Regression Analysis

Luke Chang

Last Revised July 16, 2010

This section will cover how to run simple correlation analysis and provide a general introduction to regression analyses in R . Finally, we will also briefly cover robust regression.

1 Correlations

1.1 Simple Correlations

At the beginning stages of analysis after running simple descriptive statistics, we are often interested in examining the strength of association between variables. Correlations can be run between two variables X and Y and also on a matrix with several variables using the cor function. As an example we will use the brain IQ dataset to examine the degree of association between all of the variables. Because we are only interested in the continuous values, we can select a smaller section of the data for the correlation matrix. The type of correlation can be changed using the method option (e.g., pearson, spearman, or tau). Before we begin we must make sure the data file is first loaded in the R session.

```
> brainIQ.data<-read.table(paste(website,"BrainIQ.csv",sep=""),sep=",",
    header=TRUE)
> cor(cbind(brainIQ.data[,1:3],brainIQ.data[,7:9]),method="pearson")
```

```
           CCMIDSA         FIQ        HC       TOTSA      TOTVOL       WEIGHT
CCMIDSA 1.00000000  0.156251519 0.6127741  0.33719800  0.66178913  0.081694540
FIQ     0.15625152  1.000000000 0.1378395 -0.29131921 -0.06339202 -0.002702510
HC      0.61277412  0.137839520 1.0000000  0.33667365  0.50791438  0.239980555
```

```
TOTSA   0.33719800 -0.291319208 0.3366737  1.00000000  0.60099543  0.064106714
TOTVOL  0.66178913 -0.063392016 0.5079144  0.60099543  1.00000000  0.207922711
WEIGHT  0.08169454 -0.002702510 0.2399806  0.06410671  0.20792271  1.000000000
```

We can see that there is no correlation between IQ and any of the other variables. If anything it seems to be negatively correlated with the total surface area. At this point I'm sure that many of you may be wondering how we know which correlations are significant. The default way to check significance of correlations in `R` is not on a matrix of data, but pairwise using the `cor.test` command.

```
> cor.test(brainIQ.data$FIQ,brainIQ.data$TOTSA,method="pearson")

        Pearson's product-moment correlation

data:  brainIQ.data$FIQ and brainIQ.data$TOTSA
t = -1.292, df = 18, p-value = 0.2127
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.6500402  0.1735784
sample estimates:
       cor
-0.2913192
```

Using this pairwise method we see that the correlation between FIQ and total surface area doesn't even approach significance. This can be more clearly illustrated graphically using a scatterplot, which reveals that there is considerable variability in the degree of association between the two variables.

```
> plot(brainIQ.data$FIQ,brainIQ.data$TOTSA)
```

For those that desperately need to see a correlation matrix with significance values there are two main approaches. Unlike other statistical analysis packages, it is possible to write your own functions in `R` which includes statistical analyses. Because we know from our undergraduate introductory statistics class that it is relatively easy to calculate a correlation coefficient and its accompanying p-value, we could just write the function ourselves. The details of creating your own functions will be discussed in a later chapter. Alternatively, we could take advantage of the open source nature of `R` and scour the internet for somebody else who has spent hours of their own time to write the function and selflessly shares it on their own personal webpage. It probably goes without saying, but it is important to be cautious of using other people's work without first checking to make sure it works correctly. Here I illustrate how to

