MAST09104: Introduction to Statistical Learning

Week 5 Workshop and Lab Solutions

Workshop questions

Note that some of these involve R.

1. The following questions are about leverage

   (a) Suppose that H is an idempotent and symmetric matrix. Show that the diagonal entries of H are in $[0, 1]$ and that their sum is the rank of H.

   Solution: If H is idempotent and symmetric, then it is easy to show that $I - H$ is also idempotent and symmetric. Let v_j denote the vector with 1 at the j-th entry, and 0 otherwise. Then, $h_{jj} = v_j^T H v_j \underbrace{=}_{idempotent} v_j^T H^2 v_j \underbrace{=}_{symmetry} v_j^T H^T H v_j = \|Hv_j\|^2 \geq 0$.

   On the other hand, $1 - h_{jj} = v_j^T(I - H)v_j = v_j^T(I - H)^2 v_j = v_j^T(I - H)^T(I - H)v_j = \|(I - H)v_j\|^2 \geq 0$.

   Since $h_{jj} \geq 0$ and $1 - h_{jj} \geq 0$, we have $0 \leq h_{jj} \leq 1$.
   Suppose H is p by p. Since H is idempotent and symmetric, we have $\sum_{j=1}^{p} h_{jj} = tr(H) = r(H)$ by theorem 2.4.

   (b) Interpret part (a) in terms of leverage.
   Solution: Note that the leverage of obs. i is the i-th diagonal entry of $H = X(X^T X)^{-1} X^T$. Since H is symmetric and idemptotent, each leverage is between 0 to 1.

   (c) The formula for the variance of the ith residual is var $(e_i) = \sigma^2(1 - h_{ii})$. Suppose observation i has leverage close to 1. What can you conclude about the sensitivity of y_i to $\widehat{y}_i$? Can we conclude that the least squares regression estimate $\widehat{\boldsymbol{\beta}}$ is sensitive to the i-th observation?

   Solution: If observation i has a leverage close to 1, then the variance of the residual is close to 0. This implies that the fitted value must be close to the response value, so the fitted value is very sensitive to the ith response variable. In fact, $\partial\widehat{y}_i/\partial y_i = h_{ii}$. However, a large leverage along is not substantial for us to identify observation i as an influential outlier. We would need to compute the Cook's distance for observation i, which is defined as the weighted distance between the parameter estimates with and without the ith observation. In fact, it is proportional to the product of the square of the estimated standardised residual and the leverage, by the formula from lectures. A combination of them determines when an observation is an influential outlier.

   There is a good demonstration of this available in Mathematica. The demonstration allows you to alter one point dynamically and see the effect on the least squares line. The demonstration is available at the following web address: Influential Points in Regression.

2. For simple linear regression, $y = \beta_0 + \beta_1 x + \varepsilon$, show that a $100(1 - \alpha)\%$ confidence interval for the mean response when $x = x^*$ can be written as

$$\widehat{\beta}_0 + \widehat{\beta}_1 x^* \pm t_{\alpha/2} s \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{s_{xx}}}$$

   where $s_{xx} = \sum_{i=1}^{n} x_i^2 - n\bar{x}^2$.
   Similarly, show that a $100(1 - \alpha)\%$ prediction interval for a new response when $x = x^*$ can be written as

$$\widehat{\beta}_0 + \widehat{\beta}_1 x^* \pm t_{\alpha/2} s \sqrt{1 + \frac{1}{n} + \frac{(x^* - \bar{x})^2}{s_{xx}}}.$$

Solution: For simple linear regression we have

$$\begin{aligned} X^TX &= \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \\ &= \begin{bmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix} \\ (X^TX)^{-1} &= \frac{1}{n\sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2} \begin{bmatrix} \sum_{i=1}^n x_i^2 & -\sum_{i=1}^n x_i \\ -\sum_{i=1}^n x_i & n \end{bmatrix}. \end{aligned}$$

Let $\mathbf{t} = [1\ x^*]^T$. Then our CI for the mean response when $x = x^*$ is

$$\mathbf{t}^T\widehat{\boldsymbol{\beta}} \pm t_{\alpha/2} s\sqrt{\mathbf{t}^T(X^TX)^{-1}\mathbf{t}} = \widehat{\beta}_0 + \widehat{\beta}_1 x^* \pm t_{\alpha/2} s\sqrt{\mathbf{t}^T(X^TX)^{-1}\mathbf{t}}.$$

