Lecture 1: Model Selection

R^2, sums of squares, and mean squares

We will simulate some data from a linear model with two important covariates and one unimportant covariate. We want to look at the effects of including or excluding certain covariates on R^2 and sums of squares.

```
set.seed(1)

# sample size
n <- 200

# create three covariates
x1 <- rnorm(n)
x2 <- runif(n)
x3 <- rbinom(n,1,.5)

# true parameter vector
beta0 <- c(-1,1,0,1)

# residual variance
sigma2 <- 2^2

# get outcome
y <- cbind(1,x1,x2,x3) %*% beta0 + rnorm(n,sd=sqrt(sigma2))

# fit a model with just x1 (model 1)
mod1 <- lm(y ~ x1)

# model sum of squares is
ss.mod1 <- sum(head(anova(mod1)[,2],-1))
#total sum of squares is
ss.tot1 <- sum(anova(mod1)[,2])
# so we can calculate R^2 as
ss.mod1/ss.tot1
```

```
## [1] 0.1888769
```

```
# and compare with
summary(mod1)$r.squared
```

```
## [1] 0.1888769
```

```
# now add the (irrelevant) covariate x2 (model 2)
mod2 <- lm(y ~ x1 + x2)
```

```r
# model sum of squares is
ss.mod2 <- sum(head(anova(mod2)[,2],-1))
#total sum of squares is
ss.tot2 <- sum(anova(mod2)[,2])
# compare with model sum of squares without including x2
ss.mod1
```

```
## [1] 230.8398
```

```r
ss.mod2
```

```
## [1] 230.858
```

```r
# now add the covariate x3 to the model containing only x1 (model 3)
mod3 <- lm(y ~ x1 + x3)

# model sum of squares is
ss.mod3 <- sum(head(anova(mod3)[,2],-1))
#total sum of squares is
ss.tot3 <- sum(anova(mod3)[,2])
# compare with model sum of squares without including x2
ss.mod1
```

```
## [1] 230.8398
```

```r
ss.mod3
```

```
## [1] 320.8653
```

```r
# look at R^2 for each model
summary(mod1)$r.squared
```

```
## [1] 0.1888769
```

```r
summary(mod2)$r.squared
```

```
## [1] 0.1888918
```

```r
summary(mod3)$r.squared
```

```
## [1] 0.2625372
```

```r
# compare MSE for each model
tail(anova(mod1)[,3],1)
```

```
## [1] 5.006722
```

```r
tail(anova(mod2)[,3],1)
```

```
## [1] 5.032044
```

```r
tail(anova(mod3)[,3],1)
```

```
## [1] 4.575154
```

Partial F tests

We can compare nested models with the partial F test.

```
# model 1 is nested within models 2 and 3, but model 2 is not nested within
model 3 (for example)

# adding an unimportant covariate should not result in a significant decrease
in SSE
anova(mod1,mod2)

## Analysis of Variance Table
##
## Model 1: y ~ x1
## Model 2: y ~ x1 + x2
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    198 991.33
## 2    197 991.31  1  0.018161 0.0036 0.9522

# but adding an important covariate should
anova(mod1,mod3)

## Analysis of Variance Table
##
## Model 1: y ~ x1
## Model 2: y ~ x1 + x3
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    198 991.33
## 2    197 901.31  1    90.026 19.677 1.523e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

AIC and BIC

If we want to compare non-nested models, we can look at information criteria. This doesn't allow for p-values like the F tests, but is available in much more general scenarios.

```
# using the built-in function
AIC(mod1)

## [1] 893.7216

# can calculate this manually as well
-2*logLik(mod1)+2*(length(coef(mod1))+1)

## 'log Lik.' 893.7216 (df=3)

# have to remember to add 1 parameter for the residual variance

# for the other two models
AIC(mod2)
```

```
## [1] 895.718
```

```
AIC(mod3)
```

```
## [1] 876.6808
```

BIC is very similar, but has a larger penalty for overfitting.

