Data 501 - Assignment 2 - Solution

David Huijser

Introduction

Our intention is to create an R-package for detected outliers to linear models. We will rely heavily on linear algebra, so I will use `R` pacakge `Matrix`. I will also use package `olsrr` ONLY for verification and reference. I will also use `ggplot2` for visalization.

```
library(Matrix)
library(olsrr)
```

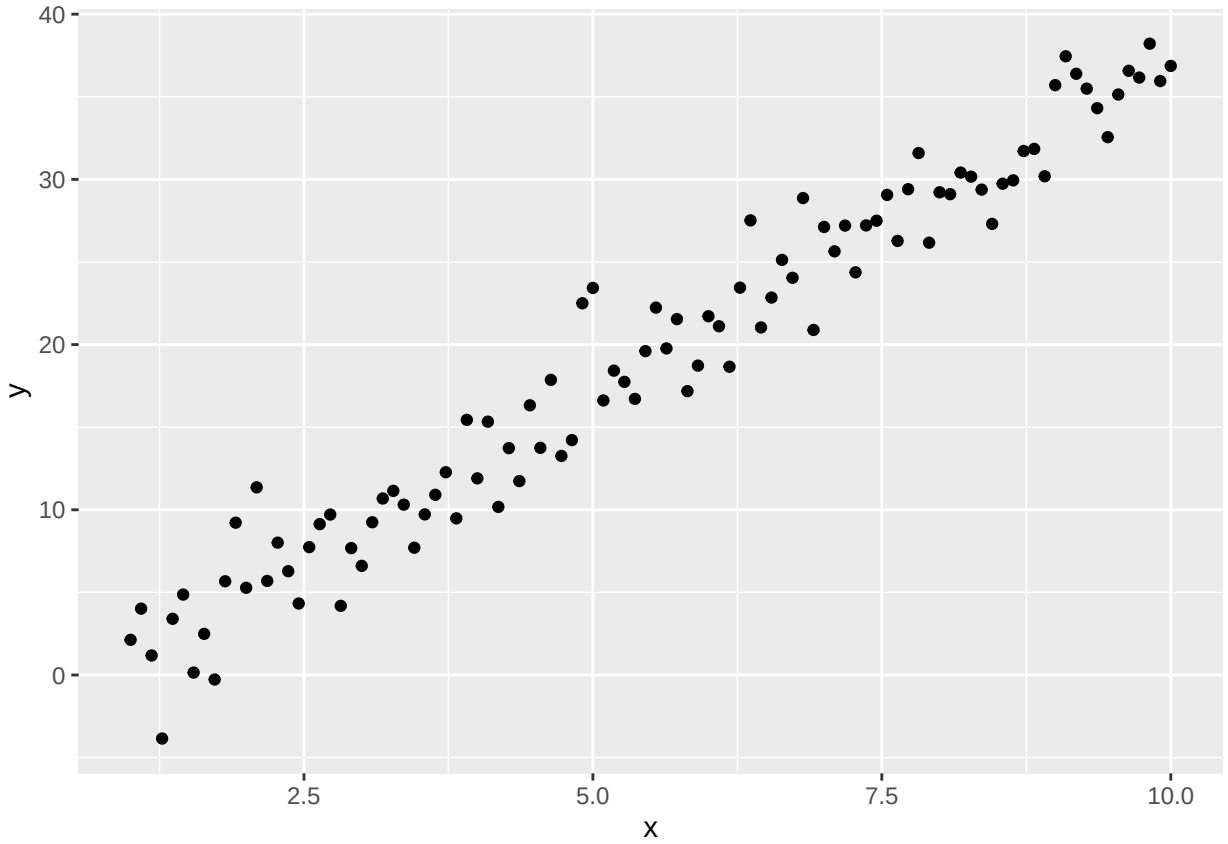
```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

```
library(ggplot2)
```

The first thing I need is to generate some test data.

