Tutorial 4: Linear Methods for Regression and Classification

Introduction to Statistical Learning with Applications in R

February 2, 2018

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

Simple Linear Regression

In this example, we are going to use the dataset `state.x77` that comes with standard `R` installation. It is a data set about the 50 states of united states.

Type `help(state.x77)` in your console window to read more about this data set.

```
library(datasets)
statedata=as.data.frame(state.x77)
```

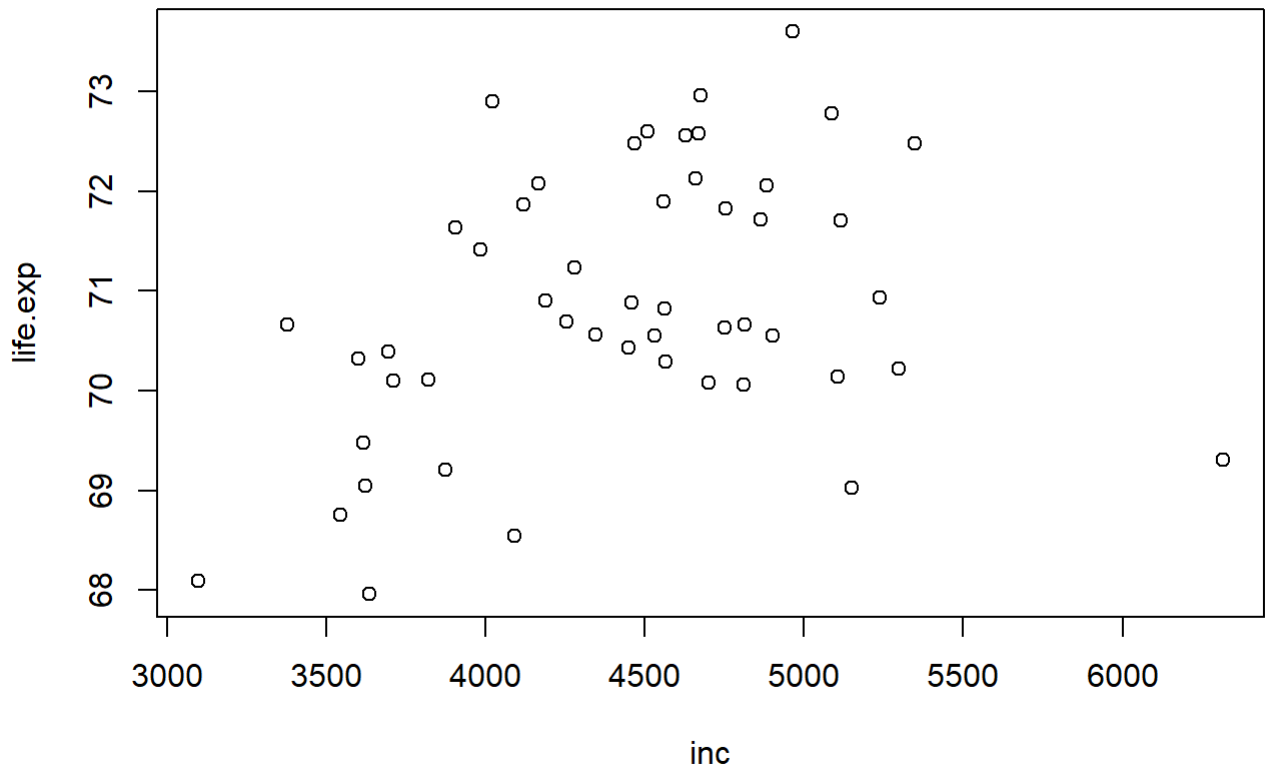
First we can modify the column names (the variable names) for easier use. The function `colnames` can be used to retrieve the colume names or assign new names.

```
colnames(statedata)=c("popu", "inc", "illit", "life.exp", "murder", "hs.grad", "frost", "area")
```

Make a scatterplot.

For this example, let's look at the association between life expectancy (`life.exp`) and income (`inc`). The scatterplot below shows a positive association between these two variables.

```
plot(life.exp~inc, data=statedata)
```



