Lab 6 - Regression Diagnostics

Lab Goals

1. Establish some matrix identities for regression diagnostics and check some of these in R.
2. Use R function to perform these diagnostics in practical data sets.
3. Review DFBETA's in `R` and demonstrate that their fomula gives what it says it does.
4. Review effect/reference coding and their relationships with contrasts.
5. Test contrasts in linear models.

Some matrix identities

1. Show that
$$(X_{(i)}^T X_{(i)})^{-1} = (X^T X)^{-1} + \frac{(X^T X)^{-1} x_i x_i^T (X^T X)^{-1}}{1 - h_{ii}}$$
2. Check Cook's distance in `R`:

Read in cherry tree data from text file with no header variables are D=diameter, H=height, and V=volume 31 cherry trees

```
data=read.table("cherry.txt",sep="",header=FALSE)
colnames(data)=c("D","H","V")
attach(data)
summary(data)
```

```
##        D               H            V
##  Min.   : 8.30   Min.   :63   Min.   :10.20
##  1st Qu.:11.05   1st Qu.:72   1st Qu.:19.40
##  Median :12.90   Median :76   Median :24.20
##  Mean   :13.25   Mean   :76   Mean   :30.17
##  3rd Qu.:15.25   3rd Qu.:80   3rd Qu.:37.30
##  Max.   :20.60   Max.   :87   Max.   :77.00
```

```
n=length(D)
```

fit MLR model to logged variables

```
LD=log(D); LH=log(H); LV=log(V)
```

We'll create an X matrix to predict volume from height and diameter

```
X = cbind(rep(1,31),LD,LH)
```

and the corresponding hat matrix

```
H = X%*%solve(t(X)%*%X)%*%t(X)
```

The "classic" calculation of Cook's distance for observation 1 is to first obtain the standardized residuals

```
yhat = H%*%LV
resid = LV - yhat
sigest2 = t(resid)%*%resid/28
rstd = resid/sqrt(sigest2*(1-diag(H)))
```

```
## Warning in sigest2 * (1 - diag(H)): Recycling array of length 1 in array-vector arithmetic is deprec
##   Use c() or as.vector() instead.
```

```
CooksD = rstd^2/3 * diag(H)/(1-diag(H))
```

Note that we have used element-wise multiplication to calculate all Cook's distances at once.

The other way we could do this would be to just look at the first observation and remove it from the data set (note that `X[-1,]` is `x` without its first row)

```
yhatm1 = X%*%solve(t(X[-1,])%*%X[-1,])%*%t(X[-1,])%*%LV[-1]
CooksD1 = t(yhat-yhatm1)%*%(yhat-yhatm1)/(3*sigest2)
```

And we can check that these are the same

```
c(CooksD[1],CooksD1)
```

```
## [1] 0.005061491 0.005061491
```

Regression Diagnostics in R

Here we will provide diagnostics for the cherry tree data manually. First fit a linear model

```
logfit1=lm(LV~LD+LH)
summary(logfit1)
```

```
##
## Call:
## lm(formula = LV ~ LD + LH)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.168561 -0.048488  0.002431  0.063637  0.129223
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.63162    0.79979  -8.292 5.06e-09 ***
## LD           1.98265    0.07501  26.432  < 2e-16 ***
## LH           1.11712    0.20444   5.464 7.81e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08139 on 28 degrees of freedom
## Multiple R-squared:  0.9777, Adjusted R-squared:  0.9761
## F-statistic: 613.2 on 2 and 28 DF,  p-value: < 2.2e-16
```

```
betahat = logfit1$coefficients
```