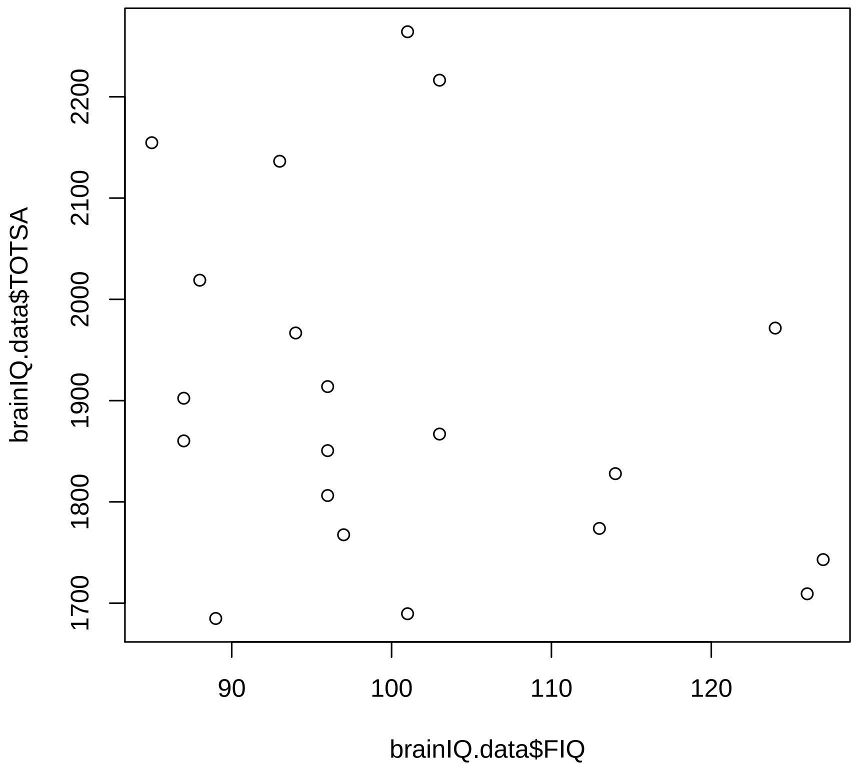

Figure 1: Scatterplot illustrating the degree of association between IQ and Total Surface Area

use somebody else's function in your own session. It is first necessary to load the function using the `source` command.

```
> source(paste(website,"cor.prob.R",sep=""))
```

Now that the function is loaded, we can use it to create another correlation matrix on the same data as earlier. This function however reports the correlation coefficients in the lower triangle of the matrix (i.e. below the diagonal) and the p-values on the upper triangle (i.e. above the diagonal)

```
> cor.prob(cbind(brainIQ.data[,1:3],brainIQ.data[,7:9]))
```

```
           CCMIDSA          FIQ          HC      TOTSA      TOTVOL    WEIGHT
CCMIDSA 1.00000000  0.510638667 0.004072067 0.14598007 0.001481729 0.7320518
FIQ     0.15625152  1.000000000 0.562236477 0.21269901 0.790616477 0.9909779
HC      0.61277412  0.137839520 1.000000000 0.14664764 0.022233887 0.3081493
TOTSA   0.33719800 -0.291319208 0.336673652 1.00000000 0.005070081 0.7883070
TOTVOL  0.66178913 -0.063392016 0.507914378 0.60099543 1.000000000 0.3790457
WEIGHT  0.08169454 -0.002702510 0.239980555 0.06410671 0.207922711 1.0000000
```

1.2 Partial Correlations

Occasionally we are interested in examining the relationship between two variables while controlling for a third variable. There isn't a default function in `R` to do this, because it can also be easily performed using the regression function. However, just like the correlation significance function we used earlier, we can simply use somebody else's hard work. In this case we will load the `pcor.test` function by using the `source("pcor.R")` command. To use this function it is necessary to specify an `X` a `Y`, and a `Z`, which can be either a single variable or a data frame of several variables. For example, suppose we were interested in examining the relationship between IQ and the the total surface area, while taking into account the surface area of the corpus callosum.

```
> source(paste(website,"pcor.R",sep=""))
> pcor.test(brainIQ.data$FIQ,brainIQ.data$TOTSA,brainIQ.data$CCMIDSA)
```

```
    estimate   p.value statistic  n gn  Method            Use
1 -0.3699515 0.1006241 -1.641835 20  1 Pearson Var-Cov matrix
```

We can see that the negative relationship between IQ and total surface area increases after taking into account the corpus callosum surface area.

2 Model Building

Building models is one of the most relied on tools of statistical analysis and fortunately is very straightforward in `R` . This section will cover the basics of model building in `R` including running a regression analysis using the `lm` function, adding factors, interactions, and polynomials, and comparing models.

2.1 Sum of Squares

It is important to note that `lm` uses Type I Sum of Squares, which means that it sequentially adds the predictors to the model. This is different than the defaults in SPSS and SAS which employ the Type III Sum of Squares and means that order of entry matters, with shared variance going to the preceding terms.

2.2 Using lm

The standard function to run a regression is the `lm` function (linear model). To specify a model it is necessary to indicate that the dependent variable Y is predicted `~` by the independent variables X. It is also necessary to specify a data file `(data=?)`.