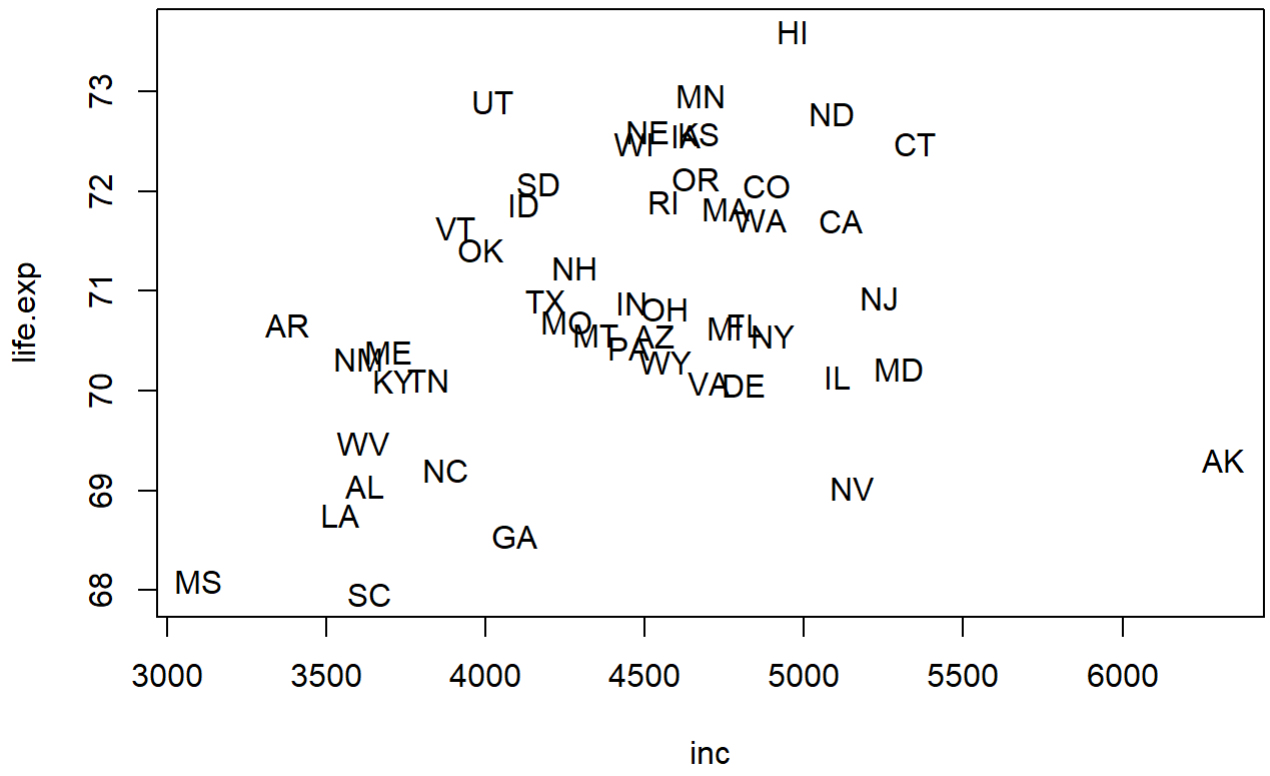
We can compute the correlation between these two variables. The value of the correlation indicates a weak positive linear association.

```
cor(statedata[,"life.exp"], statedata[,"inc"])
```

```
## [1] 0.3402553
```

There is one observation that is far away from the rest of the points. We would like to know which state is corresponding to that point. Let's add state abbreviations to the plot.

```
plot(life.exp~inc, data=statedata, type="n")
text(life.exp~inc, data=statedata, state.abb)
```



Fit a simple linear regression

Now let's fit the following linear regression model between Y (`life.exp`) and X (`inc`)

$$Y = \beta_0 + \beta_1 X + \varepsilon.$$

Here β_0 and β_1 are the regression coefficients of the model and ε represents independent random errors.

Fitting a linear regression is to derive

$$\hat{Y} = b_0 + b_1 X.$$

Here $\hat{Y}$ is a fitted prediction for the observed life expectancies. The difference $Y - \hat{Y}$ is the prediction error, or residual.

b_0 and b_1 are estimates for the regression coefficients. They are identified via the least square regression method that minimizes

$$\sum_{i=1}^{n} (Y - \hat{Y})^2,$$

i.e., the sum of squared prediction errors.

```
model1=lm(life.exp~inc, data=statedata)
summary(model1)
```

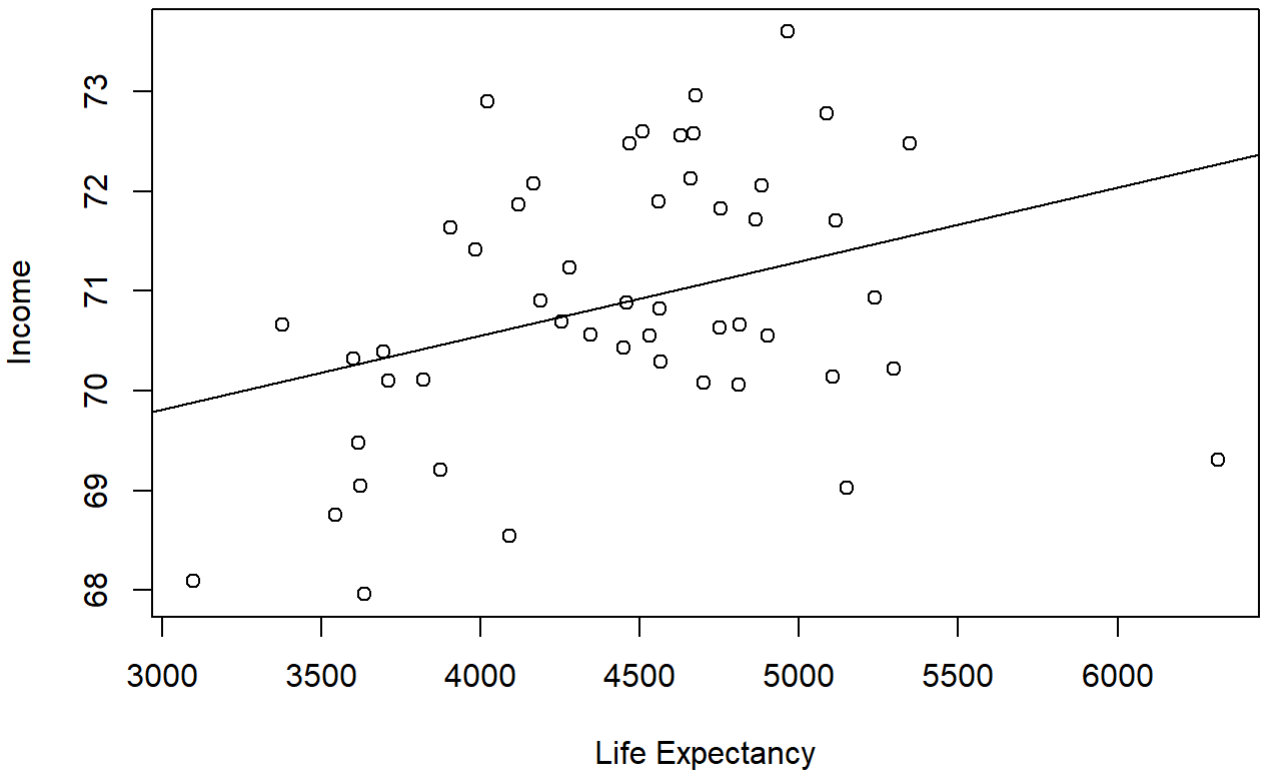
```
##
## Call:
## lm(formula = life.exp ~ inc, data = statedata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.96547 -0.76381 -0.03428  0.92876  2.32951
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.758e+01  1.328e+00  50.906   <2e-16 ***
## inc         7.433e-04  2.965e-04   2.507   0.0156 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.275 on 48 degrees of freedom
## Multiple R-squared:  0.1158, Adjusted R-squared:  0.09735
## F-statistic: 6.285 on 1 and 48 DF,  p-value: 0.01562
```