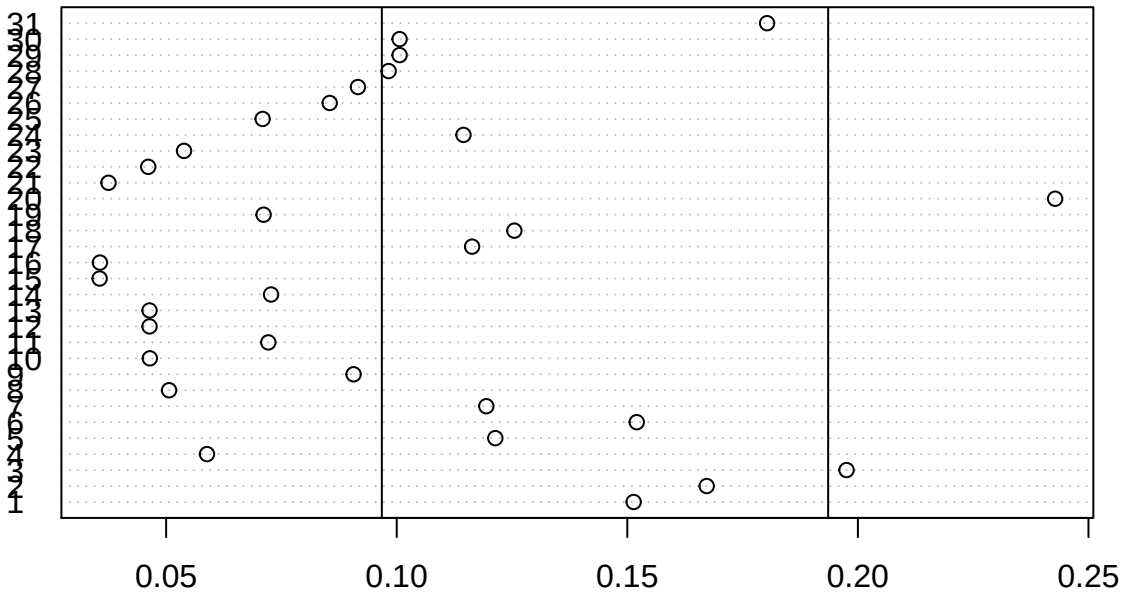
And look at the sequential `anova` table

```
anova(logfit1)
```

```
## Analysis of Variance Table
##
## Response: LV
##           Df Sum Sq Mean Sq F value    Pr(>F)
## LD         1 7.9254  7.9254 1196.53 < 2.2e-16 ***
## LH         1 0.1978  0.1978   29.86 7.805e-06 ***
## Residuals 28 0.1855  0.0066
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

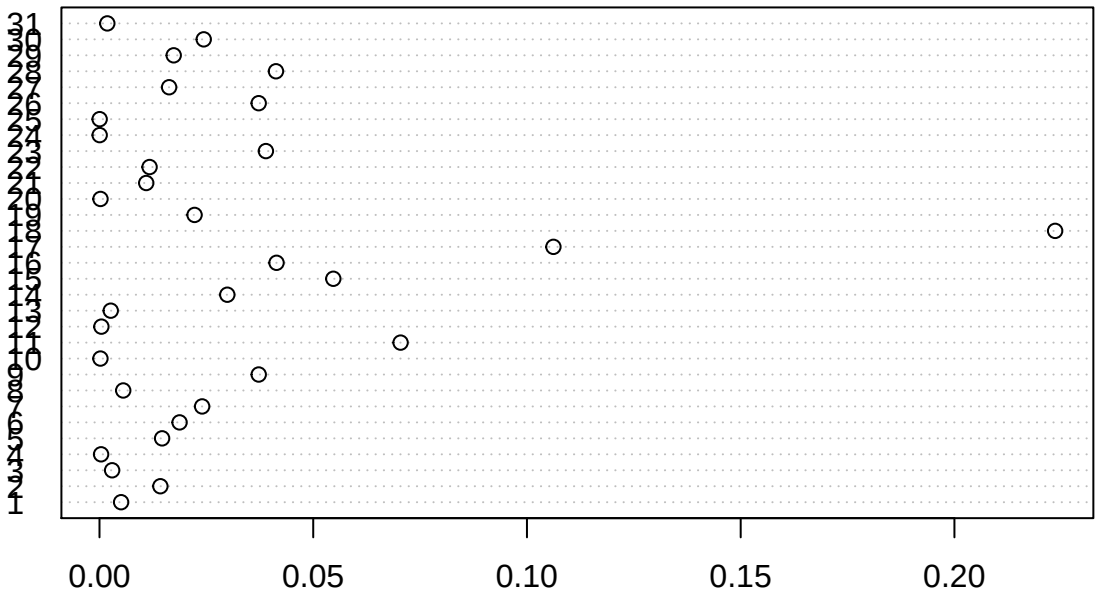