The term in the square root is

$$\begin{aligned} \mathbf{t}^T(X^TX)^{-1}\mathbf{t} &= \frac{1}{n\sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2} \begin{bmatrix} 1 & x^* \end{bmatrix} \begin{bmatrix} \sum_{i=1}^n x_i^2 & -\sum_{i=1}^n x_i \\ -\sum_{i=1}^n x_i & n \end{bmatrix} \begin{bmatrix} 1 \\ x^* \end{bmatrix} \\ &= \frac{1}{ns_{xx}} \begin{bmatrix} \sum_{i=1}^n x_i^2 - x^*\sum_{i=1}^n x_i & -\sum_{i=1}^n x_i + nx^* \end{bmatrix} \begin{bmatrix} 1 \\ x^* \end{bmatrix} \\ &= \frac{1}{ns_{xx}}[\textstyle\sum_{i=1}^n x_i^2 - 2x^*\sum_{i=1}^n x_i + n(x^*)^2] \\ &= \frac{1}{ns_{xx}}[\textstyle\sum_{i=1}^n x_i^2 - n\bar{x}^2 + n\bar{x}^2 - 2nx^*\bar{x} + n(x^*)^2] \\ &= \frac{1}{ns_{xx}}[s_{xx} + n(x^* - \bar{x})^2] \\ &= \frac{1}{n} + \frac{(x^* - \bar{x})^2}{s_{xx}} \end{aligned}$$

as required.

The proof for the prediction interval is similar.

3. We can generate some data for a simple linear regression as follows:

```
n <- 30
x <- 1:n
y <- x + rnorm(n)
```

Construct 95% CI's for $\mathbb{E}(y)$ and 95% PI's for y (assuming that we do not know the random error variance), when $x = 1, 2, \dots, n$. Draw a graph of $\widehat{y}$ against x, and add lines for the CI and PI. What proportion of the y's should you expect to lie beyond the CI and PI?

Solution: We setup the matrix representation of the regression model as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\mathbf{y} = (y_1, \dots, y_n)^T$, $\boldsymbol{\epsilon} = (\epsilon_1, \dots, \epsilon_n)^T$, $\boldsymbol{\beta} = (\beta_0, \beta_1)^T$, and

$$\mathbf{X} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ \vdots & \vdots \\ 1 & n \end{pmatrix}$$

Note that $\sum_{i=1}^n i = \frac{1}{2}n(n+1)$ and $\sum_{i=1}^n i^2 = \frac{1}{6}n(n+1)(2n+1)$. The least squares estimates are

$$\begin{aligned}\widehat{\beta}_1 &= \frac{\sum iy_i - \overline{y}\sum i}{\sum i^2 - (\sum i)^2/n} \\ &= \frac{\sum iy_i - \frac{1}{2}\overline{y}n(n+1)}{\frac{1}{6}n(n+1)(2n+1) - \frac{1}{4}n(n+1)^2} \\ &= \frac{\sum iy_i - \frac{1}{2}\overline{y}n(n+1)}{n(n+1)(n-1)/12}\end{aligned}$$

and

$$\widehat{\beta}_0 = \overline{y} - \tfrac{1}{n}\widehat{\beta}_1 \sum i = \overline{y} - \widehat{\beta}_1(n+1)/2$$

Hence, a 95% CI for $\beta_0 + \beta_1 i$ is

$$\widehat{\beta}_0 + \widehat{\beta}_1 i \pm t_{n-2;0.025} s\sqrt{n^{-1} + (i-(n+1)/2)^2/s_{xx}},$$

where $s_{xx} = \sum i^2 - n(n+1)^2/4$. A 95% PI for y^* given $x = i$ is

$$\widehat{\beta}_0 + \widehat{\beta}_1 i \pm t_{n-2;0.025} s\sqrt{1 + n^{-1} + (i-(n+1)/2)^2/s_{xx}},$$

and

$$s^2 = \sum_{i=1}^{n}(y_i - \widehat{\beta}_0 - i\widehat{\beta}_1)^2/(n-2)$$

Graph should intersect vertical axis at $\widehat{\beta}_0$ and pass horiztonal axis at $-\widehat{\beta}_0/\widehat{\beta}_1$. The width of confidence interval and prediction intervals do not change with x. The confidence interval should always be narrower than prediction interval.