In the above output, the intercept is b_0 and the coeffient under the column (`inc`) is b_1–the slope. The slope is estimated to be 7.4×10^{-4}. The maganitude of this value does not mean that the effect of income on life expectance is very small. This maganitude is decided by the maganitude of the X variable and Y variable.

Sampling uncertainty in regression analysis

Consider a population of 50 states and we identify the true regression line in this population. Here the function `abline` add a straight line to an existing plot.

```
plot(life.exp~inc, data=statedata,
      xlab="Life Expectancy", ylab="Income")
abline(model1)
```



Now we can consider 4 random samples. We use a `for` loop (https://en.wikipedia.org/wiki/For_loop) to run an identical regression analysis on 4 randomly selected samples.

Within the loop, we will implement the following steps for each repetition.

- Step 1: randomly select 10 states using the `sample` function of `R`.
- Step 2: run least square regression on the selected states only
- Step 3: compute a 95% confidence band for the true regression line in the population using the sample.

[Note] The states are randomly selected in the following. Therefore, every time you run this file, you will produce different random samples that will give different estimated least regression lines.

```
par(mfrow=c(2,2)) # create a panel of four plotting areas

for(i in 1:4){
  ## Plot the population
  plot(life.exp~inc, data=statedata,
       xlab="Life Expectancy", ylab="Income",
       title=paste("Random sample", format(i)),
       ylim=c(min(life.exp), max(life.exp)+0.3))
  abline(model1)
  if(i==1){
    legend(3030, 74.2,
           pch=c(NA, NA, NA, 1, 16),
           lty=c(1, 1, 2, NA, NA),
           col=c(1, 2, 2, 1, 2),
           c("population truth", "sample estimate",
             "sample confidence band",
             "states", "sampled"),
           cex=0.7,
           bty="n"
           )
  }
  ## Select the sample
  selected.states=sample(1:50, 10)
  points(statedata[selected.states,"inc"],
         statedata[selected.states,"life.exp"], pch=16, col=2)
  ## Fit a regression line using the sample
  model.sel = lm(life.exp~inc, data=statedata[selected.states,])
  abline(model.sel, col=2)
  ## Make a confidence band.
  #### first calculate the width of the band, W.
  ww=qt(0.975, 10-2)
  #### generate plotting X values.
  plot.x<-data.frame(inc=seq(3000, 7000, 1))
  #### se.fit=T is an option to save
  #### the standard error of the fitted values.
  plot.fit<-predict(model.sel, plot.x,
                    level=0.95, interval="confidence",
                    se.fit=T)

  #### lines is a function to add connected lines
  #### to an existing plot.
  lines(plot.x$inc, plot.fit$fit[,1]+ww*plot.fit$se.fit,
        col=2, lty=2)
  lines(plot.x$inc, plot.fit$fit[,1]-ww*plot.fit$se.fit,
        col=2, lty=2)
}
```

```
## Warning in plot.window(...): "title" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter
```

```
## Warning in box(...): "title" is not a graphical parameter
```

```
## Warning in title(...): "title" is not a graphical parameter
```

```
## Warning in plot.window(...): "title" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter
```

```
## Warning in box(...): "title" is not a graphical parameter
```

```
## Warning in title(...): "title" is not a graphical parameter
```

```
## Warning in plot.window(...): "title" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter
```

```
## Warning in box(...): "title" is not a graphical parameter
```

```
## Warning in title(...): "title" is not a graphical parameter
```

```
## Warning in plot.window(...): "title" is not a graphical parameter
```