To perform diagnostics we can look at leverage

```
h=hatvalues(logfit1)
dotchart(h)
p=logfit1$rank # rank of X-matrix (including intercept column)
abline(v=p/n)   # average leverage value
abline(v=2*p/n)
```



Or Cook's distance

```
d=cooks.distance(logfit1)
dotchart(d)
```




```
data[(1:n)[d>0.10],]
```

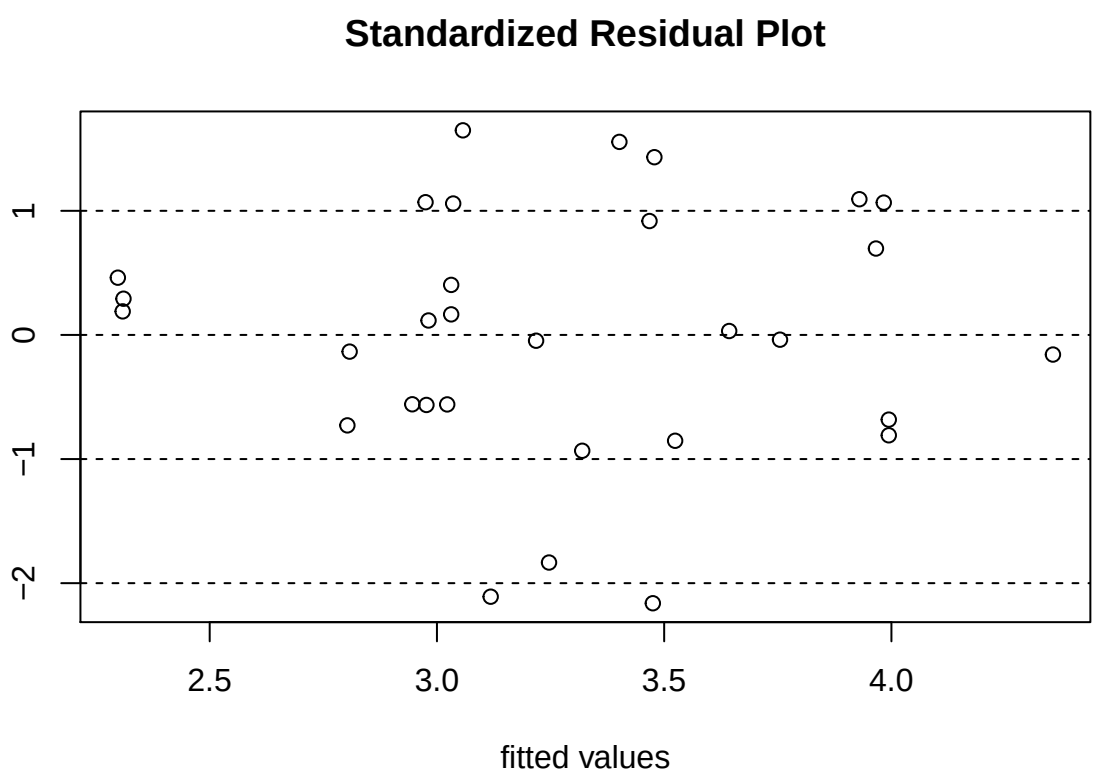
```
##       D  H    V
## 17 12.9 85 33.8
## 18 13.3 86 27.4
```