For this section we are going to use a dataset from an experiment using the Ultimatum Game, which is a simple financial bargaining task from behavioral economics.[1] In this game there are two players. Player A is endowed with a sum of money, let's say \$10, and is charged with the task of splitting the money between them any way they see fit. The twist to this game is Player B ultimately decides if they want to accept the offer as proposed, in which case both players receive the proposed division of money, or reject the offer, in which case neither player receives any money. We already know that participants usually reject unfair offers with greater frequency playing with humans compared to computers (Sanfey, et al. 2003). In this dataset we are interested in looking at decision conflict (Pochon et al., 2009) by examining the amount of time it takes to make a decision. We assume that longer reaction times (RT) indicate greater decision conflict.

First we need to load the dataset.

[1] This data is taken from Chang & Sanfey (2009) Frontiers in Behavioral Neuroscience and Chang, Smith, & Sanfey (In Preparation)

```
> data<-read.table(paste(website,"UG_Data.txt",sep=""),header=TRUE,
    na.strings=999999)
```

Next we will build our first model by testing the question "Does the amount of money offered predict changes in RT?"

```
> m1<-lm(RT~Offer,data=data)
```

To examine the results of our analysis, we look at a `summary` of `m1`

```
> summary(m1)

Call:
lm(formula = RT ~ Offer, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-2135.5  -765.4  -359.2   507.1  5913.6

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2260.71      97.00   23.31  < 2e-16 ***
Offer        -125.17      31.06   -4.03 6.25e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1170 on 646 degrees of freedom
Multiple R-squared: 0.02452,        Adjusted R-squared: 0.02301
F-statistic: 16.24 on 1 and 646 DF,  p-value: 6.253e-05
```

We see that overall unfair offers seem to increase decision conflict.

2.3 Adding factors, interactions, and polynomials

However, we know that people often adhere to simple decision rules, such as "I'm only going to accept offers greater than \$2, and I will always accept \$5 offers." This means that we may need to model a quadratic term for the amount of money offered. To do this we use the `I()` identity function to specify a squared term.

```
> m2<-lm(RT~Offer+I(Offer^2),data=data)
> summary(m2)
```

```
Call:
lm(formula = RT ~ Offer + I(Offer^2), data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-2143.0  -779.4  -285.3   529.8  5781.0

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1431.87     205.87   6.955 8.67e-12 ***
Offer         587.64     159.78   3.678 0.000255 ***
I(Offer^2)   -116.04      25.53  -4.545 6.55e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1152 on 645 degrees of freedom
Multiple R-squared: 0.05479,        Adjusted R-squared: 0.05186
F-statistic:  18.7 on 2 and 645 DF,  p-value: 1.280e-08
```

Interestingly, we see that adding the quadratic term reverses the direction of the amount of offer, with larger offers resulting in slower response times. More importantly we observe a negative parameter for the quadratic term, which indicates, consistent with our hypothesis, that people have greater decision conflict when the offers deviate from their simple heuristics and become more ambiguous. Next we might be interested in seeing if this effect interacts with the type of partner with which they are playing (e.g., a computer or a human). `R` automatically will add main effects terms if you only specify the interaction term in the model. In addition, the default mode in `R` is to use dummy codes, which means you must specify a reference group if there are nominal factors. First it is useful to check and see if the factor is ordered using the `is.ordered` function. Next it is important to make sure that the correct level of the factor is being set as the reference group. This can be inspected using the `levels` command, with the first level being the reference. The reference group can be easily changed if necessary using the `relevel` command.

```
> is.ordered(data$Condition)

[1] FALSE

> levels(data$Condition)

[1] "Computer" "Human"
```

```
> data$Condition<-relevel(data$Condition,ref="Computer")
```

After ensuring that the factor has the correct reference group, we can now proceed with the analysis.

```
> m3<-lm(RT~Offer*Condition+I(Offer^2),data=data)
> summary(m3)

Call:
lm(formula = RT ~ Offer * Condition + I(Offer^2), data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-2198.9  -793.5  -278.7   508.7  5725.1 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)           1577.55     246.42   6.402 2.96e-10 ***
Offer                  542.73     165.64   3.276  0.00111 ** 
ConditionHuman        -218.53     202.82  -1.077  0.28169    
I(Offer^2)            -116.04      25.55  -4.542 6.64e-06 ***
Offer:ConditionHuman    67.36      64.96   1.037  0.30012    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1153 on 643 degrees of freedom
Multiple R-squared: 0.05655,	Adjusted R-squared: 0.05068 
F-statistic: 9.635 on 4 and 643 DF,  p-value: 1.43e-07
```