```
BIC(mod1)
```

```
## [1] 903.6166
```

```
BIC(mod2)
```

```
## [1] 908.9112
```

```
BIC(mod3)
```

```
## [1] 889.874
```

The best-fitting model will have lowest AIC or BIC.

General strategies for model selection

Let's consider now the same basic setup as before but add more "noise" variables.

```
set.seed(2)
x4 <- runif(n)
x5 <- rbinom(n,1,.25)
x6 <- rbinom(n,1,.9)
x7 <- rexp(n,1)
x8 <- rnorm(n,sd=2)
```

There is an R package that does best subsets. It gets the best model at each model size, so which is considered the best overall depends on AIC, BIC, etc., but which within each size does not.

```
library(leaps)
regsubs <- regsubsets(y ~ x1+x2+x3+x4+x5+x6+x7+x8,
                      data=data.frame(y,x1,x2,x3,x4,x5,x6,x7,x8),
                      nvmax=3)

data.frame(summary(regsubs)$outmat)
```

```
##          x1 x2 x3 x4 x5 x6 x7 x8
## 1  ( 1 )  *
## 2  ( 1 )  *     *
## 3  ( 1 )  *     *        *
```

```
summary(regsubs)$bic
```

```
## [1] -31.27045 -45.01298 -41.55611
```

This correctly identifies the model with x1 and x3 as best-fitting model since the 2-covariate model has the lowest BIC and the best 2-covariate model includes these two covariates.

Forward and backward selection

There is another package that allows for forward, backward, and stepwise model selection based on AIC.

```
library(MASS)

back.fit <- stepAIC(lm(y ~ x1+x2+x3+x4+x5+x6+x7+x8), # starting model
                    direction='backward', # specify direction of selection
                    trace=0) # suppress output during computations
forw.fit <- stepAIC(lm(y ~ x1), # starting model
                    scope=y ~ x1+x2+x3+x4+x5+x6+x7+x8, # full model for
consideration
                    direction='forward',
                    trace=0)

back.fit

##
## Call:
## lm(formula = y ~ x1 + x3)
##
## Coefficients:
## (Intercept)           x1           x3
##      -1.266        1.156        1.342

forw.fit

##
## Call:
## lm(formula = y ~ x1 + x3)
##
## Coefficients:
## (Intercept)           x1           x3
##      -1.266        1.156        1.342
```

Both methods correctly identify the model with x1 and x3.

We can also force inclusion of some covariate of interest.

```
force.x5 <- stepAIC(lm(y ~ x1+x2+x3+x4+x5+x6+x7+x8), # starting model
                    scope=list(upper=~ x1+x2+x3+x4+x5+x6+x7+x8,
                               lower=~ x5), # say we want to force x5 into
the model
                    direction='backward',
                    trace=1)
```

```
## Start:  AIC=314.7
## y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8
##
##        Df Sum of Sq     RSS    AIC
## - x7    1     0.281  881.96 312.77
## - x8    1     1.316  883.00 313.00
## - x2    1     3.390  885.07 313.47
## - x4    1     5.680  887.36 313.99
## <none>               881.68 314.70
## - x6    1     8.982  890.66 314.73
## - x3    1    86.369  968.05 331.39
## - x1    1   219.686 1101.37 357.20
##
## Step:  AIC=312.77
## y ~ x1 + x2 + x3 + x4 + x5 + x6 + x8
##
##        Df Sum of Sq     RSS    AIC
## - x8    1     1.319  883.28 311.07
## - x2    1     3.282  885.25 311.51
## - x4    1     5.565  887.53 312.02
## <none>               881.96 312.77
## - x6    1     9.080  891.04 312.82
## - x3    1    86.113  968.08 329.40
## - x1    1   219.426 1101.39 355.20
##
## Step:  AIC=311.07
## y ~ x1 + x2 + x3 + x4 + x5 + x6
##
##        Df Sum of Sq     RSS    AIC
## - x2    1     3.826  887.11 309.93
## - x4    1     5.969  889.25 310.41
## - x6    1     8.780  892.06 311.04
## <none>               883.28 311.07
## - x3    1    85.135  968.42 327.47
## - x1    1   233.140 1116.42 355.91
##
## Step:  AIC=309.93
## y ~ x1 + x3 + x4 + x5 + x6
##
##        Df Sum of Sq     RSS    AIC
## - x4    1     4.840  891.95 309.02
## - x6    1     8.764  895.87 309.90
## <none>               887.11 309.93
## - x3    1    81.781  968.89 325.57
## - x1    1   231.787 1118.89 354.36
##
## Step:  AIC=309.02
## y ~ x1 + x3 + x5 + x6
##
##        Df Sum of Sq     RSS    AIC
```

