Linear and Generalized Linear Models

Lecture 10

Nicholas Christian

BIOST 2094 Spring 2011

Outline

1. Fit linear models
2. Inference
3. Model Diagnostics
4. Model Selection
5. Descriptive Plots
6. Generalized Linear Models

Fit Linear Models

- Functions for fitting linear models

  `lm()` Fits linear models (linear regression or ANOVA)

  `aov()` Fits *balanced* ANOVA model; returns Type I, sequential sum of squares

- Main difference between `lm()` and `aov()` is the way `summary()` handles the results. The summary table for `aov()` is one row for each categorical variable and the summary table for `lm()` has one row for each each estimated parameters (i.e. one row for each factor level)
- Basic syntax for `lm()` (similar syntax for `aov()`),

```
lm(formula, data)
```

  `formula` Symbolic description of the model

  `data` Optional dataframe containing the variables in the model

- `summary.lm()` and `summary.aov()` summarize a linear model and ANOVA model, respectively

Formulas

- Basic form of a formula,

$$\texttt{response} \sim \texttt{model}$$

- Formula notation,

'+'	Separates main effects
':'	Denotes interactions
'*'	All main effects and interactions
'^n'	Include all main effects and n-order interactions
'-'	Removes the specified terms
'\'	Nested effects
`I()`	Brackets the portions of a formula where operators are used mathematically
'.'	Main effect for each column in the dataframe, except the response

Formulas

- Sample formulas, for a model with response `y` and predictors `a`, `b` and `c`

Model	**Interpretation**
`y ~ 1`	Just the intercept
`y ~ a`	One main effect
`y ~ -1+a`	No intercept
`y ~ a+b`	Two main effects
`y ~ a+b+c+a:b`	Three main effects and an interaction between `a` and `b`
`y ~ a*b`	All main effects and interactions (same as `a+b+a:b`)
`y ~ factor(a)`	Create dummy variables for `a` (if not already a factor)
`y ~ (a+b+c)^2`	All main effects and second-order interactions
`y ~ I(a^2)`	Transform `a` to `a^2`
`log(y) ~ a`	Log transform `y`
`y ~ a/b/c`	Factor `c` nested within factor `b` within factor `a`
`y ~ .`	Main effect for each column in the dataframe

Example - Linear Regression

```
# Motor trend car data on 32 cars, response is miles per gallon (MPG)
head(mtcars)

# Basic linear model with one main effect, vehicle weight (wt)
fit <- lm(mpg~wt, data=mtcars)

# Summarize results
sum.fit <- summary(fit)

# Extract Information - model fit
names(fit)
fit$coef    # Coefficients, notice partial matching

# Extract Information - summary of model fit
names(sum.fit)
sum.fit$coef      # Estimates and p-values
sum.fit$coef[,4]  # P-values
sum.fit$r.sq      # R^2

# Linear model with wt, horse power (hp) and an interaction
fit <- lm(mpg~wt*hp, data=mtcars)
```

Example - ANOVA

```
# ToothGrowth dataset
head(ToothGrowth)

# Need to use factor(dose) since dose is a
# numeric variable in ToothGrowth
fit <- aov(len~factor(dose)+supp, data=ToothGrowth)
summary(fit)

# Could also use lm()
# Default reference category is the first factor level
fit <- lm(len~factor(dose)+supp, data=ToothGrowth)
summary(fit)
anova(fit)

# Change reference categories so that
# 2mg is now the reference level for dose
dose.2 <- relevel(factor(ToothGrowth$dose), 3)
fit <- lm(len~dose.2+supp, data=ToothGrowth)
summary(fit)
```

Inference for Linear Models

- Functions used for performing inference

`anova()`	Compute an ANOVA table for model terms or compare nested models; returns Type I, sequential sum of squares
`drop1()`	Test factors using the Type III, marginal sum of squares
`confint()`	Confidence intervals for model parameters
`predict.lm()`	Get the average response value for predictors included and not included in the model; get confidence and prediction intervals for the fitted values
`TukeyHSD()`	Multiple comparisons, Tukey's Honest Significant Difference
`pairwise.t.test()`	Pairwise t-tests, correcting for multiple comparisons

Example - Inference

```
# Basic linear model with one main effect, vehicle weight (wt)
fit <- lm(mpg~wt, data=mtcars)
confint(fit)   # Confidence intervals of parameters

# Plot data with fitted line as well as confidence bands
# using formula interface
plot(mpg~wt, data=mtcars, xlab="Weight (lb/1000)", ylab="MPG")
abline(fit, lwd=2)         # Add fitted line

# Use predict() to evaluate the model at each value of new,
# this way we get a smooth line across the graph
new <- data.frame(wt=seq(0, 6, len=20))
conf.band <- predict(fit, new, interval="confidence")
lines(new$wt, conf.band[,2], col="blue", lwd=2)  # Add lower CI band
lines(new$wt, conf.band[,3], col="blue", lwd=2)  # Add upper CI band
```