Here we find that their is no main effect for the type of partner nor is there an interaction with the amount of money offered in predicting RT. However, we are actually interested in the interaction with the quadratic term, so we will now add a new interaction term to the model.

```
> m4<-lm(RT~Offer*Condition+I(Offer^2)*Condition,data=data)
> summary(m4)

Call:
lm(formula = RT ~ Offer * Condition + I(Offer^2) * Condition, 
    data = data)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-2159.3  -772.4  -272.6   521.8  5813.6 

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)    
(Intercept)                2130.50     355.81   5.988 3.54e-09 ***
Offer                        67.19     276.15   0.243   0.8078    
ConditionHuman            -1047.95     435.77  -2.405   0.0165 *  
I(Offer^2)                  -38.63      44.12  -0.875   0.3817    
Offer:ConditionHuman        780.66     338.22   2.308   0.0213 *  
ConditionHuman:I(Offer^2)  -116.12      54.04  -2.149   0.0320 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 1150 on 642 degrees of freedom
Multiple R-squared: 0.06328,	Adjusted R-squared: 0.05599 
F-statistic: 8.675 on 5 and 642 DF,  p-value: 5.705e-08 
```

Here we observe a significant interaction term for the quadratic offer amount and the type of partner. This supports our hypothesis that people have more decision conflict when trying to decide to accept or reject ambiguously fair offers from humans in comparison to computers.

We can plot this effect by fitting separate models for each condition (e.g., Human and Computer). We then extract the parameter estimates using the `coef` function and use matrix multiplication to create a vector of predicted `Y` values by multiplying the matrix of `X` values by the parameter estimates. Next we create a data frame containing the predicted RT values for each offer amount and plot a line going through these points using spline interpolation.

```
> h<-subset(data,data$Condition=="Human")
> c<-subset(data,data$Condition=="Computer")
> mh<-lm(RT~Offer+I(Offer^2),data=h)
> mc<-lm(RT~Offer+I(Offer^2),data=c)
> coefh<-coef(mh)
> coefc<-coef(mc)
> yh<-cbind(1,h$Offer,h$Offer^2) %*% coefh
> yc<-cbind(1,c$Offer,c$Offer^2) %*% coefc
> hpt<-cbind(h$Offer,yh)
```