```
data[d>0.10,]  # equivalent
```

```
##       D  H    V
## 17 12.9 85 33.8
## 18 13.3 86 27.4
```

And standardized residuals

```
r=rstandard(logfit1)
f=logfit1$fitted
plot(r~f,xlab="fitted values",ylab="",main="Standardized Residual Plot")
abline(h=-2:2,lty=2)
```




```
data[(1:n)[abs(r)>2],]
```

```
##       D  H    V
## 15 12.0 75 19.1
## 18 13.3 86 27.4
```

We can identify influential points using

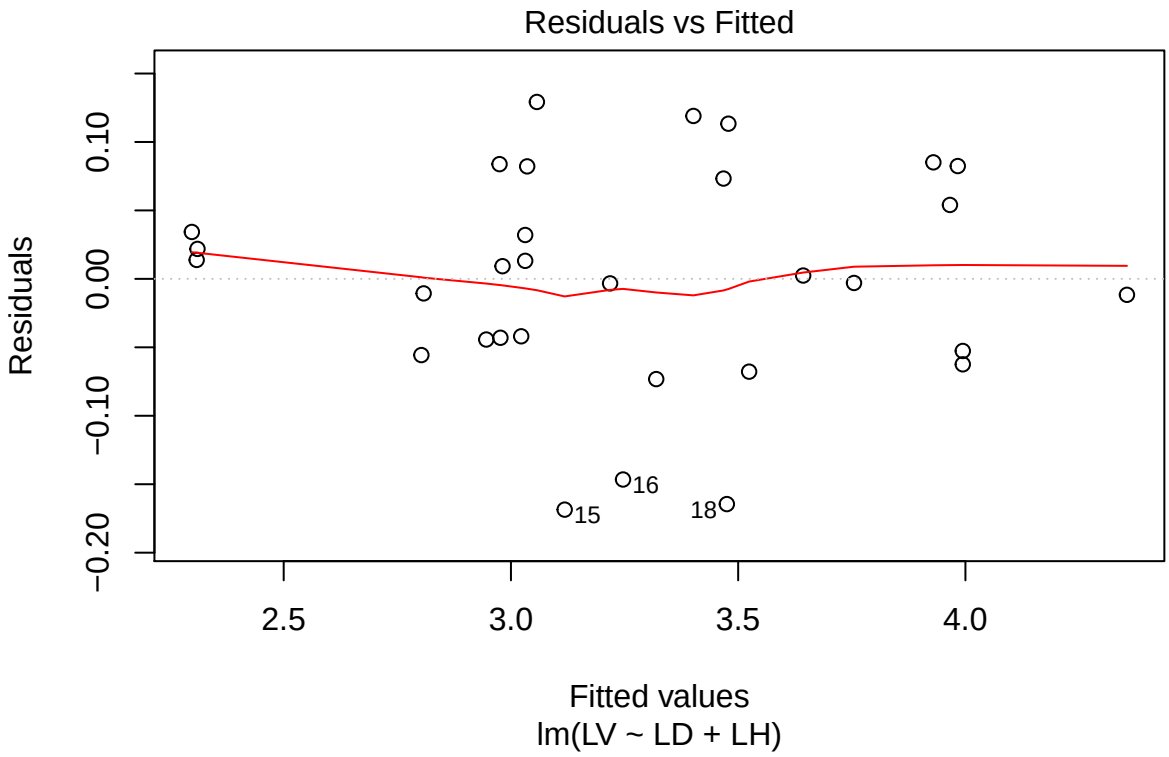
```
identify(f,r,n=2) # identify 2 points in the plot by clicking on them
```

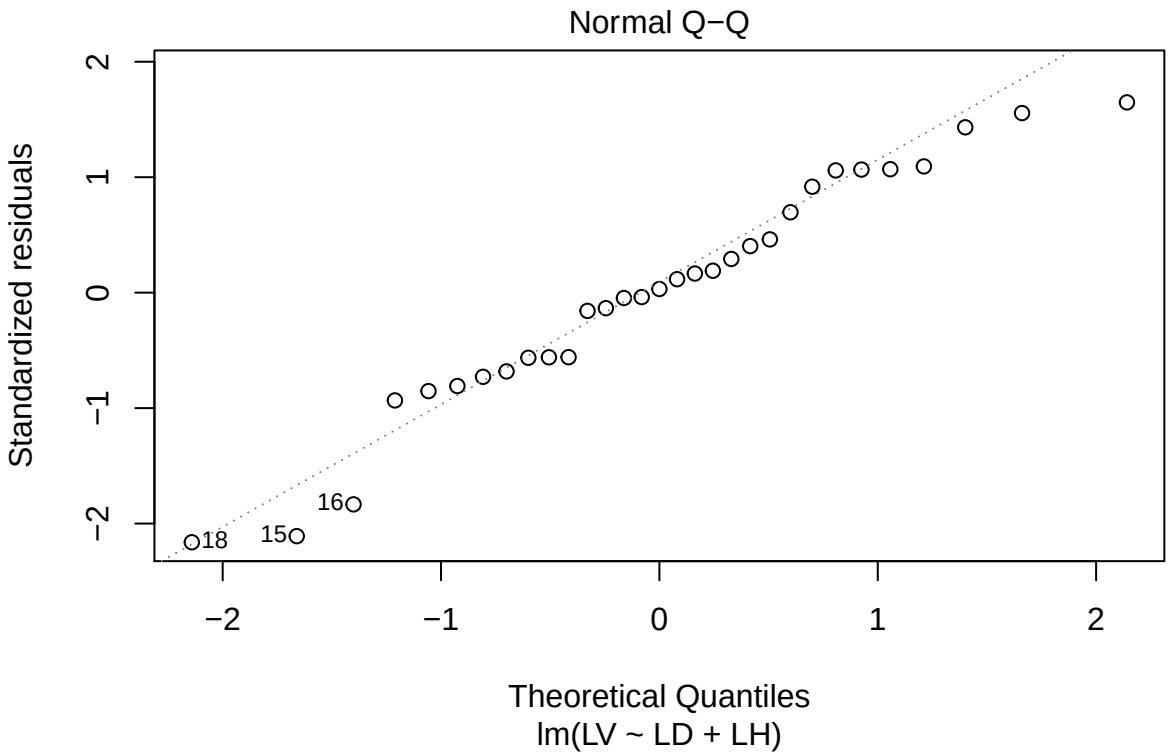
or

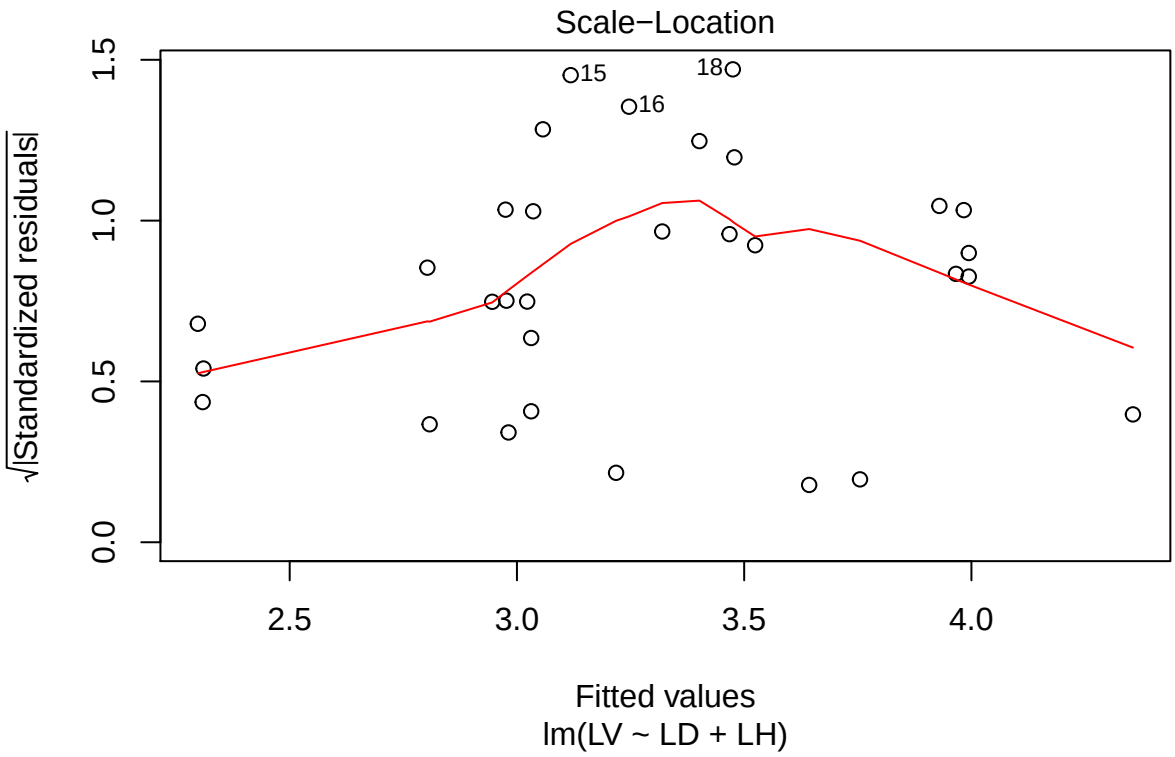
```
identify(f,r) # identify any number of points. Esc to exit.
```

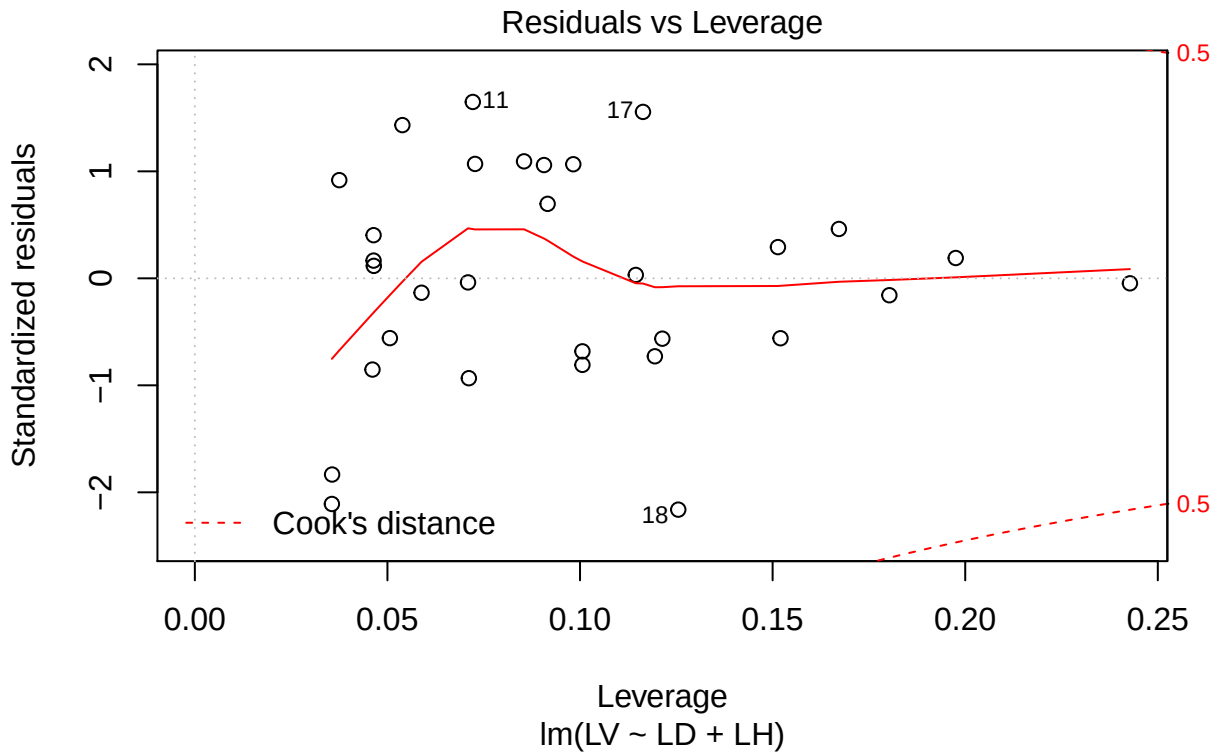
There are generic built in plots