Example - Inference

```
# Treat number of cylinders (cyl) as a factor variable
fit <- lm(mpg~wt+factor(cyl), data=mtcars)
summary(fit)

# Test the effect of the number of cylinders
anova(fit)     # Uses Type I sum of squares
               # Not appropriate for testing cyl order of terms matters
drop1(fit, ~., test="F")  # Tests each term using Type III sum of squares
                          # Conditional on other terms being in the model
drop1(fit, ~factor(cyl), test="F")  # Test just factor(cyl)

# Alternative approach, compare models with and without cyl using anova()
fit.1 <- lm(mpg~wt, data=mtcars)
fit.2 <- lm(mpg~wt+factor(cyl), data=mtcars)
anova(fit.2, fit.1)

# With balanced designs we can use anova()
fit <- lm(len~supp+factor(dose), data=ToothGrowth)
anova(fit)

# Same as using summary() and aov()
fit <- aov(len~supp+factor(dose), data=ToothGrowth)
summary(fit)
```

Example - Multiple Comparisons

- The package `multcomp` contains several other methods for multiple comparisons

```
fit <- aov(len~supp+factor(dose), data=ToothGrowth)

# Confidence interval and adjusted p-value using Tukey's HSD
# Note TukeyHSD is only appropriate for balanced designs
TukeyHSD(fit)                          # For all terms
TukeyHSD(fit, "factor(dose)")          # Just dose
plot(TukeyHSD(fit, "factor(dose)")) # Plot method for TukeyHSD()

# Adjusted p-values using Bonferroni
with(ToothGrowth, pairwise.t.test(len, factor(dose), "bonferroni"))
```

Model Diagnostics

- Several functions provide information used with model diagnostics

`fitted.values()`	Returns fitted values
`residuals()`	Returns residuals
`rstandard()`	Standardized residuals, variance one; residual standardized using overall error variance
`rstudent()`	Studentized residuals, variance one; residual standardized using leave-one-out measure of the error variance
`qqnorm()`	Normal quantile plot
`qqline()`	Add a line to the normal quantile plot
`plot.lm()`	Given a `lm` object produces six diagnostic plots, selected using the `which` argument; default is plots 1-3 and 5

1. Residual versus fitted values
2. Normal quantile-quantile plot
3. $\sqrt{|\text{Standardized residuals}|}$ versus fitted values
4. Cook's distance versus row labels
5. Standardized residuals versus leverage along with contours of Cook's distance
6. Cook's distance versus leverage/(1-leverage) with $\sqrt{|\text{Standardized residuals}|}$ contours

Model Diagnostics

`dffits()`	Return DFFITS
`dfbeta()`	Return DFBETAS
`covratio()`	Return covariance ratio; vector whose *i*th element is the ratio of the determinants of the estimated covariance matrix with and without data point *i*
`cooks.distance()`	Return Cook's distance
`hatvalues()`	Diagonal of the hat matrix
`influence.measures()`	Returns the previous five measure of influence and flags influential points
`lm.influence()`	Returns four measures of influence:
`hat`	Diagonal of the hat matrix, measure of leverage
`coefficients`	Matrix whose *i*th row contains the *change* in the estimated coefficients when the *i*th case is removed
`sigma`	Vector whose *i*th element contains the estimate of the residual standard error when the *i*th case is removed
`wt.res`	Vector of weighted residuals or raw residuals if weights are not set

Example - Model Diagnostics

```
fit <- lm(mpg ~ wt, data=mtcars)

# Influential points are labeled
plot(fit)             # Returns four diagnostics plots (1-3 and 5)
plot(fit, which=1:6)  # Returns all six diagnostic plots

# Clicking to advanced the slide is a par() setting
par(ask=TRUE)
plot(residuals(fit), fitted.values(fit))
qqnorm(residuals(fit)); qqline()
plot(cooks.distance(fit), rownames(fit), type="h")

# Influence measures
influence.measures(fit)

# Extract influential points, uses $is.inf
inf.temp <- influence.measures(fit)
inf.pts <- which(apply(inf.temp$is.inf, 1, any))
mtcars[inf.pts,]
```

Example - Model Diagnostics

```
# Influence measures
lm.influence(fit)

# Extract points that cause the greatest change in the estimates
lm.inf.coef <- lm.influence(fit)$coefficients
lm.inf.pts <- apply(lm.inf.coef, 2, FUN=function(x) which.max(abs(x)))

lm.inf.coef[lm.inf.pts,]  # This result agrees with the diagnostic plots

# Get the five points that cause the greatest change in the estimates
lm.inf.pts.top5 <- apply(lm.inf.coef, 2, FUN=function(x)
                           names(rev(sort(abs(x)))[1:5]))
lm.inf.pts.top5
```