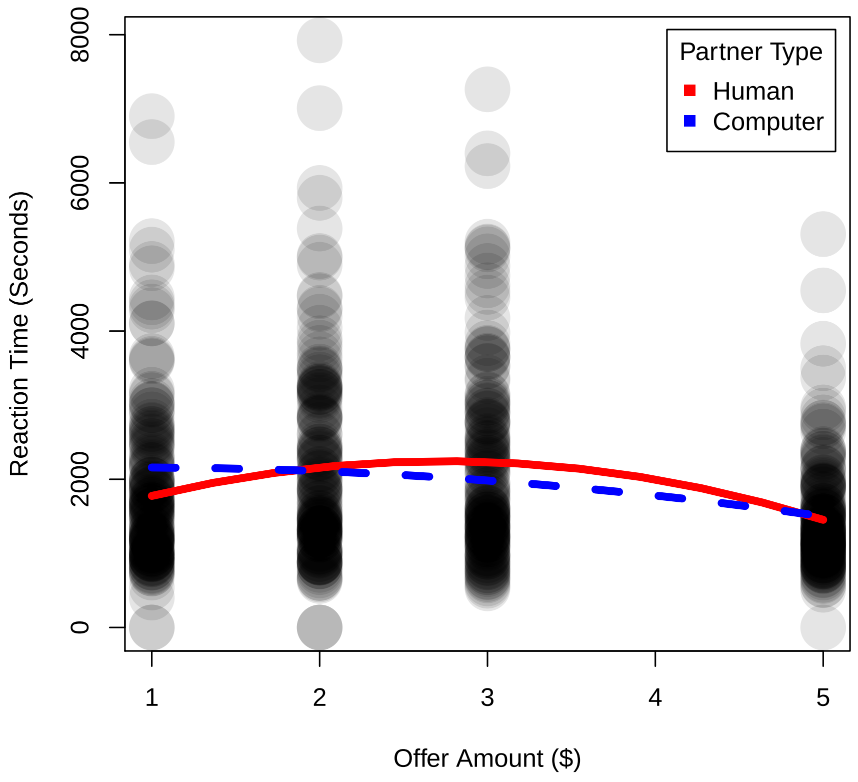

Figure 2: Increased decision conflict for ambiguous offers from humans compared to computers

```
> cpt<-cbind(c$Offer,yc)
> hdat<-data.frame(unique(hpt))[order(data.frame(unique(hpt))[1]),]
> cdat<-data.frame(unique(cpt))[order(data.frame(unique(cpt))[1]),]
> plot(RT~Offer,data=data,col=rgb(0,0,0,.1),pch=16,cex=4,
    ylab="Reaction Time (Seconds)",xlab="Offer Amount ($)")
> lines(spline(hdat),col="red",lwd=5)
> lines(spline(cdat),col="blue",lwd=5,lty=2)
> legend("topright", c("Human","Computer"), pch=15, col=c("red","blue"),
    title="Partner Type", inset = .02)
```

2.4 Model comparisons

While the last model supports our hypothesis, it is important to test whether the model provides the best account of our data given that it is also the most complex model with a total of six free parameters. To compare nested models we can use the `anova` function.

Comparing a series of models that incrementally increase in complexity is essentially how you would conduct a hierarchical or step-wise regression analysis. The significance of the F value determines if the additional level explains a significant amount of variance above and beyond the previous level.

```
> anova(m1,m2,m3,m4)
```

```
Analysis of Variance Table

Model 1: RT ~ Offer
Model 2: RT ~ Offer + I(Offer^2)
Model 3: RT ~ Offer * Condition + I(Offer^2)
Model 4: RT ~ Offer * Condition + I(Offer^2) * Condition
  Res.Df       RSS Df Sum of Sq       F    Pr(>F)
1    646 883590681
2    645 856168411  1  27422270 20.7490 6.267e-06 ***
3    643 854579582  2   1588829  0.6011   0.54852
4    642 848477290  1   6102291  4.6173   0.03202 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We can see from our example that model 2, in which we added a quadratic term for the amount of money offered fits the data significantly better than model 1. Model 3 does not fit better than model 2, indicating that adding the interaction between amount of money offered and condition does not improve the model fit. Finally, we see that adding the interaction between the quadratic amount of money offered with the type of partner in model 4 provides the best account of the data.

2.5 Model Assumptions

Before drawing any grand conclusions from our modeling results, it is important to test that all of the assumptions of the regression analysis were met. Regression diagnostics will be covered in a later chapter. Some of you may have already noticed this, but there is a very large violation of the statistical independence assumption. We have used 36 trials from each subject, but have treated them as independent subjects, which has dramatically boosted our power, but also changed the variance structure of our dataset. In a later chapter we will continue with this example to demonstrate how this can be framed as a mixed model problem and easily addressed using a more sophisticated stats package.

3 Robust Regression

Ordinary least squares regression has several underlying assumptions such as homoscedasticity and homogeneity of variance. Violations of these assumptions can lead to spurious estimates. While it is often argued that OLS is robust to outliers, it is typically meant in terms of Type-I error, not Type-II error. Extreme outliers will dramatically decrease the likelihood of observing a significant effect. Often these extreme values are removed from data analysis. An alternative approach is to use robust regression. Robust regression adopts a different approach to estimation that uses iteratively re-weighted least squares and is robust to both heterscedasticity and extreme outliers. Basically, robust regression will reduce the weight given to data points that have a strong influence on the parameters.

First we need to create some fake data. To demonstrate the effect of outliers on the model estimations, we will add two outliers.

```
> n <- 20
> x <- seq(1, n)
> a.true <- 3
> b.true <- 1.5
> y.true <- a.true + b.true * x
> s.true <- 20
> y <- y.true + s.true * rnorm(n,sd=.15)
> y[n]<-0
> y[n-2]<-1
```

Next we need to load the robust library and fit the two models using `lm` and `lmrob`, the latter of which uses an MM-estimator.

```
> library(robust)
```

```
pcaPP 0.1-1 loaded
Scalable Robust Estimators with High Breakdown Point (version 1.0-00)
```

```
> fm1<- lm(y ~ x)
> fm2<- lmrob(y ~ x)
```

Finally, we will examine differences in the model fits and plot the estimated effects to highlight the influences of the outliers.