```
## - x6    1     8.266  900.21 308.86
## <none>               891.95 309.02
## - x3    1    87.824  979.77 325.80
## - x1    1   237.354 1129.30 354.21
##
## Step:  AIC=308.86
## y ~ x1 + x3 + x5
##
##        Df Sum of Sq     RSS    AIC
## <none>               900.21 308.86
## - x3    1    90.336  990.55 325.99
## - x1    1   229.136 1129.35 352.22
```

```
# in tracing, we see that x5 is never considered for elimination
force.x5
```

```
##
## Call:
## lm(formula = y ~ x1 + x3 + x5)
##
## Coefficients:
## (Intercept)           x1           x3           x5
##     -1.2236       1.1551       1.3449      -0.1695
```

Real data example

Weight (`wgt`), height (`hgt`), and age (`age`) of a random sample of 12 nutritionally deficient children (KKM, p. 385).

```
nutrition.data <- data.frame(wgt=c(64,71,53,67,55,58,77,57,56,51,76,68),
                             hgt=c(57,59,49,62,51,50,55,48,42,42,61,57),
                             age=c(8,10,6,11,8,7,10,9,10,6,12,9))

# examine the data set
nutrition.data
```

```
##    wgt hgt age
## 1   64  57   8
## 2   71  59  10
## 3   53  49   6
## 4   67  62  11
## 5   55  51   8
## 6   58  50   7
## 7   77  55  10
## 8   57  48   9
## 9   56  42  10
## 10  51  42   6
## 11  76  61  12
## 12  68  57   9
```

```
# fit the full model
nutr.fullfit <- lm(wgt ~ age + hgt + age:hgt + I(age^2) + I(hgt^2),
data=nutrition.data,
                   x=TRUE,y=TRUE) # these options return the X matrix and the
outcome as elements of the fitted model object

nutr.subsets <- regsubsets(wgt ~ age + hgt + age:hgt + I(age^2) + I(hgt^2),
data=nutrition.data)

# calculate fit statistics for best model at each model size
nutr.bic <- summary(nutr.subsets)$bic
nutr.adjr2 <- summary(nutr.subsets)$adjr2
nutr.cp <- summary(nutr.subsets)$cp

data.frame(summary(nutr.subsets)$outmat,nutr.bic,nutr.adjr2,nutr.cp)

##          age hgt I.age.2. I.hgt.2. age.hgt   nutr.bic nutr.adjr2
## 1  ( 1 )                                 * -11.836834  0.7288936
## 2  ( 1 )   *   *                       -10.714052  0.7310941
## 3  ( 1 )       *                *       *  -9.541548  0.7288214
## 4  ( 1 )   *   *                *       *  -7.146736  0.6923997
## 5  ( 1 )   *   *        *       *       *  -4.662126  0.6411418
##             nutr.cp
## 1  ( 1 ) -0.4453040
## 2  ( 1 )  0.7440388
## 3  ( 1 )  2.0453664
## 4  ( 1 )  4.0001480
## 5  ( 1 )  6.0000000
```

Looking at the output from the best subsets function, BIC would select the most parsimonious model, but that includes an interaction without the main effect. Sticking with models obeying the proper hierarchy, we would go with a model containing just age and height as the best fit: it has the highest adjusted R^2 and a low C_p.

Multicollinearity

When there is high correlation between predictors, multicollinearity may be an issue.

