Linear models

1 Fetch data and create model objects

```
library(MASS)
#R: data(mammals, cabbages)
#S: mammals <- mammals
#S: cabbages <- cabbages
mammals.lm <- lm(log(brain)~log(body), data=mammals)
cabbages.aov <- aov(VitC~Cult+Date, data=cabbages)
cabbages.lm <- lm(VitC~HeadWt, data=cabbages)
cabbages.ancova <- lm(VitC~HeadWt+Cult+Date, data=cabbages)
```

2 Object anatomy

How an lm object prints on the screen

```
mammals.lm
```

```
  Call:
  lm(formula = log(brain) ~ log(body), data = mammals)

  Coefficients:
  (Intercept)    log(body)
       2.1348       0.7517

  #S: Degrees of freedom: 62 total; 60 residual
  #S: Residual standard error: 0.6942947
```

How a summary of an lm object prints on the screen

```
summary(mammals.lm)
#R: summary(mammals.lm, cor=T)
```

```
  Call:
  lm(formula = log(brain) ~ log(body), data = mammals)

  Residuals:
       Min       1Q   Median       3Q      Max
  -1.71550 -0.49228 -0.06162  0.43597  1.94829

  Coefficients:
              Estimate Std. Error t value Pr(>|t|)
  (Intercept)  2.13479    0.09604   22.23   <2e-16 ***
  log(body)    0.75169    0.02846   26.41   <2e-16 ***
  ---
  Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

  Residual standard error: 0.6943 on 60 degrees of freedom
  Multiple R-Squared: 0.9208,     Adjusted R-squared: 0.9195
  F-statistic: 697.4 on 1 and 60 DF,  p-value: < 2.2e-16

  Correlation of Coefficients:
            (Intercept)
  log(body) -0.3964
```

What's inside an lm object

```
names(mammals.lm)

  call           # recipe, what we can type to create this model
  coefficients   # parameter estimates
  fitted.values
  residuals
  rank           # number of parameters estimates, df used
  df.residual    # residual degrees of freedom, df left

mammals.lm$call
mammals.lm$coe
mammals.lm$fit
mammals.lm$res
mammals.lm$rank
mammals.lm$df.res
```

What's inside a summary of an lm object

```
names(summary(mammals.lm))

  coefficients  # parameter estimates and t test of β=0
  r.squared
  correlation   # between parameter estimates

summary(mammals.lm)$coe
x <- summary(mammals.lm)
x$coe
x$r.s
x$cor
```

3 Symbols

Formula notation

We have already learned how to use # for comments and $ to extract object elements
Model formulas are expressed using the following symbols:

```
~  # is a function of                 y ~ x
+  # add term                         y ~ x1 + x2
:  # interaction term                 y ~ x1 + x2 + x1:x2
I  # do not interpret                 y ~ x1 + I(x2+x3)

*  # both terms and their interaction  y ~ x1 * x2         same as  y ~ x1 + x2 + x1:x2
-  # but not this term                y ~ x1 * x2 - x2    same as  y ~ x1 + x1:x2
.  # same as before                   y ~ . + x3          same as  y ~ x1 + x1:x2 + x3

lm(y~1)              # estimate intercept only, null model
lm(y~-1+x)           # estimate slope, fix intercept at 0
lm(offset(y-3)~-1+x) # estimate slope, fix intercept at 3
lm(y~offset(3*x))    # estimate intercept, fix slope at 3
?formula
```

Parentheses, brackets, and braces

```
f(x)   # Pass argument x to function f
x[i]   # Extract element i from vector x
{cmd}  # Lump commands together as a block, used when programming
```

4 Regression plots

Scatterplot and low-level plotting commands

```
plot(log(mammals$body), log(mammals$brain))
abline(mammals.lm)
points(5, 0)
points(5, 0, cex=2)
lines(c(6,4,5), c(0,1,-1))
x.human <- log(mammals$body)[row.names(mammals)=="Human"]
x.human
y.human <- log(mammals$brain)[row.names(mammals)=="Human"]
points(x.human, y.human, pch=3, cex=2)
text(x.human, y.human+0.5, "me")
```

Smoothing with loess

```
#R: library(modreg)
plot(log(mammals$body), log(mammals$brain))
mammals.loess <- loess(log(brain)~log(body), data=mammals)
mammals.loess
summary(mammals.loess)
names(mammals.loess)

  call    # recipe, what we can type to create this model
  fitted

mammals.loess$fit
cbind(log(mammals), mammals.loess$fit)
points(log(mammals$body), mammals.loess$fit, col=6)
lines(log(mammals$body), mammals.loess$fit, col=6)
x <- log(mammals$body)
y <- mammals.loess$fit
plot(log(mammals$body), log(mammals$brain))
lines(x[order(x)], y[order(x)])
```