```
plot(logfit1) # produces four diagnostic plots
```


Residuals vs Fitted
Residuals
0.10
0.00
−0.10
−0.20
2.5
3.0
3.5
4.0
16
15
18
Fitted values
lm(LV ~ LD + LH)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(LV ~ LD + LH)
18
15
16
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
15
16
18
2.5
3.0
3.5
4.0
Fitted values
lm(LV ~ LD + LH)



Or we can manually look at standard diagnostic measures:

```
im=influence.measures(logfit1) # computes standard diagnostic measures
im # how to print only flagged cases?
```

```
## Influence measures of
##   lm(formula = LV ~ LD + LH) :
##
##      dfb.1_   dfb.LD   dfb.LH   dffit cov.r   cook.d    hat inf
## 1   0.01744 -0.09396  0.00553  0.1212 1.302 5.06e-03 0.1514
## 2   0.12437 -0.08484 -0.09257  0.2038 1.309 1.42e-02 0.1672
## 3   0.06951 -0.02237 -0.05737  0.0926 1.384 2.96e-03 0.1975   *
## 4  -0.00842  0.01668  0.00359 -0.0330 1.183 3.77e-04 0.0589
## 5   0.13180  0.15568 -0.15468 -0.2070 1.227 1.47e-02 0.1214
## 6   0.16739  0.17165 -0.19034 -0.2341 1.272 1.87e-02 0.1520
## 7  -0.22381 -0.02794  0.20611 -0.2661 1.196 2.40e-02 0.1194
## 8   0.01342  0.07588 -0.03026 -0.1274 1.136 5.55e-03 0.0506
## 9  -0.19975 -0.23545  0.23495  0.3350 1.085 3.72e-02 0.0906
## 10 -0.00210 -0.01375  0.00523  0.0253 1.168 2.20e-04 0.0465
## 11 -0.25264 -0.31544  0.30206  0.4750 0.885 7.04e-02 0.0722
## 12 -0.00748 -0.01979  0.01155  0.0359 1.166 4.46e-04 0.0464
## 13 -0.01824 -0.04826  0.02816  0.0876 1.149 2.64e-03 0.0464
## 14  0.22386  0.03779 -0.20728  0.3003 1.061 2.99e-02 0.0727
## 15 -0.00944  0.12365 -0.02510 -0.4337 0.688 5.47e-02 0.0356
## 16 -0.11770 -0.06012  0.11365 -0.3690 0.788 4.14e-02 0.0356
## 17 -0.47697 -0.26149  0.49296  0.5799 0.962 1.06e-01 0.1163
## 18  0.74505  0.35129 -0.75714 -0.8811 0.737 2.24e-01 0.1255
```

```
## 19 -0.17507 -0.13654  0.18502 -0.2577 1.092 2.22e-02 0.0711
## 20 -0.02307 -0.01505  0.02396 -0.0259 1.472 2.32e-04 0.2428   *
## 21 -0.04019  0.03363  0.03181  0.1806 1.057 1.09e-02 0.0375
## 22  0.08758 -0.01339 -0.07886 -0.1866 1.080 1.17e-02 0.0461
## 23  0.14652  0.20802 -0.17312  0.3486 0.938 3.89e-02 0.0539
## 24  0.00650  0.00880 -0.00769  0.0112 1.259 4.34e-05 0.1145
## 25 -0.00145 -0.00753  0.00283 -0.0104 1.200 3.72e-05 0.0709
## 26 -0.07836  0.20756  0.02925  0.3355 1.070 3.72e-02 0.0855
## 27 -0.06859  0.12672  0.03668  0.2190 1.165 1.63e-02 0.0916
## 28 -0.03022  0.25760 -0.02519  0.3528 1.092 4.13e-02 0.0982
## 29  0.01786 -0.16669  0.01788 -0.2260 1.179 1.74e-02 0.1006
## 30  0.02124 -0.19825  0.02127 -0.2688 1.155 2.44e-02 0.1006
## 31  0.03227 -0.04227 -0.02051 -0.0728 1.357 1.83e-03 0.1803   *
```

```
# a case is flagged if:
# - any of its absolute dfbetas values are larger than 1, or
# - its absolute dffits value is larger than 3*sqrt(p/(n-p)), or
# - abs(1 - covratio) is larger than 3*p/(n-p), or
# - its Cook's distance is larger than the 50% percentile of
#   an F-distribution with p and n-p degrees of freedom, or
# - its hatvalue is larger than 3*p/n,
names(im) # is.inf = is influential
```

```
## [1] "infmat" "is.inf" "call"
```

```
i=(1:n)[apply(im$is.inf,1,sum)>0]; i
```

```
## [1]  3 20 31
```

```
im$infmat[i,]
```

```
##         dfb.1_      dfb.LD      dfb.LH       dffit    cov.r       cook.d
## 3   0.06950554 -0.02237133 -0.05736942  0.09255489 1.384455 0.0029574158
## 20 -0.02306766 -0.01505035  0.02396247 -0.02590395 1.472493 0.0002319377
## 31  0.03226841 -0.04227489 -0.02051500 -0.07279172 1.356970 0.0018299942
##          hat
## 3  0.1975359
## 20 0.2427687
## 31 0.1803079
```

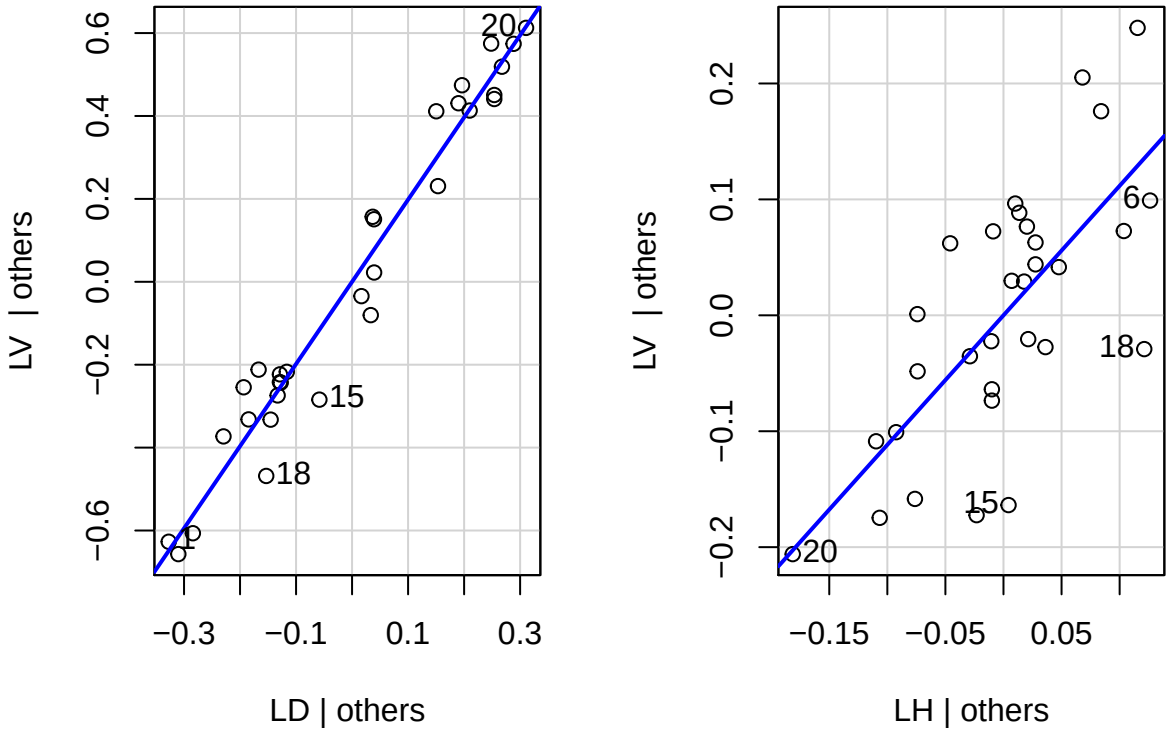
There is also an added variable plot to visualize the association of one variable after controling for the other

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.5.1
```