```
# height as just a linear term
mod.hgt1 <- lm(wgt ~ hgt, data=nutrition.data)
# height and height squared
mod.hgt2 <- lm(wgt ~ hgt + I(hgt^2), data=nutrition.data)
# orthogonal polynomials of degree 2
mod.hgt3 <- lm(wgt ~ poly(hgt,2,raw=FALSE), data=nutrition.data,x=TRUE)

orthcovs <- as.data.frame(mod.hgt3$x[,-1])
colnames(orthcovs) <- c('p1','p2')
mod.hgt4 <- lm(nutrition.data$wgt ~ orthcovs$p1+orthcovs$p2)

summary(mod.hgt2)$coef # high variance for intercept estimate
```

```
##                Estimate  Std. Error    t value  Pr(>|t|)
## (Intercept) 48.02672376 113.1821672  0.4243312 0.6812872
## hgt         -0.56763215   4.4122139 -0.1286502 0.9004635
## I(hgt^2)     0.01580953   0.0424672  0.3722762 0.7183022
```

```
summary(mod.hgt3)$coef # lower variance for intercept estimate and linear
term
```

```
##                             Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)                62.750000   1.652126 37.9813719 3.017760e-11
## poly(hgt, 2, raw = FALSE)1 24.267726   5.723131  4.2402882 2.173367e-03
## poly(hgt, 2, raw = FALSE)2  2.130586   5.723131  0.3722762 7.183022e-01
```

```
summary(mod.hgt4)$coef # same as previous
```

```
##              Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) 62.750000   1.652126 37.9813719 3.017760e-11
## orthcovs$p1 24.267726   5.723131  4.2402882 2.173367e-03
## orthcovs$p2  2.130586   5.723131  0.3722762 7.183022e-01
```

```
vif(mod.hgt2) # high VIF from correlation between height and height squared
```

```
##      hgt I(hgt^2)
## 304.4582 304.4582
```

```
vif(mod.hgt4) # orthogonal predictors have no correlation between themselves
```

```
## orthcovs$p1 orthcovs$p2
##           1           1
```

The `poly()` function offers a way to orthogonalize possibly collinear covariates, but does make the model fit harder to interpret.

We can compare this with another method of dealing with multicollinearity in the case of polynomial models: centering the covariate. This is good practice anyway, as it makes the intercept more interpretable.

```
nutrition.data$hgt.ctr <- nutrition.data$hgt-mean(nutrition.data$hgt)
mod.hgt5 <- lm(wgt ~ hgt.ctr + I(hgt.ctr^2),data=nutrition.data)
summary(mod.hgt5)$coef
```

```
##                Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  62.07513072  2.4527191 25.3086995 1.128886e-09
## hgt.ctr       1.10027317  0.2638488  4.1700897 2.411297e-03
## I(hgt.ctr^2)  0.01580953  0.0424672  0.3722762 7.183022e-01
```

```
vif(mod.hgt5)
```

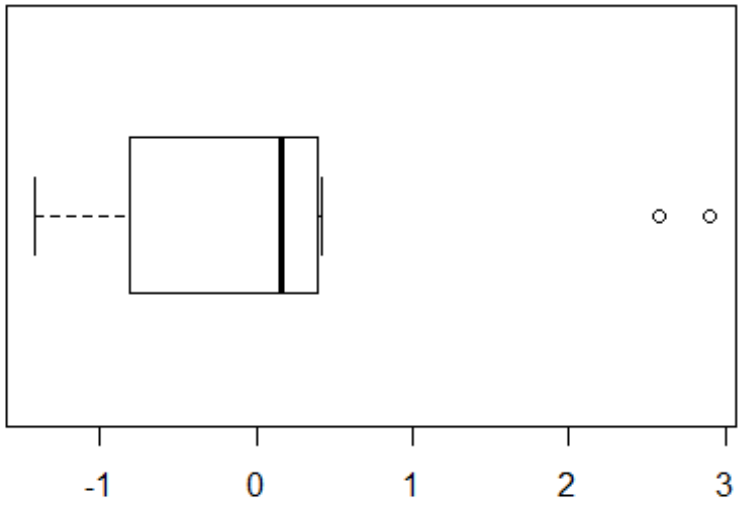
```
##      hgt.ctr I(hgt.ctr^2)
##     1.088742     1.088742
```

This gives VIF very close to 1, and should allow easier interpretation of the slope parameters as well.

Outliers

We can look at diagnostic plots to check for outlying observations. We will use the example of the weight data set to see how this is done. Recall that we fit a model containing height, age, their quadratics and their interaction.

