data(sleepstudy)
head(sleepstudy)
str(sleepstudy)

# Trellis plot of data
# Clear that there are different intercepts and slopes for each subject
# type = c("g","p","r"),  plots grid lines, points, and regression line
xyplot(Reaction ~ Days | Subject, data=sleepstudy,
       type = c("g","p","r"),
       xlab = "Days of sleep deprivation",
       ylab = "Average reaction time (ms)", aspect = "xy")
```

Example - Mixed Models

```
# Random intercept model
fit1 <- lmer(Reaction ~ Days + (1 | Subject), data=sleepstudy)
summary(fit1)

# Extract information
names(fit1)       # S4 class need to look at the slots
getSlots("mer")   # Slot names for a mer object returned by lmer()

fit1@deviance     # Get deviance

sum.fit1 <- summary(fit1)  # Additional information returned by summary
class(sum.fit1)
getSlots("summary.mer")
sum.fit1@coefs             # Coefficients

VarCorr(fit1)     # Extract variance estimates
```

Example - Mixed Models

```
# Model Diagnostics
y.hat   <- fitted(fit1)           # Fitted values
int.hat <- ranef(fit1)[[1]][[1]] # Predicted intercepts
res.hat <- residuals(fit1)        # Estimated residuals

qqnorm(int.hat, main="Random Intercepts"); qqline(int.hat)
qqnorm(res.hat, main="Residuals"); qqline(res.hat)
plot(y.hat, res.hat, xlab="Fitted Values", ylab="Residuals")
abline(h=0, lty=2)

# Random intercept and slope model
fit2 <- lmer(Reaction ~ Days + (Days | Subject), data=sleepstudy)
summary(fit2)

# Make outcome a binary variable
sleepstudy$react.YesNo <- with(sleepstudy, cut(Reaction,
       breaks=c(min(Reaction), mean(Reaction), max(Reaction)),
       labels=c("Yes", "No")))

# Generalized linear mixed model - binomial outcome
lmer(react.YesNo ~ Days + (Days | Subject),
     data=sleepstudy, family=binomial)
```

Frailty Models

- Frailty models are used to model correlated survival data
- This could be recurrent failures on the same subject or clustered event times
- Similar to a mixed model with a random intercept
- Suppose V is an independent identically distributed random variable then the frailty model given $V = v$ for time T is,

$$h(t|v) = h_0(t)exp(X\beta + v)$$

Frailty Models

- To fit a frailty model in R use `coxph()` along with the function `frailty()` on the right-hand side of the formula
- The argument of `frailty()` is the variable to be added as a random effect; such as an ID variable for a subject-specific model

`frailty()`	Gamma/Normal frailty, specify the distribution
`frailty.gamma()`	Gamma frailty
`frailty.gaussian()`	Normal frailty

- Functions that apply to Cox models also apply to frailty models
- Several other packages exist for analyzing frailty models, see the CRAN Task View: Survival Analysis for more information, `http://cran.r-project.org/web/views/Survival.html`

Example - Frailty Models

```
library(survival)

# A catheter is inserted and ramains in place until an infection occurs.
# Then the catheter is removed and is reinserted after the infection has
# cleared up. Subjects may have several infections, time is recorded
# from insertion to the next infection (assume no carry over effects)
head(kidney)
str(kidney)

# Cox model without accounting for the correlation
coxph(Surv(time, status)~age+sex+disease, data=kidney)

# Cox model with a frailty term accounting for the correlation
# Assuming a normally distributed frailty term
fit <- coxph(Surv(time, status)~age+sex+disease+frailty.gaussian(id),
             data=kidney)

# Functions that apply to Cox models also apply to frailty models
summary(fit)
confint(fit)
anova(fit)
drop1(fit, ~.)
step(fit)
```