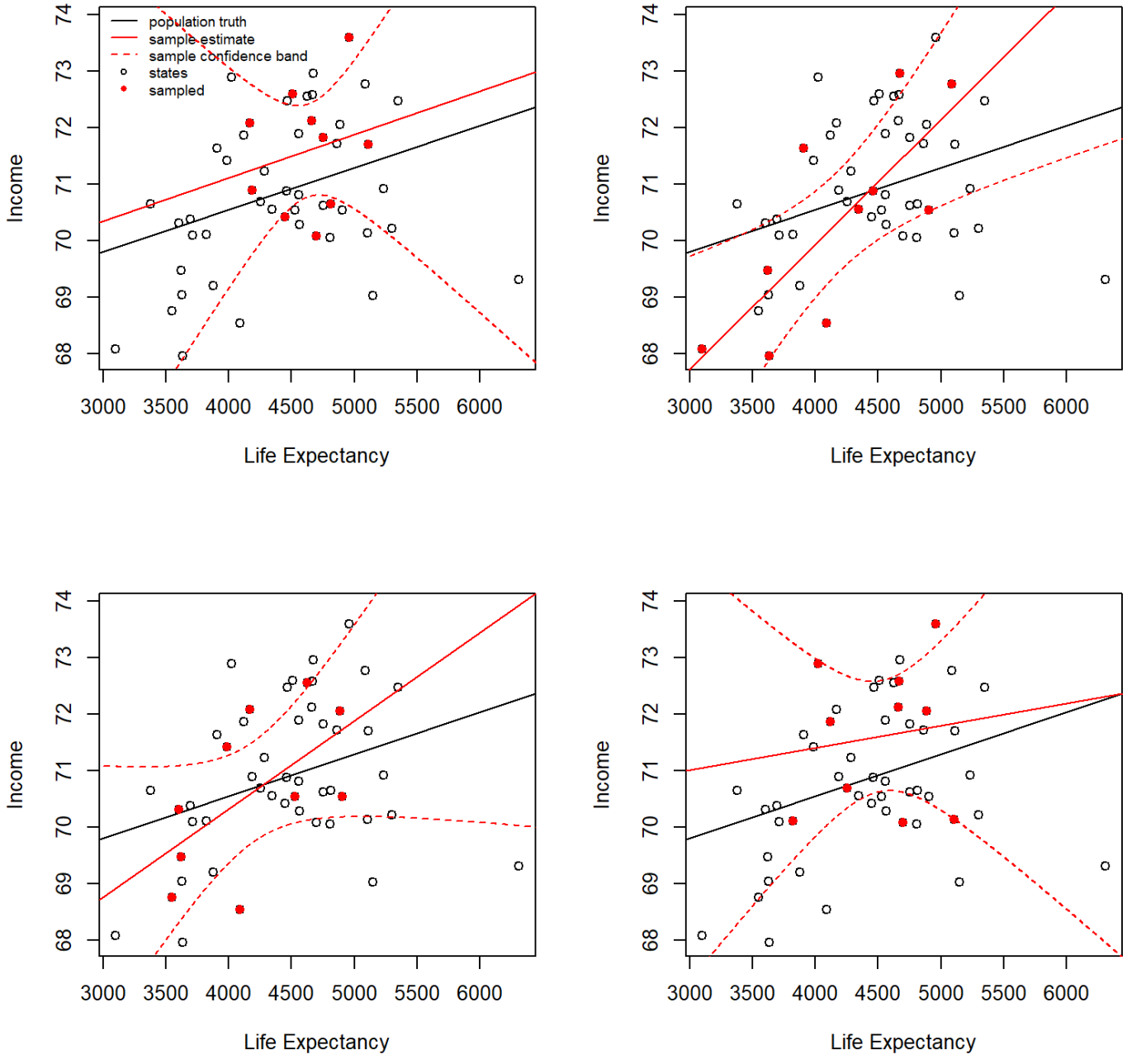
```
## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter
```

```
## Warning in box(...): "title" is not a graphical parameter
```

```
## Warning in title(...): "title" is not a graphical parameter
```



Multiple Linear Regression

The `MASS` library contains the `Boston` data set, which records `medv` (median house value) for 506 neighborhoods around Boston. We will seek to predict `medv` using 13 predictors such as `rm` (average number of rooms per house), `age` (average age of houses), and `lstat` (percent of households with low socioeconomic status).

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.4.3
```

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.4.2
```

In order to fit a multiple linear regression model using least squares, we again use the `lm()` function. The syntax `lm(yâ□¼x1+x2+x3)` is used to fit a model with three predictors, `x1` , `x2` , and `x3` . The `summary()` function now outputs the regression coefficients for all the predictors.

```
attach(Boston)
lm.fit=lm(medv ~ lstat+age,data=Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

The Boston data set contains 13 variables, and so it would be cumbersome to have to type all of these in order to perform a regression using all of the predictors. Instead, we can use the following short-hand:

```
lm.fit=lm(medv~.,data=Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

What if we would like to perform a regression using all of the variables but one? For example, in the above regression output, `age` has a high p-value. So we may wish to run a regression excluding this predictor. The following syntax results in a regression using all predictors except `age`.

```
lm.fit1=lm(medv ~.-age,data=Boston)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.436927   5.080119   7.172 2.72e-12 ***
## crim         -0.108006   0.032832  -3.290 0.001075 **
## zn            0.046334   0.013613   3.404 0.000719 ***
## indus         0.020562   0.061433   0.335 0.737989
## chas          2.689026   0.859598   3.128 0.001863 **
## nox         -17.713540   3.679308  -4.814 1.97e-06 ***
## rm            3.814394   0.408480   9.338  < 2e-16 ***
## dis          -1.478612   0.190611  -7.757 5.03e-14 ***
## rad           0.305786   0.066089   4.627 4.75e-06 ***
## tax          -0.012329   0.003755  -3.283 0.001099 **
## ptratio      -0.952211   0.130294  -7.308 1.10e-12 ***
## black         0.009321   0.002678   3.481 0.000544 ***
## lstat        -0.523852   0.047625 -10.999  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.74 on 493 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7343
## F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

Alternatively, the `update()` function can be used.