```
# get jackknife residuals (also called studentized residuals)
res.nutr <- rstudent(nutr.fullfit)

boxplot(res.nutr,horizontal=TRUE)
```



Since jackknife residuals beyond either +3 or -3 may be of concern, we might want to look more closely.

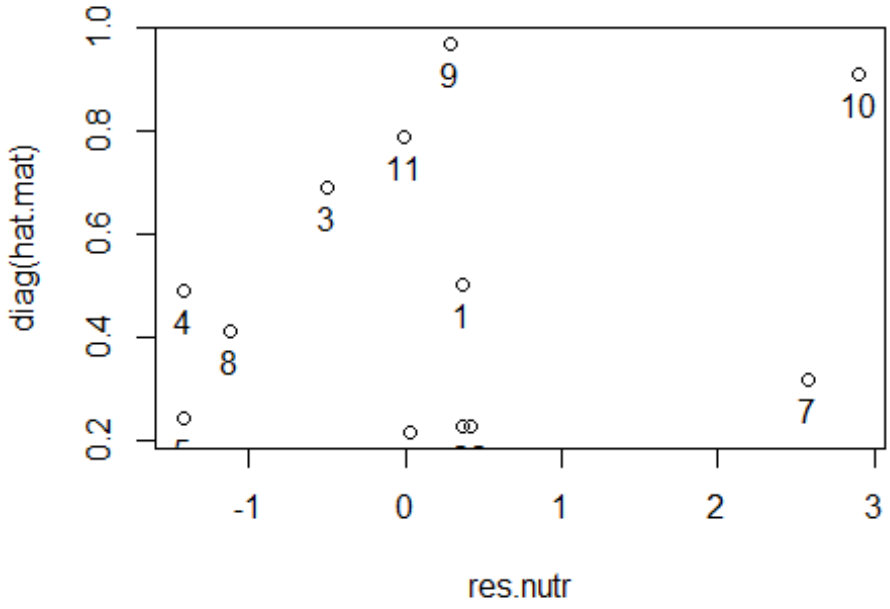
```
res.nutr
```

```
##             1             2             3             4             5
##  0.3636225167  0.3637890412 -0.4921522854 -1.4206290540 -1.4173961242
##             6             7             8             9            10
##  0.0317775464  2.5765068868 -1.1188807403  0.2940191764  2.8987320556
##            11            12
## -0.0005000641  0.4218443408
```

Observation 10 has a jackknife residual of almost 3. Let's look at its leverage.

```
XtX <- t(nutr.fullfit$x) %*% nutr.fullfit$x # the X'X matrix
```

```
hat.mat <- nutr.fullfit$x %*% solve(XtX) %*% t(nutr.fullfit$x) # the hat
matrix

