

Multiple linear regression

Trevor Hastie and Robert Tibshirani

Here, I am adapting the lab associated with Chapter 3 of the textbook.

We load the ISLR2 package, which includes the data sets associated with this book.

```
library(ISLR2)
```

The ISLR2 library contains the `Boston` data set, which records `medv` (median house value) for 506 census tracts in Boston. We will seek to predict `medv` using 12 predictors such as `rmvar` (average number of rooms per house), `age` (proportion of owner-occupied units built prior to 1940) and `lstat` (percent of households with low socioeconomic status).

```
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

We should also investigate the variables therein and `attach` so that we don't have to use the `$` notation.

```
names(Boston)
```

```
## [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
## [8] "dis"     "rad"     "tax"     "ptratio" "lstat"   "medv"
```

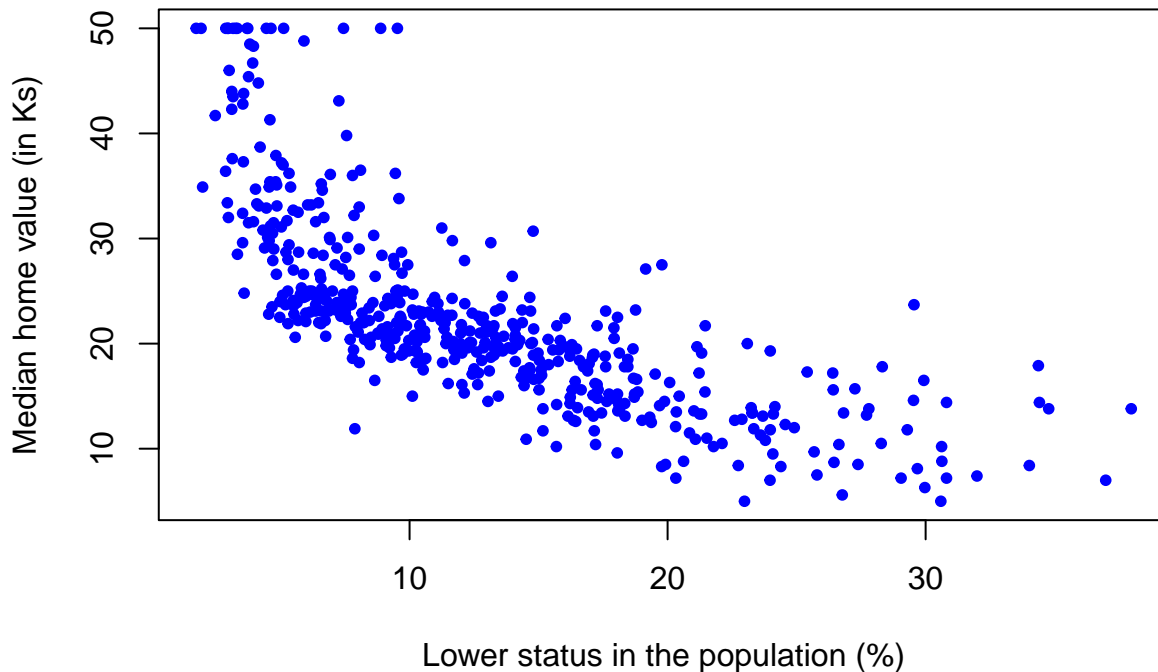
```
attach(Boston)
```

To find out more about the data set, we can type `?Boston`.

We will start by fitting a simple linear regression model, with `medv` as the response and `lstat` as the predictor. Here is the scatterplot.