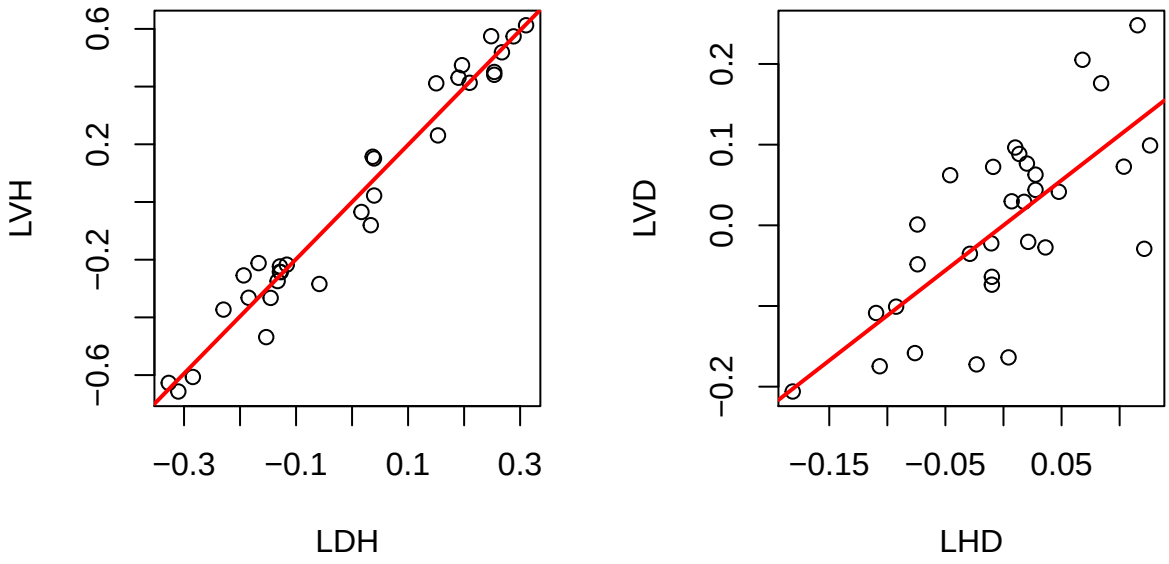
```
## Loading required package: carData
```

```
avPlots(logfit1) # added variable plots for LD and LH
```



Let's construct these manually:

```
LDH=lm(LD~LH)$resid # extract residuals from regression of LD on LH
LHD=lm(LH~LD)$resid
LVH=lm(LV~LH)$resid
LVD=lm(LV~LD)$resid
par(mfrow=c(1,2),oma=c(0,0,3,0)) # multi-frame row-wise, side by side plots
plot(LVH~LDH); abline(lm(LVH~LDH),col="red",lwd=2)
plot(LVD~LHD); abline(lm(LVD~LHD),col="red",lwd=2)
mtext("Added-Variable Plots",outer=TRUE,cex=1.2)
```



We can also put multiple graphics in one window

```
dev.new() # new graphics window
avPlots(logfit1)
dev.set(which=4) # switch to another graphics window
```

```
## pdf
##   2
```

```
dev.off() # switch off graphics window
```

```
## pdf
##   3
```

Or print them out as a hard copy

```
pdf("AVPlots.pdf"); avPlots(logfit1); dev.off() # save plot as PDF
```

```
## pdf
##   2
```