```
# 1. Generate data with a few outliers
slope <-  4
offset <- -3
sds <- 7
x <- seq(from = 1, to = 10, length.out= 100)

y <- slope*x + offset

# Add regular noise
noise <- rnorm(length(x),sd=2)
y <- y + noise

# generate number of outliers
num <- 3
s <- sample(length(x), num)
y[s] <- y[s] +  rnorm(num, sd=sds) + runif(num, -sds,sds)
#index <- seq(size)

# Create plot
df <- data.frame(x,y)

ggplot(data=df, aes(x=x, y=y)) +
geom_point()
```



I would like to use as little external packages as possible and for educational purposes I dive a bit into the linear algebra in of linear models. I assume everyone has seen in their undergrad courses how find a linear model using an R package `lm` or `glm`. I assume you have seen the equation that describes a linear model

$$y = \beta_0 + \beta_1 x + e$$

where y is the dependent variable, x is the independent variable, a is the slope and b is the offset. However, if we use proper notation, we acknowledge that x and y are vectors and each values has an error associated with. Therefore we get the following

$$\mathbf{y} = \beta_0 \mathbf{x} + \beta_1$$

or in matrix form

$$\mathbf{y} = \mathbf{X}\beta$$

where $\mathbf{X}$ is a matrix [$\mathbf{1}$ $\mathbf{x}$]. Our estimate of $\hat{\beta}$ can be found directly

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

This will give vector $\beta^T = [\beta_0 \quad \beta_1]$.

I prefer to stick with the linear algebra approach, because some other quantities we use later can be easily be calculated using linear linear algebra. Finding β in `R` is straight forward.

```
# Solve linear fit
n <- length(x)
Y <- as.matrix(y)
X <- cbind(rep(1,length(x)), x)
XtX <- t(X) %*% X

# Use solve to calculates the inverse
```

```r
beta <- solve(XtX)%*%t(X)%*%Y
offset_estimate <- beta[1]
slope_estimate <- beta[2]

# Compare our solution with lm
model <- lm(y~x)

# Confirm the sum of square residuals of from the LA method are identical to those from the lm-package
residuals <- y - (slope_estimate*x + offset_estimate)
sum(residuals**2)
```

```
## [1] 536.8739
```

```r
model$coefficients
```

```
## (Intercept)           x
##   -3.117401    4.041915
```

```r
sum(model$residuals**2)
```

```
## [1] 536.8739
```

Quantities

There are a few quantities we will need. We will need the least-square residuals

$$e_i = y_i - \hat{y}$$

and the SSE (sum of square errors)

$$SSE = \sum_{i=1}^{n} e_i^2$$

We can use this to calculate the estimate of σ^2

$$\hat{\sigma}^2 = \frac{SSE}{n-p-1} = \frac{\mathbf{e}^T\mathbf{e}}{n-p-1}$$

where n is the number of observations and p the number of variables (in our case $p = 1$). The next quantity we need is the so-called projection matrix defined as

$$\mathbf{P} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$$

The elements on the diagonal,$ p{ii}$ are called the **leverage** values for the i^{th} observations. These are also needed to calculate the (internally) studentized residuals

$$r_i = \frac{e_i}{\hat{\sigma}\sqrt{1-p_{ii}}}$$

Implemented in are `R` it looks like this

```r
# Residuals
residuals <- y - (slope_estimate*x + offset_estimate)


#  Projection matrix
P <- X %*%solve(XtX)%*%t(X)
```

```r
# Leverage P
pii <- diag(P)