```
plot(lstat, medv,
     main="Median value vs. lower status",
     xlab="Lower status in the population (%)",
     ylab="Median home value (in Ks)",
     pch=20, col="blue")
```

Median value vs. lower status



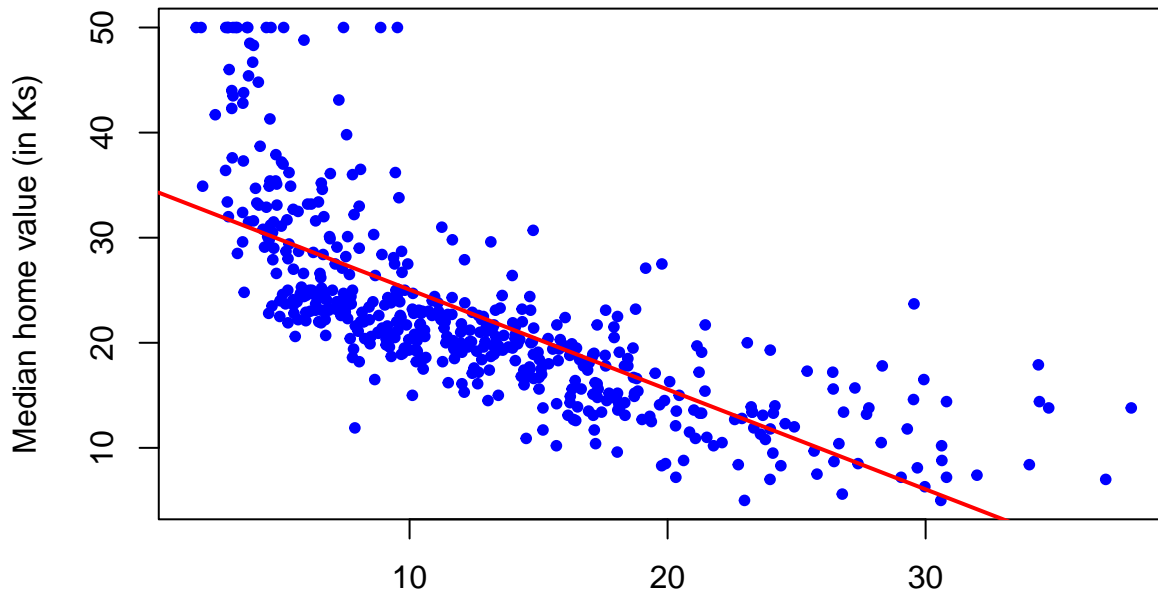
For the implementation of the linear model, the basic syntax is `lm(y ~ x, data)`, where `y` is the response, `x` is the predictor, and `data` is the data set in which these two variables are kept.

```
lm.fit <- lm(medv ~ lstat)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  34.55384    0.56263   61.41  <2e-16 ***
## lstat        -0.95005    0.03873  -24.53  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF, p-value: < 2.2e-16
```

```
plot(lstat, medv,
     main="Median value vs. lower status",
     xlab="Lower status in the population (%)",
     ylab="Median home value (in Ks)",
     pch=20, col="blue")
abline(lm.fit, col="red", lwd=2)
```

Median value vs. lower status



Lower status in the population (%)

We can

use the `names()` function in order to find out what other pieces of information are stored in `lm.fit`. Although we can extract these quantities by name—e.g. `lm.fit$coefficients`—it is safer to use the extractor functions like `coef()` to access them.

```
names(lm.fit)
```

```
## [1] "coefficients" "residuals"      "effects"        "rank"
## [5] "fitted.values" "assign"         "qr"            "df.residual"
## [9] "xlevels"      "call"          "terms"         "model"
```

```
coef(lm.fit)
```

```
## (Intercept)      lstat
## 34.5538409   -0.9500494
```

In order to obtain a confidence interval for the coefficient estimates, we can use the `confint()` command.

```
confint(lm.fit)
```

```
##              2.5 %      97.5 %
## (Intercept) 33.448457 35.6592247
## lstat      -1.026148 -0.8739505
```

The `predict()` function can be used to produce confidence intervals and prediction intervals for the prediction of `medv` for a given value of `lstat`.

```
predict(lm.fit, data.frame(lstat = (c(5, 10, 15))),
        interval = "confidence")
```

```
##      fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

```
predict(lm.fit, data.frame(lstat = (c(5, 10, 15))),
       interval = "prediction")
```

```
##          fit          lwr          upr
## 1 29.80359 17.565675 42.04151
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

How about the full array of confidence and prediction intervals?