We expect around 5% of the y's to lie beyond the PI's.

Lab questions

1. In this exercise, we look at the dangers of overfitting. Generate some observations from a simple linear regression:

```
set.seed(3)
X <- cbind(rep(1,100), 1:100)
beta <- c(0, 1)
y <- X %*% beta + rnorm(100)
```

Put aside some of the data for testing and some for fitting:

```
Xfit <- X[1:50,]
yfit <- y[1:50]
Xtest <- X[51:100,]
ytest <- y[51:100]
```

(a) Using only the fitting data, estimate $\boldsymbol{\beta}$ and hence the residual sum of squares. Also calculate the residual sum of squares for the test data, that is, $\sum_{i=51}^{100}(y_i - b_0 - b_1 x_i)^2$.
Solution:

```
betafit <- solve(t(Xfit)%*%Xfit, t(Xfit)%*%yfit)
( SSfit <- sum((yfit - Xfit%*%betafit)^2) )

## [1] 38.16794

( SStest <- sum((ytest - Xtest%*%betafit)^2) )

## [1] 36.22107
```

Now add 10 extra predictor variables which we know have nothing to do with the response:

```
X <- cbind(X, matrix(runif(1000), 100, 10))
Xtest <- X[51:100,]
Xfit <- X[1:50,]
```

Again using only the fitting data, fit the linear model $\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, and show that the residual sum of squares has reduced (this has to happen). Then show that the residual sum of squares for the test data has gone up (this happens most of the time).
Explain what is going on.
Solution:

```
( betafit <- solve(t(Xfit)%*%Xfit, t(Xfit)%*%yfit) )

##              [,1]
##  [1,] -0.63651872
##  [2,]  1.01113287
##  [3,]  0.06578001
##  [4,]  0.60104434
##  [5,]  0.29716283
##  [6,]  0.22470900
##  [7,] -0.40165740
##  [8,]  0.39186181
##  [9,] -0.02118371
## [10,] -0.34238715
## [11,] -0.53698933
## [12,]  0.12811422

( SSfit <- sum((yfit - Xfit%*%betafit)^2) )

## [1] 34.27347

( SStest <- sum((ytest - Xtest%*%betafit)^2) )

## [1] 39.07738
```

With the training data, we can match some of the noise (the $\boldsymbol{\varepsilon}$ term) using the new predictor variables. However, this is of no use when applied to the test data, as there is no systematic relationship between the noise and the new variables.

(b) Repeat the above, but this time add x^2, x^3 and x^4 terms:

```
X <- cbind(X[, 1:2], (1:100)^2, (1:100)^3, (1:100)^4)
```

Solution:

```
Xfit <- X[1:50,]
Xtest <- X[51:100,]

( betafit <- solve(t(Xfit)%*%Xfit, t(Xfit)%*%yfit) )

##               [,1]
## [1,] -5.605561e-01
## [2,]  1.124641e+00
## [3,] -1.252249e-02
## [4,]  4.589517e-04
## [5,] -5.217886e-06

( SSfit <- sum((yfit - Xfit%*%betafit)^2) )

## [1] 34.94215

( SStest <- sum((ytest - Xtest%*%betafit)^2) )

## [1] 270310.6
```

Here the fit for the test data is absolutely horrendous. The problem is that the term $\beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$ is small for the training data, where it is fitting the noise term ε, but for the test data this term becomes very large, resulting in a very poor fit. So overfitting is generally a bad thing, but overfitting with polynomials can be a very bad thing, in particular when you try and apply the fit beyond the range of the fitting data.

2. What will be the output of the following code? Try to answer this without typing it up.

```
fb <- function(n) {
    if (n == 1 || n == 2) {
        return(1)
    } else {
        return(fb(n - 1) + fb(n - 2))
    }
}
fb(8)
```

Solution: `fb(n)` returns the n-th Fibonacci number, so `fb(8)` returns 21.

3. Let $A = (a_{i,j})_{i,j=1}^n$ be a square matrix, and denote by $A_{(-i,-j)}$ the matrix with row i and column j removed. If A is a 1×1 matrix then $det(A)$, the determinant of A, is just $a_{1,1}$. For $n \times n$ matrices we have, for any i,

$$det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{i,j} \, det(A_{(-i,-j)}).$$

Use this to write a *recursive* function to calculate $det(A)$.