# Estimate of sigma
sigma_hat <- as.numeric(sqrt(t(residuals)%*%residuals/(n-2)))

# Studentize Residuals
studentsized_residuals  <- residuals/(sigma_hat*sqrt(1-pii))
```

We will now calculate the three measure and use the `R` package `olsrr` to confirms we got all our quantities correct. After this we can implement our measures.

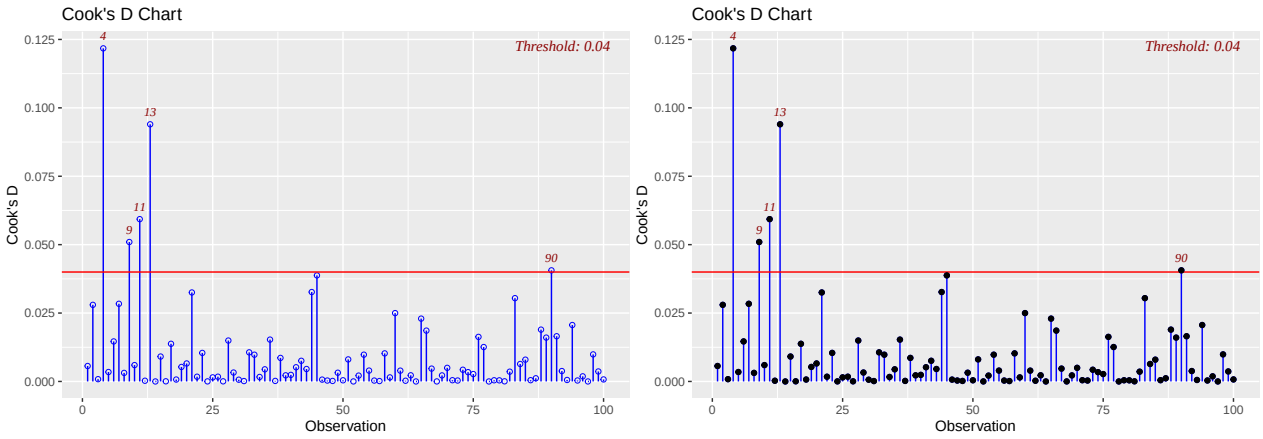
Cooks Distance

The first quantity we want to look at is the Cook distance:

$$C = \frac{r_i^2}{p+1} \times \frac{p_{ii}}{1-p_{ii}}$$

which is straight forward in `R`. We will plot the cooks distance plot for `Olsrr` package and overlay our Cook distances to confirm consitency.

```
# Calculated Cooks Distance
my_cooks_distance <- (0.5*studentsized_residuals**2)*(pii/(1-pii))

# Compare my Cooks-distance with Cooksplot from Olsrr packages by overplot my Cooksdistances (black dot
model <- lm(y~x)
temp  <-cooks.distance(model)
ols_plot_cooksd_chart(model) + geom_point(y=as.vector(my_cooks_distance))
```



Welsch and Kuh Measure

The second measure is Welsch and Kuh Measure, also called (DFITS). Before we look at the definition of this measure we need to defined two quantities which are specific to the i^{th} observation because they exclude this observation from the calculation. The first of these quantities is an alternative estimate of σ^2 which does not use the i^{th} observation

$$\hat{\sigma}^2_{(i)} = \frac{SSE_{(i)}}{n-p-2}$$

where we adopt the convention that a subscript $_{(i)}$ indicates the i^{th} observation is NOT include in the calculation. The second quantities is called the **externally** studentized residuals r^*_i. It is called externally, because because e_i is not involved in (external to) $\hat{\sigma}^2_{(i)}$.
It is defined as

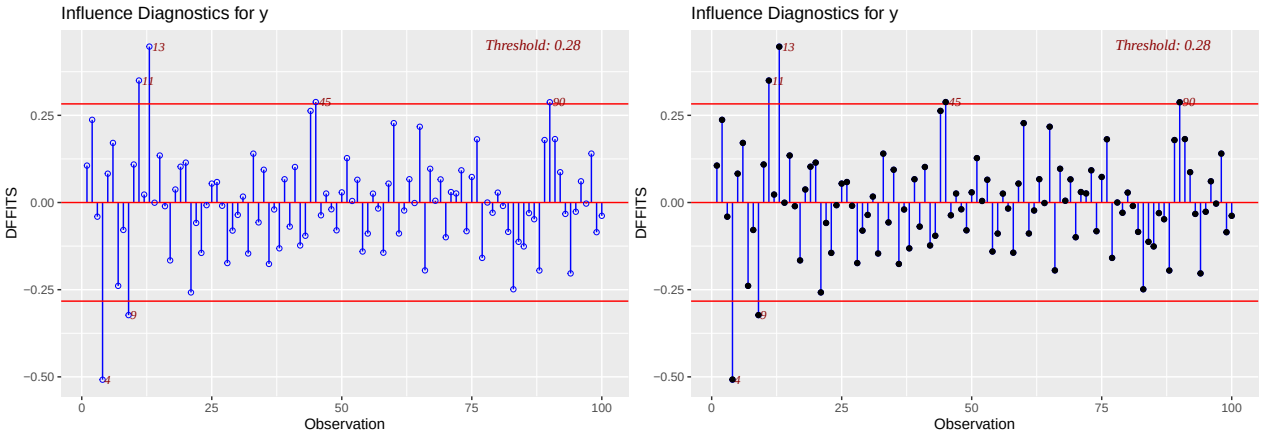
$$r^*_i = \frac{e_i}{\hat{\sigma}_{(i)}\sqrt{1-p_{ii}}}$$

With the two quantities we can calculated the DFITS define as

$$DFITS_i = r^*_i \sqrt{\frac{p_{ii}}{1-p_{ii}}}$$

In R it looks like this