```
lm.fit1=lm(medv ~ .-age,data=Boston)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.436927   5.080119   7.172 2.72e-12 ***
## crim         -0.108006   0.032832  -3.290 0.001075 **
## zn            0.046334   0.013613   3.404 0.000719 ***
## indus         0.020562   0.061433   0.335 0.737989
## chas          2.689026   0.859598   3.128 0.001863 **
## nox         -17.713540   3.679308  -4.814 1.97e-06 ***
## rm            3.814394   0.408480   9.338  < 2e-16 ***
## dis          -1.478612   0.190611  -7.757 5.03e-14 ***
## rad           0.305786   0.066089   4.627 4.75e-06 ***
## tax          -0.012329   0.003755  -3.283 0.001099 **
## ptratio      -0.952211   0.130294  -7.308 1.10e-12 ***
## black         0.009321   0.002678   3.481 0.000544 ***
## lstat        -0.523852   0.047625 -10.999  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.74 on 493 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7343
## F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

Interaction Terms

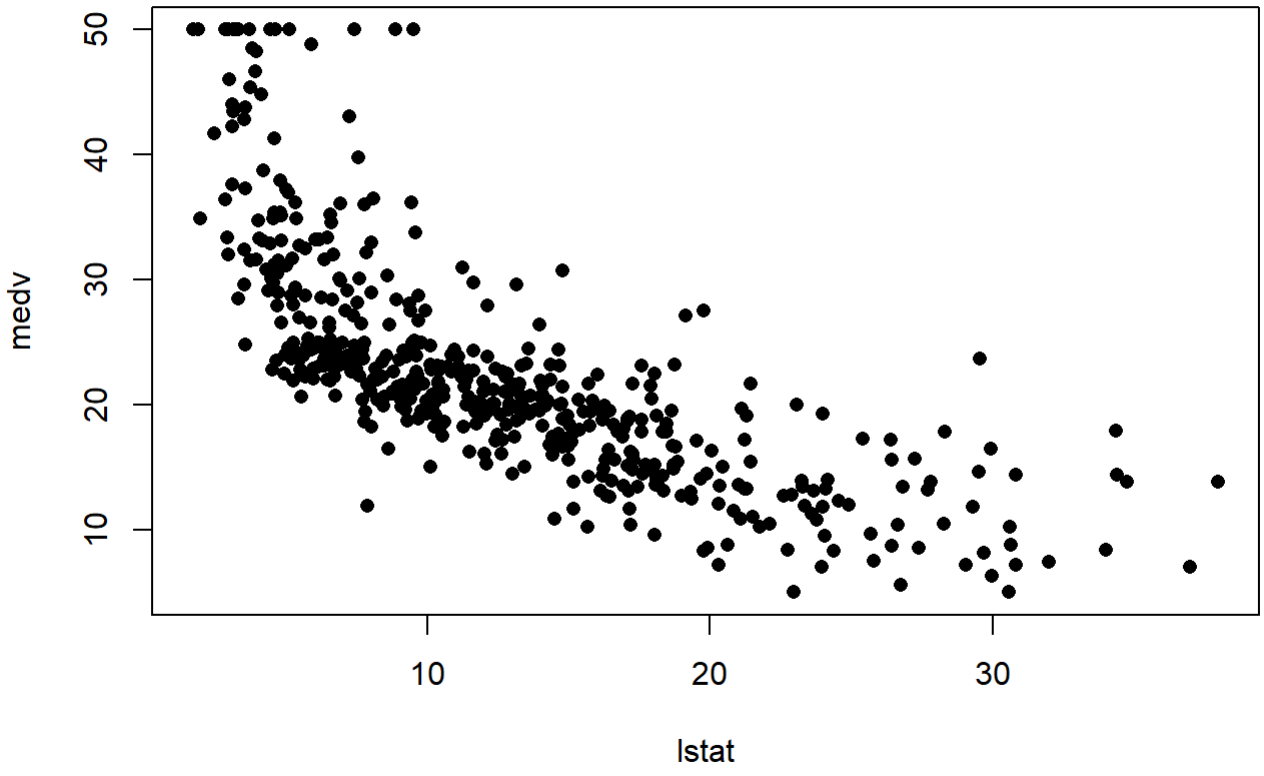
It is easy to include interaction terms in a linear model using the `lm()` function. The syntax `lstat:black` tells `R` to include an interaction term between `lstat` and `black`. The syntax `lstat*age` simultaneously includes `lstat`, `age`, and the interaction term lstatÃ□age as predictors; it is a shorthand for `lstat+age+lstat:age`.

```
summary(lm(medv ~ lstat*age,data=Boston))
```

```
##
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.806  -4.045  -1.333   2.085  27.552
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
## lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
## age         -0.0007209  0.0198792  -0.036   0.9711
## lstat:age    0.0041560  0.0018518   2.244   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.149 on 502 degrees of freedom
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5531
## F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

Non-linear Transformations of the Predictors

```
plot(lstat, medv, pch=16)
```



The `lm()` function can also accommodate non-linear transformations of the predictors. For instance, given a predictor X, we can create a predictor X^2 using `I(X^2)` . The function `I()` is needed since the `^` has a special meaning in a formula; wrapping as we do allows the standard usage in `R` , which is `I()` to raise `X` to the power 2. We now perform a regression of medv onto `lstat` and $'lstat'^2$.

```
lm.fit2=lm(medv ~ lstat+I(lstat^2))
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## lstat       -2.332821   0.123803  -18.84   <2e-16 ***
## I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

The near-zero p-value associated with the quadratic term suggests that it leads to an improved model. We use the `anova()` function to further quantify the extent to which the quadratic fit is superior to the linear fit.