Solution:

```
my_det <- function(X) {
if (length(X) == 1) {
return(X)
} else {
S <- 0
for (i in 1:dim(X)[1]) {
S <- S + X[1, i]*(-1)^(1+i)*my_det(X[-1, -i])
}
return(S)
}
}

X <- matrix(runif(25), nrow=5, ncol=5)
my_det(X)

## [1] 0.06332114

det(X) # R's built in version

## [1] 0.06332114
```

4. This question writes a function to calculate leverages and generate histograms for them, and applies it to three different datasets in order to better understand leverage.

   (a) In R, generate 20 observations from a standard bivariate normal distribution with correlation 0.7. (You can do this using the definition and pairs of standard independent normal random variables or using the function `rmvnorm` in the package `mvtnorm`). The three data sets are pairs $(x1, y1), (x2, y2), (x3, y3)$ with the first data set having 20 pairs of (x, y) and the second two having 21. Specifically, the three data sets are:

i. The 20 bivariate observations, labelling the first of each pair as `x1` in R, and the second as `y1`.

ii. New variables, `x2`, `y2`, which include the pairs (`x1`,`y1`) together with an additional point $(10, 10)$.

iii. New variables, `x3`, `y3`, which include pairs in (`x1`,`y1`) together with an additional point $(10, -0.3)$.

Solution:

```
x <- rmvnorm(20, mean=c(0,0),
sigma=matrix(c(1,0.7,0.7,1),ncol=2))
x1 <- x[,1]
y1 <- x[,2]
x2 <- c(x1,10)
x3 <- x2
y2 <- c(y1,10)
y3 <- c(y1,-0.3)
```

(b) Fit simple linear regression models of the response $y$to the predictor x for each of the three data sets in part (a), storing the results in `model1`, `model2`, `model3`, including the options `x=TRUE` in order that it is possible to access the X matrix of the models subsequently.

Solution:

```
model1 <- lm(y1 ~ x1,x=TRUE)
model2 <- lm(y2 ~ x2,x=TRUE)
model3 <- lm(y3 ~ x3,x=TRUE)
```

(c) Write a function, `histhat`, with argument, `model`, that uses matrix operations to find the leverage values in model and prints them out with the model formula as heading, as well as generates a histogram of the leverage values. To label the printout and the histogram, the following R commands may be helpful:

i. `writeLines`

ii. `noquote` to get text strings without quotes printed out

iii. `paste` including the text string "
`n"` to get a new line before a subsequent `print`

Solution:

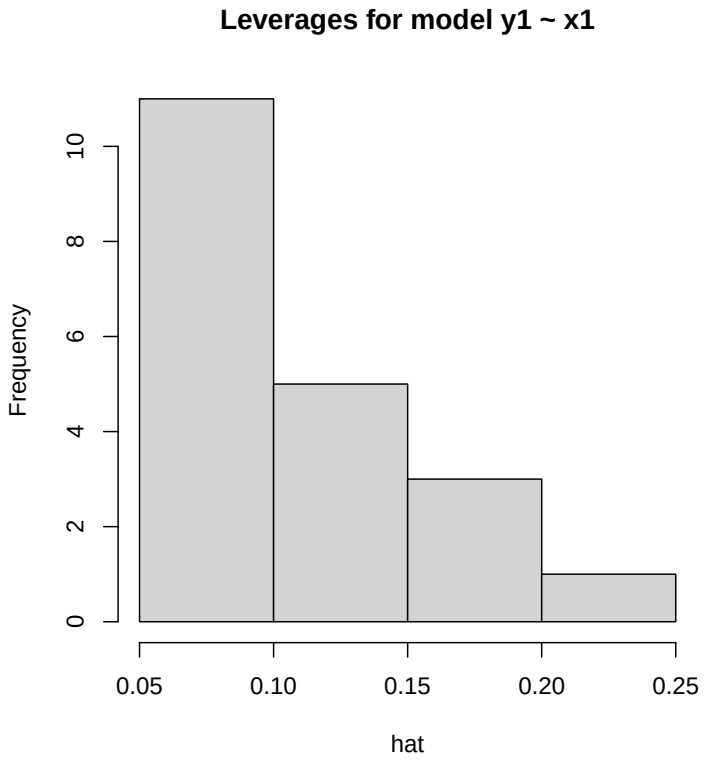
```
histhat <- function(model){
Xmat <- model$x
XX <- t(Xmat)%*%Xmat
H <- Xmat%*%solve(XX)%*%t(Xmat)
hat <- diag(H)
writeLines(
noquote(
paste("Leverages for model",model$call["formula"],"\n")
)
)
print(hat)
hist(hat,
main = paste("Leverages for model",model$call["formula"])
)
}
```