```
newdata<-data.frame(lstat=seq(min(lstat),max(lstat),0.1))
```

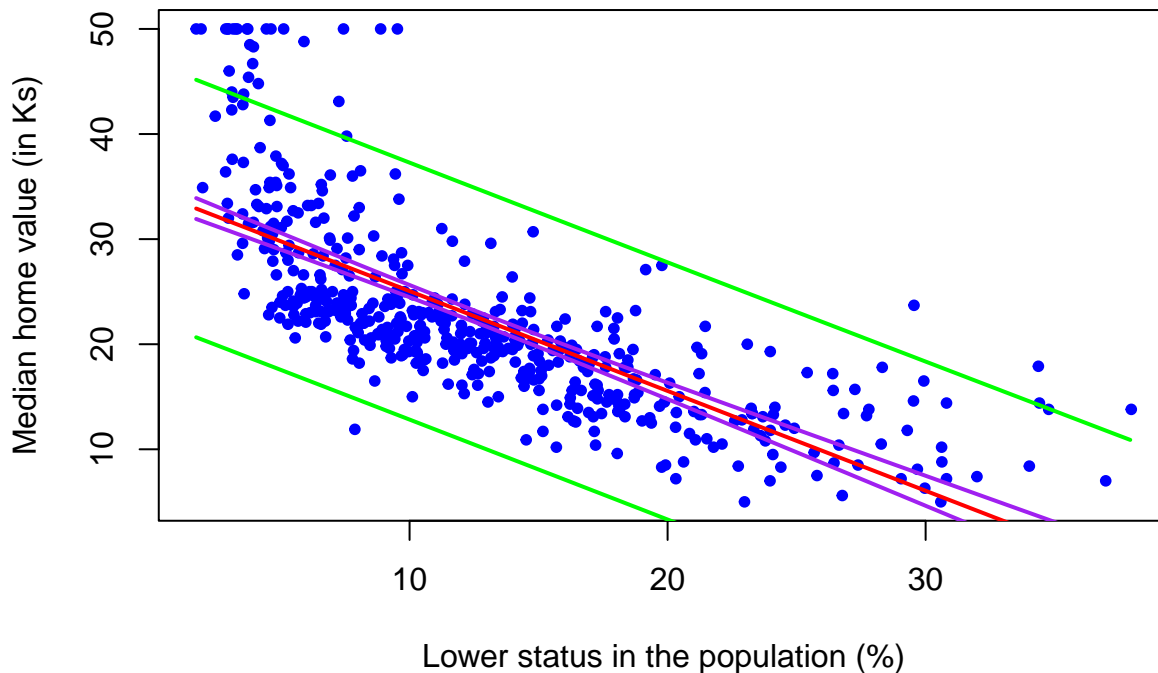
```
conf<-predict(lm.fit,newdata,interval="confidence")
pred<-predict(lm.fit,newdata,interval="prediction")
all.ints<-cbind(conf,pred[,1])
```

```
plot(lstat, medv,
```

```
     main="Median value vs. lower status",
     xlab="Lower status in the population (%)",
     ylab="Median home value (in Ks)",
     pch=20, col="blue")
```

```
matplot(newdata, all.ints, type="l", lty=1, lwd=2, col=c("red", "purple", "purple","green","green"), ad
```

Median value vs. lower status

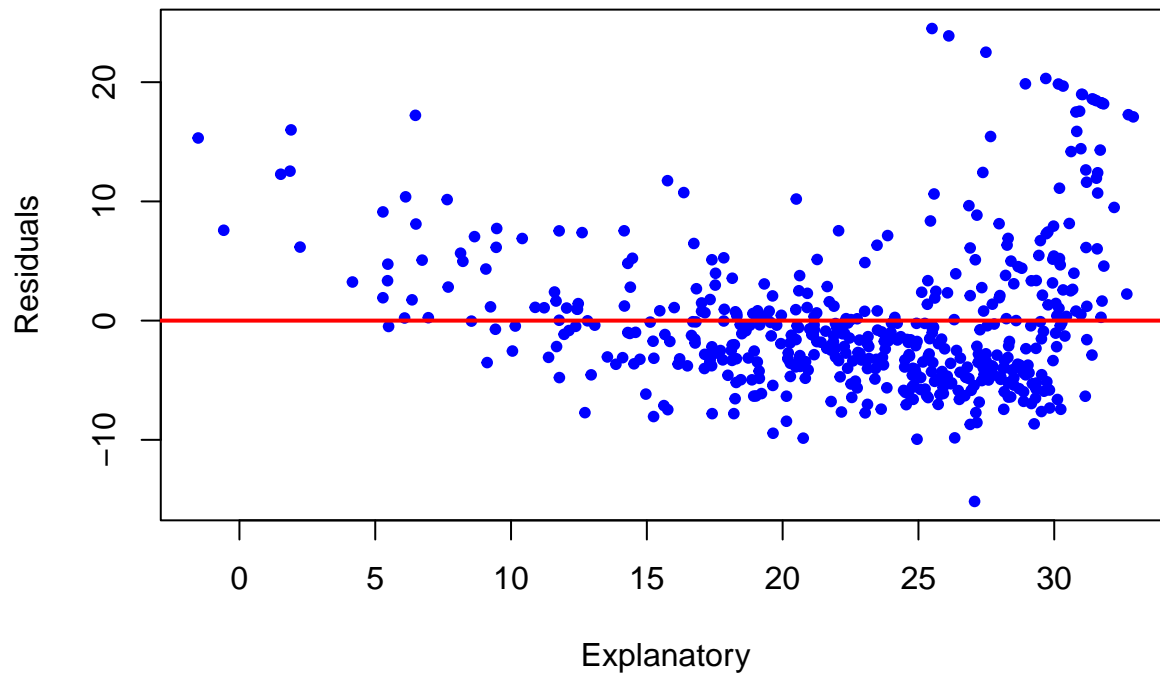


We can compute the residuals from a linear regression fit using the `residuals()` function. The function `rstudent()` will return the studentized residuals, and we can use this function to plot the residuals against the fitted values.