Finally, we could consider making model more complex and fit an MLR model to logged variables with interaction

```
logfit2=lm(LV~LD+LH+LD:LH) # LD:LH produces interaction term only
summary(logfit2)
```

```
##
## Call:
## lm(formula = LV ~ LD + LH + LD:LH)
##
```

```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.165941 -0.048613  0.006384  0.062204  0.132295
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.6869     7.6996  -0.479    0.636
## LD            0.7942     3.0910   0.257    0.799
## LH            0.4377     1.7788   0.246    0.808
## LD:LH         0.2740     0.7124   0.385    0.704
##
## Residual standard error: 0.08265 on 27 degrees of freedom
## Multiple R-squared:  0.9778, Adjusted R-squared:  0.9753
## F-statistic: 396.4 on 3 and 27 DF,  p-value: < 2.2e-16
```

This is the same as

```
logfit2=lm(LV~LD*LH) # LD*LH produces main effects and interactions
summary(logfit2) # compare with summary of logfit1
```

```
##
## Call:
## lm(formula = LV ~ LD * LH)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.165941 -0.048613  0.006384  0.062204  0.132295
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.6869     7.6996  -0.479    0.636
## LD            0.7942     3.0910   0.257    0.799
## LH            0.4377     1.7788   0.246    0.808
## LD:LH         0.2740     0.7124   0.385    0.704
##
## Residual standard error: 0.08265 on 27 degrees of freedom
## Multiple R-squared:  0.9778, Adjusted R-squared:  0.9753
## F-statistic: 396.4 on 3 and 27 DF,  p-value: < 2.2e-16
```

```
anova(logfit2) # compare with anova table for logfit1
```

```
## Analysis of Variance Table
##
## Response: LV
##           Df Sum Sq Mean Sq   F value    Pr(>F)
## LD         1 7.9254  7.9254 1160.1177 < 2.2e-16 ***
## LH         1 0.1978  0.1978   28.9509 1.097e-05 ***
## LD:LH      1 0.0010  0.0010    0.1479    0.7035
## Residuals 27 0.1845  0.0068
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cor(cbind(LD,LH,LV,LD*LH)) # LD and LD*LH are highly correlated
```

```
##           LD        LH        LV
## LD 1.0000000 0.5301949 0.9766649 0.9872614
## LH 0.5301949 1.0000000 0.6486377 0.6574216
```

```
## LV 0.9766649 0.6486377 1.0000000 0.9888084
##    0.9872614 0.6574216 0.9888084 1.0000000
```

Deletion diagnostics by hand

We will use the cherry tree data from the previous lab to calculate the influence of x_i on β by hand. That is, we will find $\hat{\beta} - \hat{\beta}_{(i)}$. This is the first step to calculating DFBETAs. From class we have the formula

$$\hat{\beta} - \hat{\beta}_{(i)} = \frac{(X^T X)^{-1} x_i \hat{e}_i}{1 - h_{ii}}$$

which we can calculate for all values at once with

```
dfbeta.quick = t( solve(t(X)%*%X)%*%t(X)%*%diag( as.vector(resid/(1-diag(H))) )
```

(why is this right?)

Alternatively, we can do this manually:

```
dfbeta.loop = matrix(NA,31,3)
for(i in 1:31){
  t.mod = lm(log(V)~log(D)+log(H),data=data[-i,])
  t.betahat = t.mod$coef

  dfbeta.loop[i,] = (betahat - t.betahat)
}
colnames(dfbeta.loop) = colnames(dfbeta.quick)
```

And we can test this with

```
all.equal(dfbeta.quick, dfbeta.loop, tolerance=1e-4)
```

```
## [1] TRUE
```

(note that setting the columnames for `dfbeta.quick` was needed to ensure that `R` decided it *could* compare the two).

Bonus: The other component we need for DFBETAs is $\sigma^2_{(i)}$; calculate this without emplying a `for` loop.

On Contrasts and Coding

This section looks at how different covdings of categorical covariates changes the contrast matrix. Here we consider a hypothetical 4-category factor, with means in each level $\mu_1, \ldots, \mu_4$. We will also consider the hypothesis

$$H_0 : \mu_4 = \frac{1}{3}(\mu_1 + \mu_2 + \mu_3)$$

That μ_4 is the same as the average response in the other levels.

a. If X gives the design matrix in reference coding (with level 1 as the reference), what contrast do you need to test H_0?

b. If X gives the design matrix in effect coding, what contrast do you need to test H_0?

Now let's suppose that these are actually the combinations of two 2-level factors, A and B; μ_1 corresponds to A_1B_1, μ_2 to A_1B_2, μ_3 to A_2B_1 and μ_4 to A_2B_2.

c. If X is coded in reference coding, how would you test for a difference in A, averaged over B?

d. In effect coding, how would you test for the interraction of A and B?

Contrasts in `R`

Let's look at contrasts in a real-world setting. The data in `golfballs.dat` provides the distance traveled of each of 4 brands of golf balls over 5 hits.

a. Load in these data and provide an analysis through the `lm` function (notice that `R` automatically ccounts for the brand being a factor and accounts for this).

b. What are the average distances in each brand?

c. How would you contrast brand A against the average of the other 3? Write out a contrast to do this.

d. Extract the covariance of the estimated coefficients from the `summary` function for `lm` and use this to test your contrast.

e. What happens if you add the argument `contrasts=list(brand="contr.sum")` to your your `lm` statement? (Try `help(contr.sum)`). What if you set `contrasts=contr.helmert`?