# plot leverage versus jackknife residuals
plot(res.nutr,diag(hat.mat))
text(res.nutr,diag(hat.mat),pos=1,labels=1:nrow(nutrition.data)) # labels
each data point with observation number
```



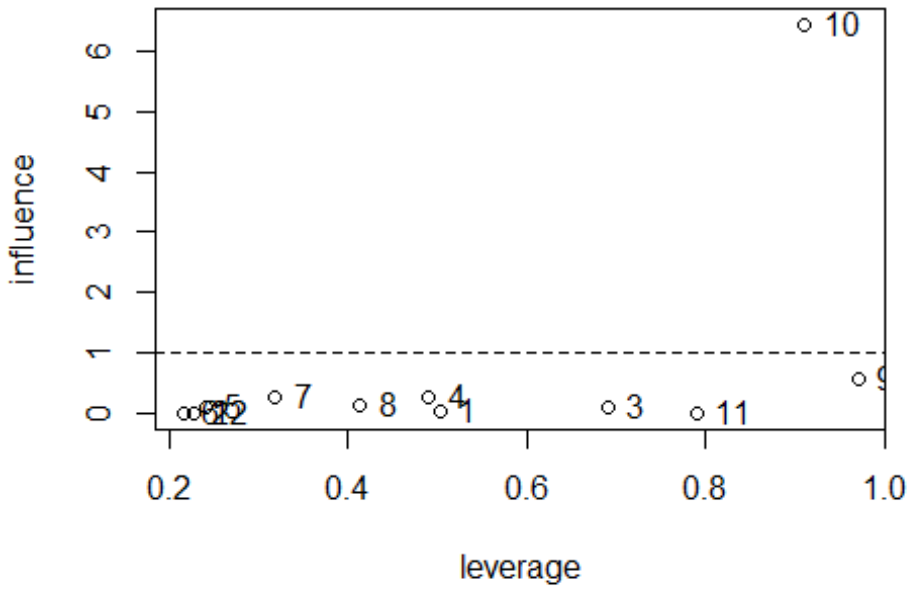
It has high leverage, but so does another data point (observation 9). Recall that high leverage doesn't necessarily mean high influence, just that there is the potential for high influence.

We can look at influence using Cook's distance.

```
# resid() returns the raw residuals:
resid(nutr.fullfit)

##           1           2           3           4           5
##  1.492552161  1.861029040 -1.576390886 -5.045813599 -6.149497418
##           6           7           8           9          10
##  0.165913066  8.220408483 -4.520002095  0.293663581  3.111236740
##          11          12
## -0.001351823  2.148252749

# observed - fitted
nutr.fullfit$y-nutr.fullfit$fitted.values
```

```
##            1            2            3            4            5
##  1.492552161  1.861029040 -1.576390886 -5.045813599 -6.149497418
##            6            7            8            9           10
##  0.165913066  8.220408483 -4.520002095  0.293663581  3.111236740
##           11           12
## -0.001351823  2.148252749
```

```
# we can now calculate the Cook's d manually
cooks.dist <- resid(nutr.fullfit)^2* # e_i^2
  diag(hat.mat)/ # h_i
  (length(coef(nutr.fullfit))* # p+1
     summary(nutr.fullfit)$sigma^2* # MSE
     (1-diag(hat.mat))^2) #(1-h_i)

# plot the measure of influence versus the measure of leverage
plot(diag(hat.mat),cooks.dist,xlab='leverage',ylab='influence')
text(diag(hat.mat),cooks.dist,pos=4,labels=1:nrow(nutrition.data))
abline(h=1,lty=2) # line for (maybe) where we should be concerned
```

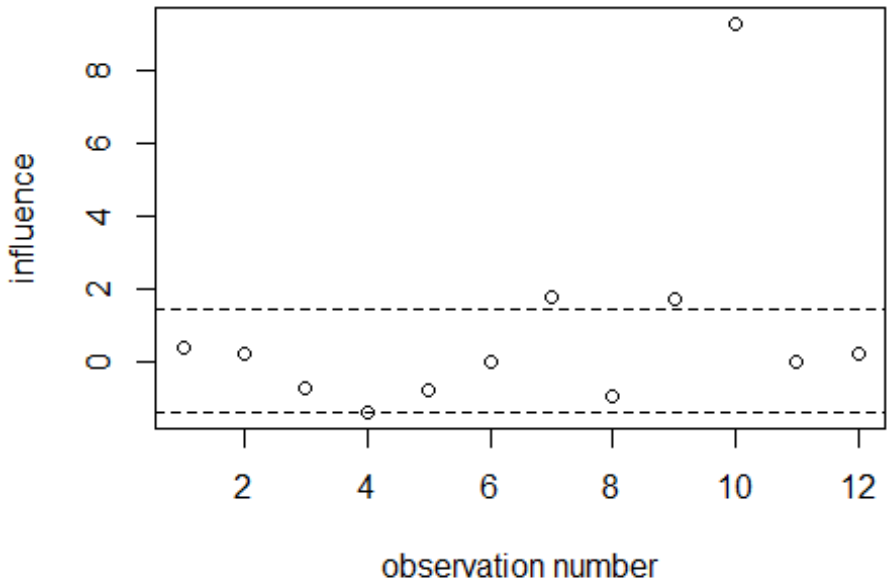


We see from this plot that although observations 9 and 10 have similar leverage, only observation 10 has high influence.

Another way to look at influence is with DFFITS. This is easily calculated from the hat matrix diagonals and jackknife residuals.