```
plot(predict(lm.fit), residuals(lm.fit),
     main="Do residuals depend on the explanatory?",
     xlab="Explanatory",
     ylab="Residuals",
```

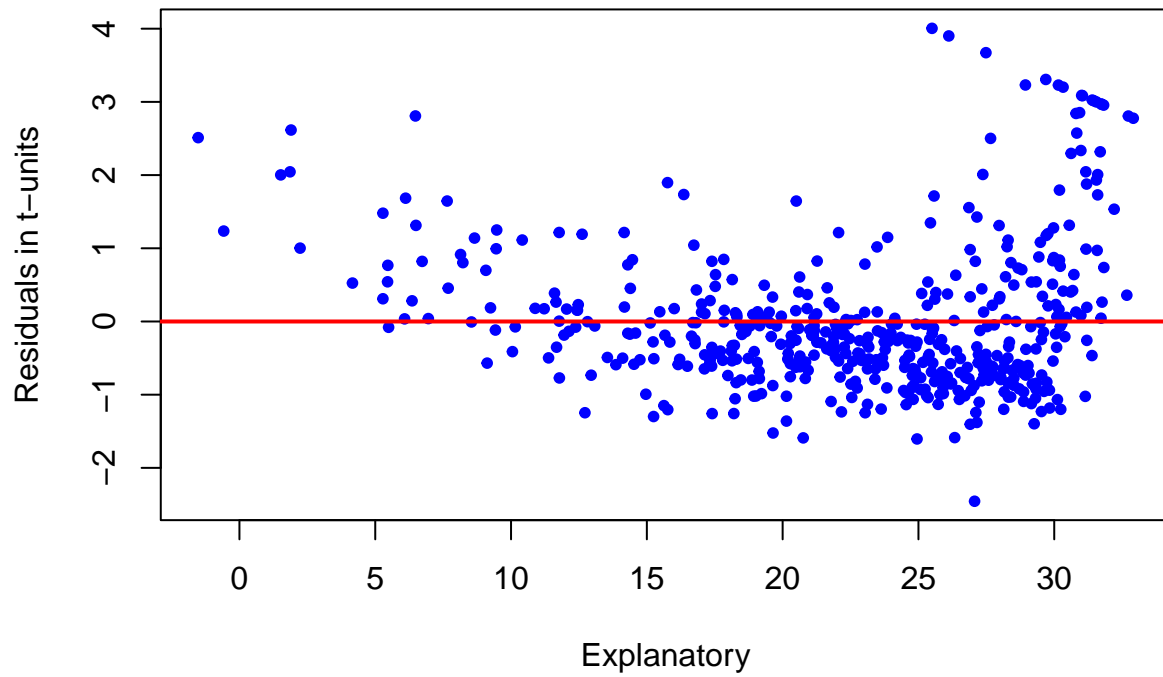
```
pch=20, col="blue")
abline(0,0, col="red", lwd=2)
```

Do residuals depend on the explanatory?



```
plot(predict(lm.fit), rstudent(lm.fit),
      main="Do residuals depend on the explanatory?",
      xlab="Explanatory",
      ylab="Residuals in t-units",
      pch=20, col="blue")
abline(0,0, col="red", lwd=2)
```

Do residuals depend on the explanatory?



On the basis of the residual plots, there is some evidence of non-linearity.