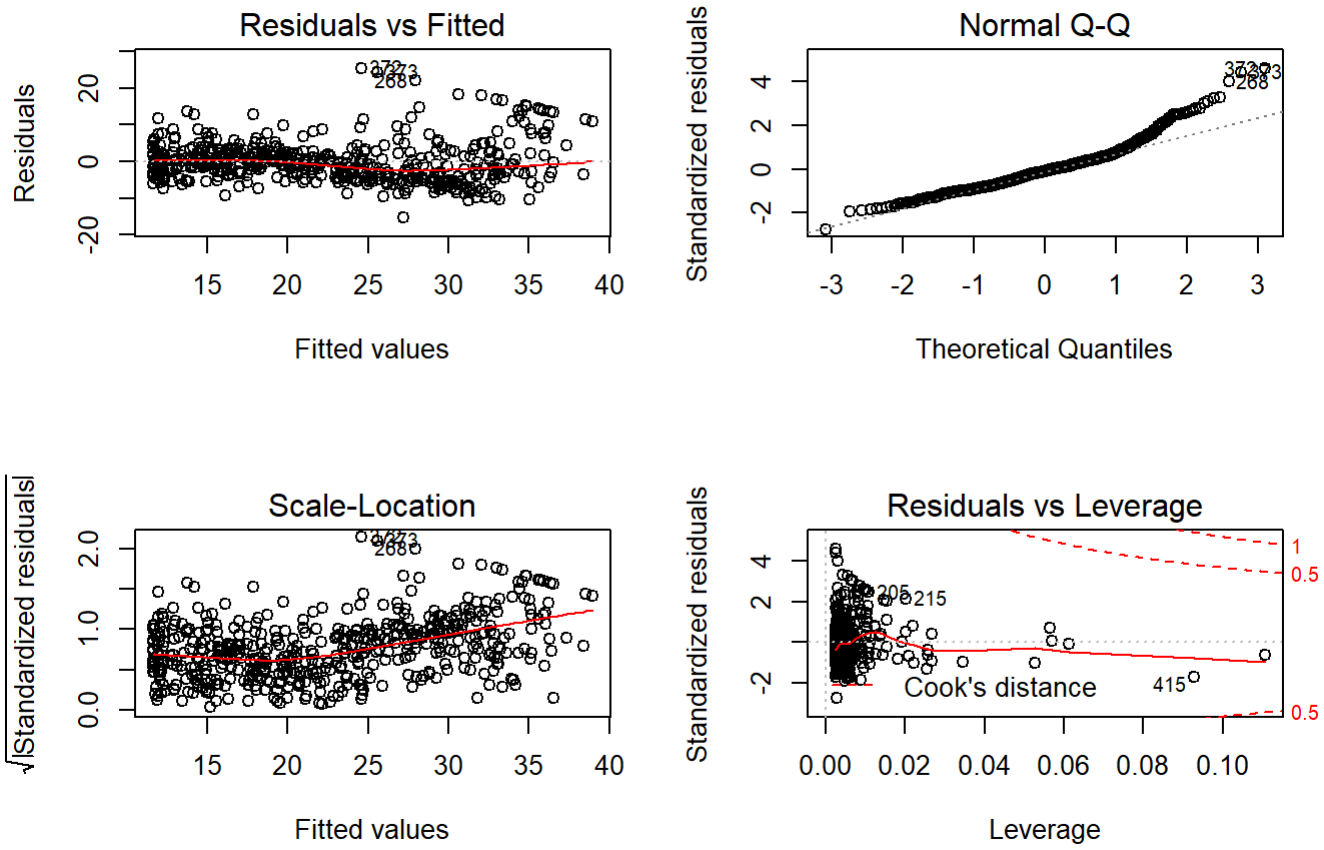
```
lm.fit=lm(medv~lstat)
anova(lm.fit ,lm.fit2)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ lstat
## Model 2: medv ~ lstat + I(lstat^2)
##   Res.Df   RSS Df Sum of Sq     F    Pr(>F)
## 1    504 19472
## 2    503 15347  1    4125.1 135.2 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Here Model 1 represents the linear submodel containing only one predictor, `lstat` , while Model 2 corresponds to the larger quadratic model that has two predictors, `lstat` and $\texttt{lstat}^2$. The `anova()` function performs a hypothesis test comparing the two models. The null hypothesis is that the two models fit the data equally well, and the alternative hypothesis is that the full model is superior. Here the F-statistic is 135 and the associated p-value is

virtually zero. This provides very clear evidence that the model containing the predictors `lstat` and $\texttt{lstat}^2$ is far superior to the model that only contains the predictor lstat. This is not surprising, since earlier we saw evidence for non-linearity in the relationship between `medv` and `lstat`. If we type

```
par(mfrow=c(2,2))
plot(lm.fit2)
```



then we see that when the $\texttt{lstat}^2$ term is included in the model, there is little discernible pattern in the residuals.

Qualitative Predictors

We will now examine the `Carseats` data, which is part of the `ISLR` library. We will attempt to predict `Sales` (child car seat sales) in 400 locations based on a number of predictors.

```
names(Carseats)
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertising" "Population"
##  [6] "Price"       "ShelveLoc"   "Age"         "Education"   "Urban"
## [11] "US"
```

The `Carseats` data includes qualitative predictors such as `Shelveloc`, an in- dicator of the quality of the shelving locationâ□□that is, the space within a store in which the car seat is displayedâ□□at each location. The predictor `Shelveloc` takes on three possible values, *Bad*, *Medium*, and *Good*.

Given a qualitative variable such as `Shelveloc`, `R` generates dummy variables automatically. Below we fit a multiple regression model that includes some interaction terms.