```
# calculate this manually using the formula with only jackknife residuals and
hat matrix diagonals
```

```
dffits.nutr <- res.nutr*sqrt(diag(hat.mat)/(1-diag(hat.mat)))

plot(dffits.nutr,xlab='observation number',ylab='influence')
# the cutoff for this to be of concern is +/- 2*sqrt((p+1)/n)
abline(h=c(-
1,1)*2*sqrt(length(coef(nutr.fullfit))/nrow(nutrition.data)),lty=2)
```



This shows again that observation 10 is influential; observations 7 and 9 fall slightly above the cutoff, but not so far as to cause too much concern. Observation 4 is borderline on the negative side.

Influence measures

R has a built-in function to calculate all these measures and a few more.

```
influence.measures(nutr.fullfit)

## Influence measures of
##   lm(formula = wgt ~ age + hgt + age:hgt + I(age^2) + I(hgt^2),      data
= nutrition.data, x = TRUE, y = TRUE) :
##
##       dfb.1_  dfb.age   dfb.hgt dfb.I.g.2. dfb.I.h.2.  dfb.ag.h     dffit
## 1   0.049039  0.19953 -0.110430   0.047375   0.195650 -2.10e-01  0.365568
## 2   0.048550  0.08500 -0.069551  -0.092471   0.052100  3.84e-02  0.197412
## 3   0.194482  0.35198 -0.267524  -0.542551   0.090654  3.64e-01 -0.735394
## 4  -0.871747 -0.21646  0.868759   0.319566  -0.749149 -1.65e-01 -1.394657
## 5   0.368171 -0.04940 -0.341519   0.328654   0.446212 -3.32e-01 -0.799091
```

```
## 6  -0.006778 -0.00381  0.007425   0.002975  -0.006274 -9.42e-04  0.016676
## 7  -1.170975 -0.28190  1.154225  -0.299605  -1.313623  6.30e-01  1.761797
## 8   0.382907 -0.04071 -0.370516   0.421532   0.552919 -4.78e-01 -0.938861
## 9   0.257847  0.63282 -0.399843   0.436759   0.655610 -8.58e-01  1.695785
## 10  6.154351 -2.12107 -4.554411  -3.093059   2.213441  4.45e+00  9.285439
## 11  0.000146  0.00065 -0.000296  -0.000483   0.000257  2.88e-05 -0.000969
## 12  0.006503  0.13501 -0.045701  -0.093999   0.046416  1.70e-03  0.229030
##      cov.r   cook.d   hat inf
## 1   5.1337 2.60e-02 0.503   *
## 2   3.3045 7.59e-03 0.227
## 3   7.2676 1.03e-01 0.691   *
## 4   0.7667 2.77e-01 0.491
## 5   0.5186 9.11e-02 0.241
## 6   3.8036 5.56e-05 0.216
## 7   0.0276 2.67e-01 0.319   *
## 8   1.3315 1.41e-01 0.413
## 9  92.3152 5.65e-01 0.971   *
## 10  0.0906 6.43e+00 0.911   *
## 11 14.2084 1.88e-07 0.790   *
## 12  3.1344 1.01e-02 0.228
```

The first p+1 columns of this matrix are the DFBETAS, followed by DFFITS, covariance ratios (which we haven't talked about), Cook's distance, and hat matrix diagonals.