```
> summary(fm1)
```

```
Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max 
-22.187  -2.645   1.423   6.608  11.867 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)  
(Intercept)   7.7960     4.1780   1.866   0.0784 .
x             0.7196     0.3488   2.063   0.0538 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 8.994 on 18 degrees of freedom
Multiple R-squared: 0.1912,	Adjusted R-squared: 0.1463 
F-statistic: 4.256 on 1 and 18 DF,  p-value: 0.05383 
```

```
> summary(fm2)
```

```
Call:
lmrob(formula = y ~ x)

Weighted Residuals:
     Min       1Q   Median       3Q      Max 
-32.7021  -2.9435  -0.7096   1.4181   5.6032 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.5820     1.6537   1.561    0.136
```

```
x             1.5060     0.1619   9.304 2.68e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Robust residual standard error: 3.784
Convergence in 9 IRWLS iterations

Robustness weights:
 2 observations c(18,20) are outliers with |weight| = 0 ( < 0.005);
 The remaining 18 ones are summarized as
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.8102  0.9114  0.9866  0.9454  0.9941  0.9980
Algorithmic parameters:
tuning.chi         bb tuning.psi refine.tol    rel.tol
 1.5476400  0.5000000  4.6850610  0.0000001  0.0000001
 nResample     max.it     groups    n.group   best.r.s   k.fast.s      k.max
       500         50          5        400          2          1        200
 trace.lev compute.rd
         0          0
seed : int(0)
```

Here we can see that the two outliers dramatically influence both the parameter estimated for x and it's standard error. The robust regression estimate is more accurate because it down weights the influence of the two outliers. This is clearly depicted in the regression lines in Figure 3. The `lmrob` output also informs you that it has identified two outliers and is unsure of 4 data points, so it weights the remaining 14 data points the most strongly.

```
> plot(y~x,pch=16, col="black",cex=2)
> abline(fm1,col="black",lwd=4)
> abline(fm2,col="red",lty=2,lwd=4)
> legend("topleft", c("OLS","MM-Estimator"), pch=15, col=c("black","red"),
    title="Estimation", inset = .02)
```

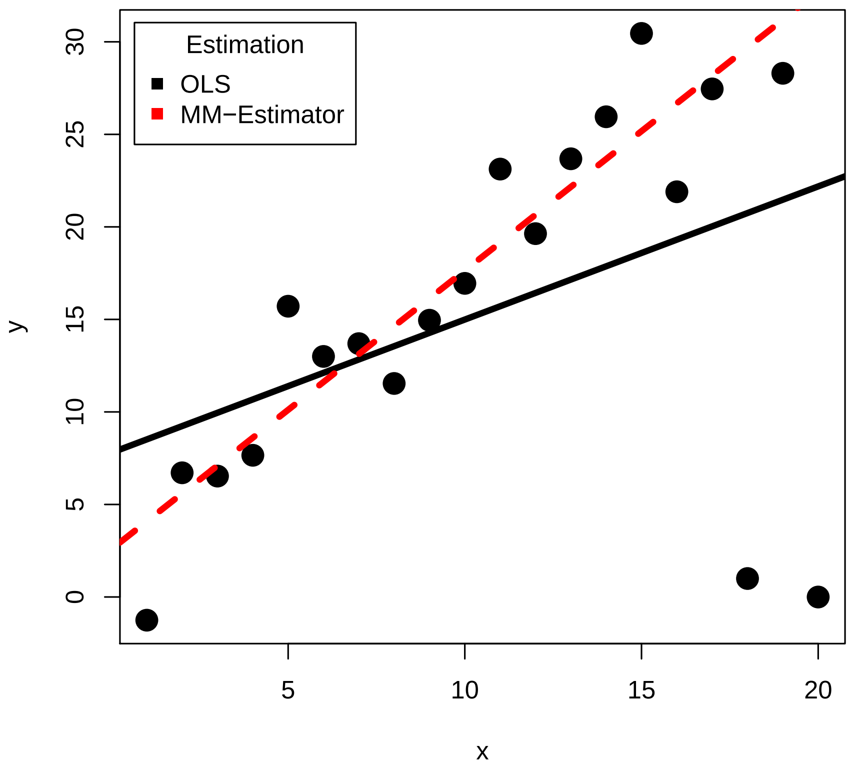

Figure 3: Influence of outliers on OLS and robust regression