Model Selection

- Functions for model selection

`step()`	Choose a model by AIC in a stepwise algorithm
`extractAIC()`	Compute the AIC for the fitted model
`anova()`	Given multiple models tests the models against one another in the order specified
`add1()`	Add one term to a model and compute the change in fit
`drop1()`	Drop one term from a model and compute the change in fit

Example - Model Selection

```
# Plot a main effect for each column of mtcars (except mpg)
# Initial model in step-wise selection
fit.all <- lm(mpg~., data=mtcars)
summary(fit.all)

# Step-wise selection for all effects
result.step.1 <- step(fit.all, direction="both")
summary(result.step.1)

# Step-wise selection for all main effects keeping hp in the model
result.step.2 <- step(fit.all, direction="both", scope=list(lower=~hp))
summary(result.step.2)

# Step-wise selection for all main effects and upto all interactions
result.step.3 <- step(fit.all, direction="both", scope=list(upper=~.^2))
summary(result.step.3)

# Step-wise selection for all main effects keeping hp in the model
# regardlessly and including all terms upto two-way interactions
result.step.4 <- step(fit.all, direction="both",
                      scope=list(lower=~hp, upper=~.^2))
summary(result.step.4)
```

Example - Model Selection

```
# Compare a full model and reduced model using anova()
model1 <- lm(mpg~wt, data=mtcars)
model2 <- lm(mpg~wt+hp+factor(cyl), data=mtcars)
anova(model1, model2)

# Does adding the quadratic term wt^2 reduce AIC?
add1(fit.all, ~. + I(wt^2), test="F")

# Does removing hp reduce AIC?
drop1(fit.all, ~hp, test="F")
```

lowess() and pairs()

```
# Motor Trend Data
head(mtcars)

# Scatterplot smoothing using LOWESS (locally weighted least squares)
plot(mtcars$wt, mtcars$mpg, ylab="MPG", xlab="Weight (lb/1000)", pch=19)
lines(lowess(mtcars$wt, mtcars$mpg), lwd=2, col="blue")

# Scatterplot matrix
pairs(mtcars[,c("mpg", "hp", "wt")], pch=19)

# Scatterplot matrix with custom panels
# panel.smooth() is a built-in function that
# adds a lowess curve to each panel
pairs(mtcars[,c("mpg", "hp", "wt")], pch=19, lwd=1,
      panel=panel.smooth)

# There are also arguments for customizing the the upper panel,
# lower panel, and diagonal panel separately
```

interaction.plot()

```
with(ToothGrowth,
     interaction.plot(x.factor=dose, trace.factor=supp, response=len,
     fun=mean,
     xlab="Dose", ylab="Average Length", trace.label="Supplement",
     lty=1, lwd=2, col=c("red", "blue")))

# Interaction plot with custom legend
with(ToothGrowth,
     interaction.plot(dose, supp, len, fun=mean,
     xlab="Dose", ylab="Average Length",
     lty=1, lwd=2, col=c("red", "blue"), legend=FALSE))

legend("bottomright", c("Orange Juice", "Ascorbic Acid"),
       col=c("red", "blue"), lty=1, bty="n", title="Supplement")
```

Generalized Linear Models

- Generalized Linear Models are fit using the function `glm()`. Basic syntax,

```
glm(formula, family = gaussian, data)
```

- The `family` argument specifies the error distribution and link function. See `?family` for more information

```
binomial(link = "logit")
gaussian(link = "identity")
poisson(link = "log")
```

- Almost all of the functions discussed previously that work with `lm` objects have corresponding methods for `glm` objects. Or are generic enough that they apply to both `lm` objects and `glm` objects. For example,

`summary.glm()`	Summarize the model fit
`anova.glm()`	Analysis of deviance table
`confint.glm()`	Confidence interval for model parameters
`predict.glm()`	Obtain predicted values
`influence.measures()`	Measures of influence
`step()`	Step-wise selection using AIC
`drop1()`	Test parameter using deviance

Example - Generalized Linear Models

```
# Create binary outcome, success if length of tooth is greater than 20
head(ToothGrowth)
y <- ifelse(ToothGrowth[,1]>20, 1, 0)

# Fit logistic model
fit <- glm(y~supp+factor(dose), family="binomial", data=ToothGrowth)
summary(fit)
confint(fit)                    # Confidence interval for the parameters
anova(fit, test="Chisq")        # Compare reduction in deviance, sequentially
drop1(fit, ~., test="Chisq") # Compare reduction in deviance, marginally

exp(coef(fit))    # Exponentiate coefficients
exp(confint(fit)) # 95% CI for Exponentiated coefficients

# Diagnostics
influence.measures(fit)

# Model selection
final <- step(fit, scope=~.^2)
summary(final)
```