(d) Generate the printouts of leverages and histogram of leverages for the three models

Solution:

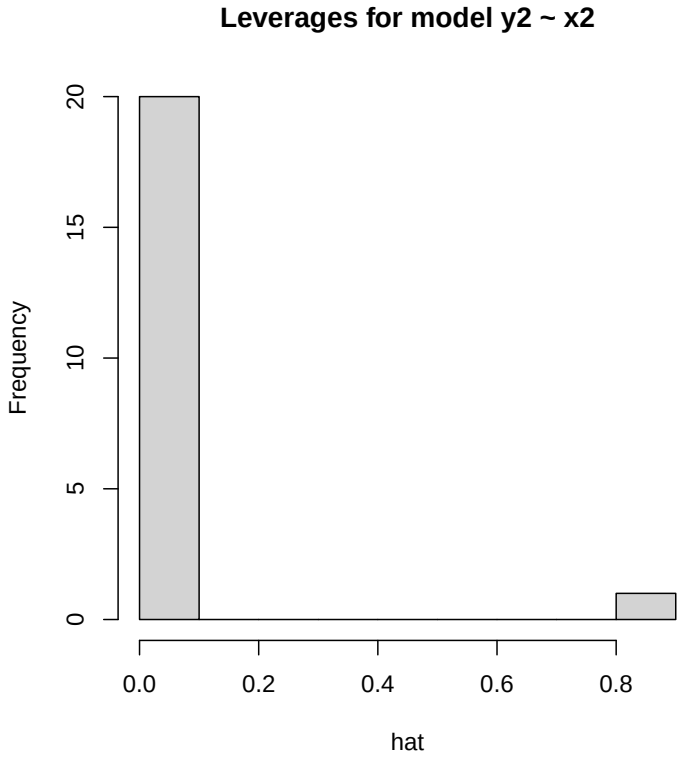
```
histhat(model1)
```

```
## Leverages for model y1 ~ x1
##
##          1          2          3          4          5          6
## 0.05082128 0.05875790 0.13381598 0.17643250 0.11626359 0.05940889
##          7          8          9         10         11         12
## 0.17586485 0.13573954 0.06235337 0.05572307 0.05419170 0.05179270
##         13         14         15         16         17         18
## 0.11022444 0.15661734 0.05131934 0.22788965 0.05917225 0.12688354
##         19         20
## 0.05196275 0.08476531
```



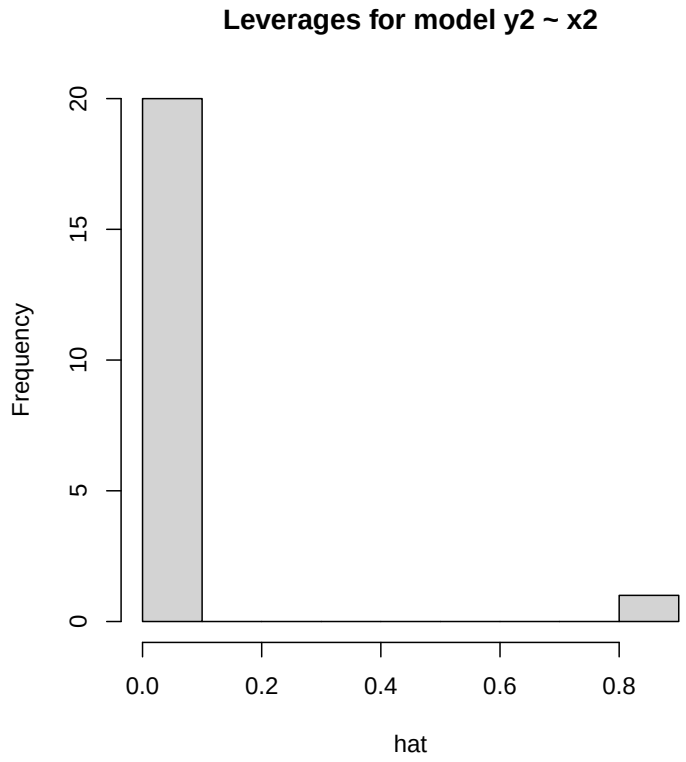

```
histhat(model2)
```

```
## Leverages for model y2 ~ x2
##
##          1          2          3          4          5          6
## 0.04866405 0.05458737 0.07499558 0.08490734 0.05163785 0.04763296
##          7          8          9         10         11         12
## 0.05846242 0.07545488 0.04762384 0.05339092 0.04791582 0.05146733
##         13         14         15         16         17         18
## 0.05102592 0.05614023 0.04846698 0.09637189 0.04763706 0.07332818
##         19         20         21
## 0.04828366 0.04875480 0.83325092
```




```
histhat(model3)
```