```
lm.fit=lm(Sales~.+Income:Advertising+Price:Age,data=Carseats)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Sales ~ . + Income:Advertising + Price:Age, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9208 -0.7503  0.0177  0.6754  3.3413
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         6.5755654  1.0087470   6.519 2.22e-10 ***
## CompPrice           0.0929371  0.0041183  22.567  < 2e-16 ***
## Income              0.0108940  0.0026044   4.183 3.57e-05 ***
## Advertising         0.0702462  0.0226091   3.107 0.002030 **
## Population          0.0001592  0.0003679   0.433 0.665330
## Price              -0.1008064  0.0074399 -13.549  < 2e-16 ***
## ShelveLocGood       4.8486762  0.1528378  31.724  < 2e-16 ***
## ShelveLocMedium     1.9532620  0.1257682  15.531  < 2e-16 ***
## Age                -0.0579466  0.0159506  -3.633 0.000318 ***
## Education          -0.0208525  0.0196131  -1.063 0.288361
## UrbanYes            0.1401597  0.1124019   1.247 0.213171
## USYes              -0.1575571  0.1489234  -1.058 0.290729
## Income:Advertising  0.0007510  0.0002784   2.698 0.007290 **
## Price:Age           0.0001068  0.0001333   0.801 0.423812
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.011 on 386 degrees of freedom
## Multiple R-squared:  0.8761, Adjusted R-squared:  0.8719
## F-statistic:   210 on 13 and 386 DF,  p-value: < 2.2e-16
```

The `contrasts()` function returns the coding that `R` uses for the dummy variables.

```
attach(Carseats)
contrasts(ShelveLoc)
```

```
##        Good Medium
## Bad       0      0
## Good      1      0
## Medium    0      1
```

`R` has created a `ShelveLocGood` dummy variable that takes on a value of 1 if the shelving location is good, and 0 otherwise. It has also created a `ShelveLocMedium` dummy variable that equals 1 if the shelving location is medium, and 0 otherwise. A bad shelving location corresponds to a zero for each of the two dummy variables. The fact that the coefficient for `ShelveLocGood` in the regression output is positive indicates that a good shelving location is

associated with high sales (relative to a bad location). And `ShelveLocMedium` has a smaller positive coefficient, indicating that a medium shelving location leads to higher sales than a bad shelving location but lower sales than a good shelving location.

Logistic Regression

This part, we focus on a classification problem. We will apply logistic regression to the iris data. First, we download the data

```
iris = read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data", sep = ",", he
ader = FALSE)
names(iris) = c("sepal.length", "sepal.width", "petal.length", "petal.width", "iris.type")
### attach name to each column so that we can directly access each column by its name
attach(iris)
```

We randomly split the data into training set and test set.

```
train = sample.int(nrow(iris), 100)
```

The `glm()` function in `R` can only deal with binary classification problems. Letâ□□s try to distinguish Setosa apart from Virginica, Versicolor.

```
Y = iris.type == "Iris-setosa"
logistic.model = glm(Y ~ sepal.length + sepal.width, data=iris, family = binomial(), subset=train)
```

```
## Warning: glm.fit: algorithm did not converge
```

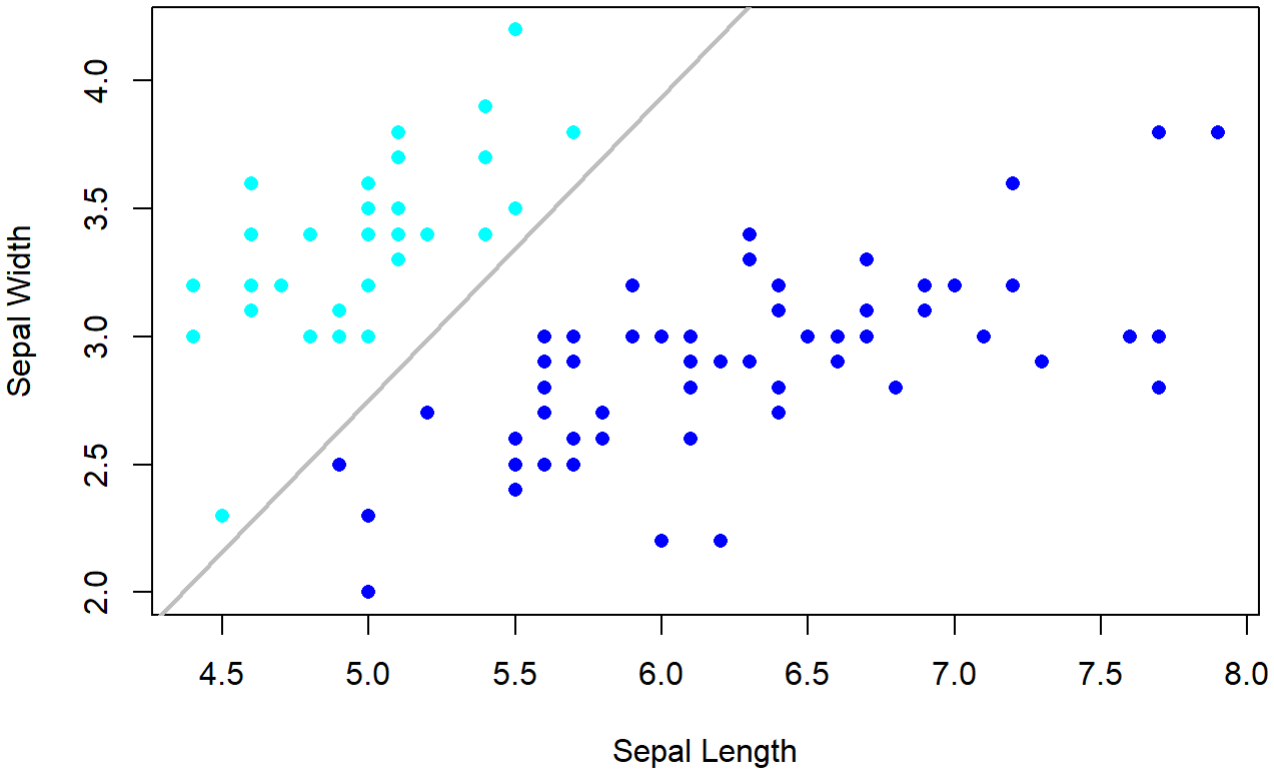
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
logistic.model
```

```
##
## Call:  glm(formula = Y ~ sepal.length + sepal.width, family = binomial(),
##     data = iris, subset = train)
##
## Coefficients:
##  (Intercept)  sepal.length   sepal.width
##        447.6        -167.3         141.2
##
## Degrees of Freedom: 99 Total (i.e. Null);  97 Residual
## Null Deviance:       132.8
## Residual Deviance: 1.719e-08     AIC: 6
```