```
SSE_i <- rep(0, n)
# Calculate SSE without the ith values for each i
for (i in 1:n){
 tempbool <- rep(T, n)
 tempbool[i] <- F
 SSE_i[i]  <- as.numeric(sqrt(  (t(residuals[tempbool])%*%residuals[tempbool])/(n-3)))
}

# Calculate r* and DFITS
ext_studentsized_residuals  <- residuals/(SSE_i*sqrt(1-pii))

my_DFITS <- ext_studentsized_residuals*sqrt(pii/(1-pii))

# Compare my DFITS  with DFITS from Olsrr packages by over plotting my DFITS-values  (black dots)
ols_plot_dffits(model)+
geom_point(y=as.vector(my_DFITS))
```



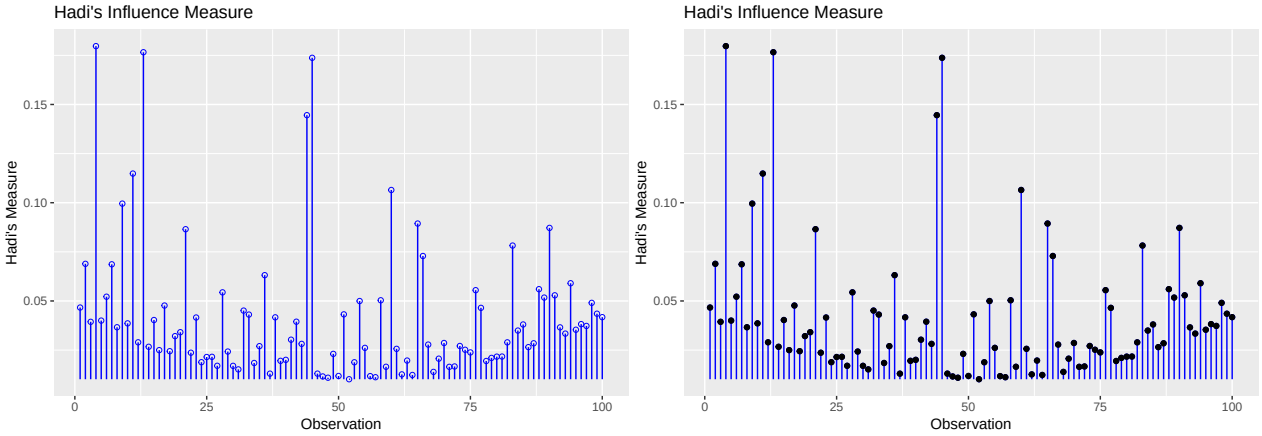
Hadis Influence Measure

Hadis Influence Measure is defined as

$$H_i = \frac{p_{ii}}{1-p_{ii}} + \frac{p+1}{1-p_{ii}} \frac{d_{ii}^2}{1-d_{ii}^2} \qquad i = 1, 2, 3, \ldots n$$

where $d_{ii} = e_i/\sqrt{SEE}$ is the normalized residual. This equation consist of two parts. The first component known as the "potential" and the second part is a function of the normalized residuals weighted by the i^{th} leverage values.

```
# Calculate the normalized residuals
dd2 <- residuals**2/as.numeric(t(residuals)%*%residuals)
# Calculate H
H <- (pii/(1-pii)) + (2/(1-pii))*(dd2)/(1-dd2)

# Plot Hadis Measure
ols_plot_hadi(model) +
geom_point(y=as.vector(H))
```



Desigm

For the design we prefer to call a function with the following arguments

- model or data.frame
- type of measure (categorical)
- type of preferred output (measure or plot)