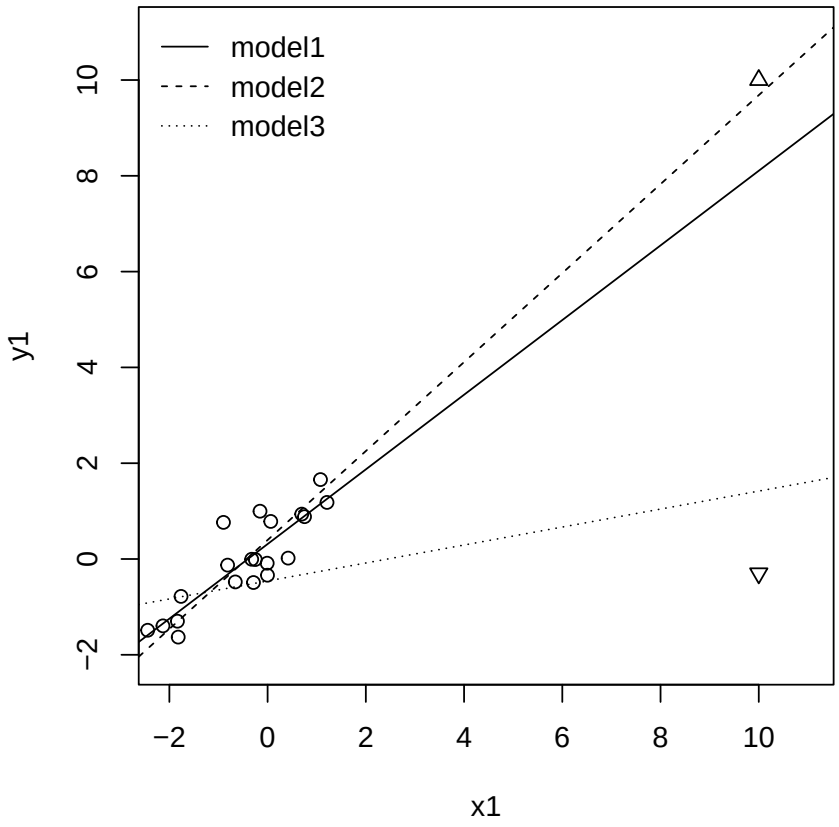
```
## Leverages for model y3 ~ x3
##
##          1          2          3          4          5          6
## 0.04866405 0.05458737 0.07499558 0.08490734 0.05163785 0.04763296
##          7          8          9         10         11         12
## 0.05846242 0.07545488 0.04762384 0.05339092 0.04791582 0.05146733
##         13         14         15         16         17         18
## 0.05102592 0.05614023 0.04846698 0.09637189 0.04763706 0.07332818
##         19         20         21
## 0.04828366 0.04875480 0.83325092
```



(e) Plot the data in `y1` versus `x1` using the options `xlim` and `ylim` to make sure that the axes go from -2.1 to 11. Add the point $(10, 10)$ with an upper triangle and point $(10, -0.3)$ with a lower triangle. Add the lines from `model2` and `model3` using dashed and dotted lines. Add the line from `model1` in a solid line using the intercept and slope inputs.

Solution: This is an example of how an influential observation can greatly affect the fitted regression line.

```
plot(x1,y1,xlim=c(-2.1,11),ylim=c(-2.1,11))
points(x=x2[21],y=y2[21],pch=2)
points(x=x3[21],y=y3[21],pch=6)
abline(model3,lty="dotted")
abline(model2,lty="dashed")
abline(a=model1$coef[1],b=model1$coef[2])
```


model1
model2
model3
y1
x1
−2
0
2
4
6
8
10