We can visualize the model in a two-dimensional plot.

```
plot(sepal.length[train], sepal.width[train], type='p',pch=16, col=(Y[train]+4), xlab="Sepal Length", ylab
="Sepal Width")
abline(a = -logistic.model$coefficients[1]/logistic.model$coefficients[3], b = -logistic.model$coefficient
s[2]/logistic.model$coefficients[3], col='gray', lwd=2)
```



To make predictions on the test set, we call the function `predict()`

```
glm.probs = predict(logistic.model, iris[-train,], type="response")
glm.pred = glm.probs>0.5
### summrize the prediction by a confusion matrix
table(Y[-train], glm.pred)
```

```
##         glm.pred
##          FALSE TRUE
##    FALSE    38    0
##    TRUE      0   12
```

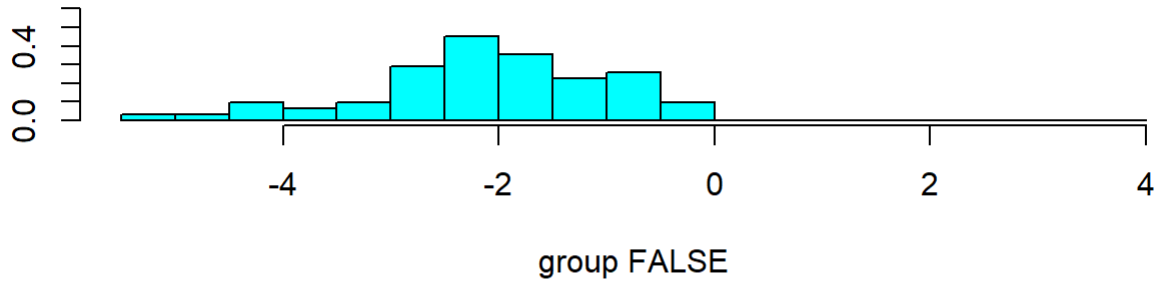
Linear Discriminant Analysis

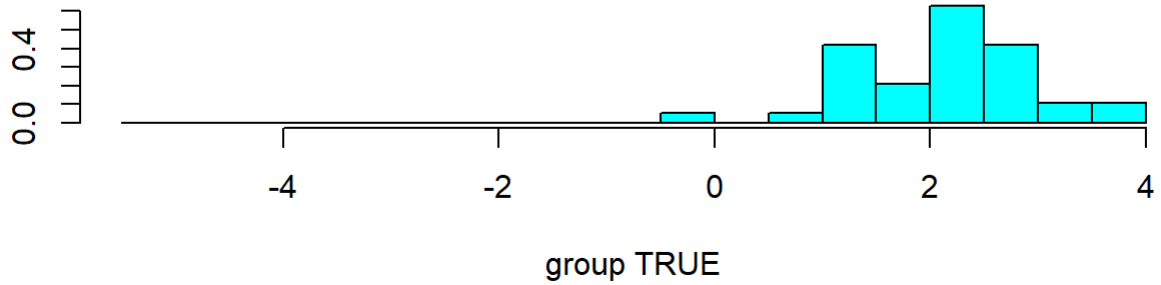
We can also perfom LDA on the same data set. In `R`, we fit a LDA model using the `lda()` function, which is part of the `MASS` library.

```
library(MASS)
lda.model<-lda(Y ~ sepal.length + sepal.width, data=iris, subset=train)
lda.model
```

```
## Call:
## lda(Y ~ sepal.length + sepal.width, data = iris, subset = train)
##
## Prior probabilities of groups:
## FALSE  TRUE
##  0.62  0.38
##
## Group means:
##       sepal.length sepal.width
## FALSE     6.279032    2.890323
## TRUE      4.994737    3.394737
##
## Coefficients of linear discriminants:
##                    LD1
## sepal.length -2.060731
## sepal.width   3.164999
```

```
plot(lda.model)
```





To make predictions, we call the function `predict()`

```
lda.pred = predict(lda.model, iris[-train,])
table(Y[-train], lda.pred$class)
```

```
##
##          FALSE TRUE
##   FALSE     38    0
##   TRUE       0   12
```

For this data set, the LDA and logistic regression predictions are almost identical.