Box plot

```
boxplot(cabbages$VitC)
boxplot(split(cabbages$VitC, cabbages$Date))
```

Conditioning plot

```
#R: library(lattice)
coplot(VitC~HeadWt|Cult, data=cabbages)
coplot(VitC~HeadWt|Cult, data=cabbages, panel=panel.smooth)
coplot(VitC~HeadWt|Date, rows=1, data=cabbages, panel=panel.smooth)
coplot(VitC~HeadWt|Date*Cult, data=cabbages, panel=panel.smooth)
#S: coplot(VitC~HeadWt|Date*Cult, data=cabbages, panel=panel.smooth, span=0.9)
```

Interaction plot

```
interaction.plot(cabbages$Cult, cabbages$Date, cabbages$VitC)
```

Plot influence diagnostics

```
par(mfrow=c(2,3))
#R: par(mfrow=c(2,2))
plot(mammals.lm)
par(mfrow=c(1,1))
plot(mammals.lm$fit, mammals.lm$res)
abline(h=0)
identify(mammals.lm$fit, mammals.lm$res, row.names(mammals))
```

5 Auxiliary functions

Formal extraction of model elements

```
coef(mammals.lm)       # same as mammals.lm$coef
fitted(mammals.lm)     # same as mammals.lm$fitted
residuals(mammals.lm)  # select one of five different kinds of residuals
args(residuals.lm)
deviance(mammals.lm)   # GLM context, for lm this is SSE=sum(mammals.lm$res^2)
```

Model building and selection

```
update(mammals.lm, .~.+I(body^2))
cabbages.0 <- lm(VitC~1, data=cabbages)
cabbages.full <- update(cabbages.0, .~.+HeadWt*Cult*Date)
add1(cabbages.0, cabbages.full)
drop1(cabbages.full)
cabbages.step <- step(cabbages.0, list(lower=cabbages.0, upper=cabbages.full))
cabbages.step
anova(cabbages.full)
cabbages.plain <- update(cabbages.0, .~.+HeadWt+Cult+Date)
AIC(cabbages.0, cabbages.plain, cabbages.full)
```

Predict response value from new observed data

```
new.cabbage <- data.frame(Cult="c39", Date="d16", HeadWt=4.0)
predict(cabbages.plain, new.cabbage)
predict(cabbages.full, new.cabbage)
exp(predict(mammals.lm, data.frame(body=100)))
```

Influence diagnostics

```
mammals.diag <- ls.diag(mammals.lm)
#S: mammals.diag <- ls.diag(lsfit(log(mammals$body), log(mammals$brain)))
mammals.diag
plot(mammals.diag$cooks, type="h")
abline(h=0)
# See section 4: Plot influence diagnostics
```

Transform response variable

```
mammals.plain <- update(mammals.lm, brain~body)
library(MASS)
plot(boxcox(mammals.plain))    # evaluate 1/Y, log(Y), Y, Y^2, Y^3, ...
plot(logtrans(mammals.plain))  # evaluate log(Y+0.1), log(Y+1), log(Y+10), ...
```

6 Models related to lm and aov

The syntax for glm(), gam(), nls() is nearly identical to lm() and aov(). Read the help for details, preferably with a textbook at hand. In R, run library(mgcv) for GAM and library(nls) for nonlinear least squares models.

7 Caveats

Attaching data frames

If you read about S-Plus and linear models in textbooks or help files, you will often see the attach() function used on data frames. This allows the user to use shorthand notation of data columns instead of explicit dollar references, which saves some typing.

I recommend never using attach() on data frames. I used to, but found out the hard way just how dangerous it is. When you make some changes to the data, the temporary attached object changes but not the real object. This doesn't only sound confusing, it is confusing. If you never attach data frames, it will always be clear what data you are referring to; mammals$body will always be the body column in the mammals data frame.

Extracting residuals

The quickest way to extract residuals from models is m$res, and it works fine for lm() and aov(). As soon as you start using other kinds of models, this can give unexpected results. For GLMs, for example, this returns the working residuals. Use formal extraction instead, specifying what type of residuals you're after: residuals(x, type="response"), residuals(x, type="deviance"), etc.

8 New functions

Data manipulation	`I`
	`c`
	`cbind`
	`order`
Import/export	`attach`
Mathematics	`exp`
Graphics	`points`
	`lines`
	`text`
	`coplot`
	`par`
	`identify`
Modelling	`loess`
	`coef`
	`fitted`
	`residuals`
	`deviance`
	`update`
	`add1`
	`drop1`
	`step`
	`anova`
	`AIC`
	`predict`
	`ls.diag`
	`boxcox`
	`logtrans`