Lab 1: R basics

Required packages:

- MASS (for Boston data)
- sandwich (for heteroskedasticity-robust or "sandwich" standard errors)

Some basic commands

Working directory

Get working directory (note () which is used to distinguish functions):

```
getwd()
```

```
## [1] "/Users/benchiang/Documents/R_Project/ECON6083"
```

Set working directory:

```
?setwd()
```

Or in R Studio, use

- Session->Set Working Directory, or
- Tools->Global Options.

Vectors and matrices

Generate a vector:

```
x<-c(1,2,3)
x
```

```
## [1] 1 2 3
```

```
typeof(x)
```

```
## [1] "double"
```

```
x<-c("No","Yes")
x
```

```
## [1] "No"  "Yes"
```

```
typeof(x)
```

```
## [1] "character"
```

Generate a matrix:

```
X<-matrix(c(1,2,3,4),ncol=2)
X
```

```
##      [,1] [,2]
## [1,]    1    3
```

```
## [2,]    2    4
```

```
typeof(X)
```

```
## [1] "double"
```

```
class(X)
```

```
## [1] "matrix"
```

Add a vector as another column:

```
x=c(5,6)
Y=cbind(X,x)
Y
```

```
##          x
## [1,] 1 3 5
## [2,] 2 4 6
```

Add a vector as another row:

```
y=c(7,8,9)
rbind(Y,y)
```

```
##       x
##   1 3 5
##   2 4 6
## y 7 8 9
```

Random matrix: generate eight independent $N(0,1)$ random variables arranged in 4 columns:

```
v=rnorm(8)
v
```

```
## [1]  0.2359781 -1.6391814  1.4359409  0.2162272 -0.2232309  0.8492491  1.5157377
## [8]  0.2598194
```

```
X=matrix(rnorm(8),ncol=4)
X
```

```
##            [,1]       [,2]        [,3]        [,4]
## [1,] -0.6839369 -0.6553746 -0.05170818 -1.57097637
## [2,]  0.4828295 -1.1256493  1.14376289 -0.06115659
```

Choose the mean and the standard deviation:

```
X=matrix(rnorm(8,mean=100,sd=.1),ncol=2)
X
```

```
##           [,1]      [,2]
## [1,] 100.07034  99.99969
## [2,] 100.03677  99.94081
## [3,]  99.81583 100.07501
## [4,] 100.15932  99.88074
```

Picking specific elements:

```
X[1,2]
```

```
## [1] 99.99969
```

Pick an entire column (first column):

```
X[,1]
```

```
## [1] 100.07034 100.03677  99.81583 100.15932
```

Pick an entire row:

```
X[1,]
```

```
## [1] 100.07034  99.99969
```

Pick rows 3 & 4:

```
X[c(3,4),]
```

```
##           [,1]      [,2]
## [1,]  99.81583 100.07501
## [2,] 100.15932  99.88074
```

Sequences:

```
?seq
x=seq(1,10,by=2)
x
```

```
## [1] 1 3 5 7 9
```

Matrix Algebra operations:

```
X=matrix(seq(-1,-4,by=-1),ncol=2)
Y=matrix(seq(1,4),ncol=2)
X
```

```
##      [,1] [,2]
## [1,]   -1   -3
## [2,]   -2   -4
```

```
Y
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```
X+Y
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
```

```
X%*%Y
```

```
##      [,1] [,2]
## [1,]   -7  -15
## [2,]  -10  -22
```

Transpose:

```
t(X)
```

```
##      [,1] [,2]
## [1,]   -1   -2
## [2,]   -3   -4
```

Element-by-element operations:

```
sqrt(Y)
```

```
##          [,1]     [,2]
## [1,] 1.000000 1.732051
## [2,] 1.414214 2.000000
```

```
X*Y
```

```
##      [,1] [,2]
## [1,]   -1   -9
## [2,]   -4  -16
```

```
1/Y
```

```
##      [,1]      [,2]
## [1,]  1.0 0.3333333
## [2,]  0.5 0.2500000
```

```
Y^X
```

```
##      [,1]       [,2]
## [1,] 1.00 0.03703704
## [2,] 0.25 0.00390625
```

Data frames

This is what we call data sets in statistics (e.g. Stata): tabular data consisting of rows (observations) and columns (variables).

```
x=c(1,2,3,4)
y=c("male","male","female","female")
X=cbind(x,y)
X
```

```
##      x   y
## [1,] "1" "male"
## [2,] "2" "male"
## [3,] "3" "female"
## [4,] "4" "female"
```

When combining x and y in a matrix, x is converted into characters:

```
X
```

```
##      x   y
## [1,] "1" "male"
## [2,] "2" "male"
## [3,] "3" "female"
## [4,] "4" "female"
```

```
typeof(X)
```

```
## [1] "character"
```

Data frames can have variables (columns) of different types. There are relationships between the columns: each row is an observation.

```
Data=data.frame(years=x,gender=as.factor(y))
typeof(Data)
```

```
## [1] "list"
```

```
Data
```

```
##   years gender
## 1     1   male
## 2     2   male
## 3     3 female
## 4     4 female
```

Note that gender is now a factor! (Factors are variables that take on limited number of values. They are used to categorize data by levels. Can be integers or characters.)

```
class(Data$years)
```

```
## [1] "numeric"
```

```
class(Data$gender)
```

```
## [1] "factor"
```

The summary() and names() commands on X and Data:

```
names(X)
```

```
## NULL
```

```
summary(X)
```

```
##  x          y
##  1:1   female:2
##  2:1   male  :2
##  3:1
##  4:1
```

```
names(Data)
```

```
## [1] "years"  "gender"
```

```
summary(Data)
```

```
##      years         gender
##  Min.   :1.00   female:2
##  1st Qu.:1.75   male  :2
##  Median :2.50
##  Mean   :2.50
##  3rd Qu.:3.25
##  Max.   :4.00
```

Installing packages

```
install.packages("MASS", repos="https://cran.rstudio.com")
```

```
##
##   There is a binary version available but the source version is later:
##      binary source needs_compilation
## MASS 7.3-55 7.3-56              TRUE
```

```
## installing the source package 'MASS'
```

Working with data

Data can be read from external files using:

- read.table()
- read.csv()
- read.xlsx()

Many R packages come with imported data sets. We'll use them for this course.

Package "MASS" containts data on housing values in Boston area. Let's load the package, and attach the data set Boston so we don't have to refer to it all the time (there can be multiple data sets in a library):

```
library(MASS)
library(help="MASS")
attach(Boston)
?Boston
```

Quick inspection of the data:

```
summary(Boston)
```

```
##       crim                zn              indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm              age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

The first 4 observations:

```
Boston[1:4,]
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
```

```
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
```

Also the first 4 observations:

```
head(Boston,4)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
```

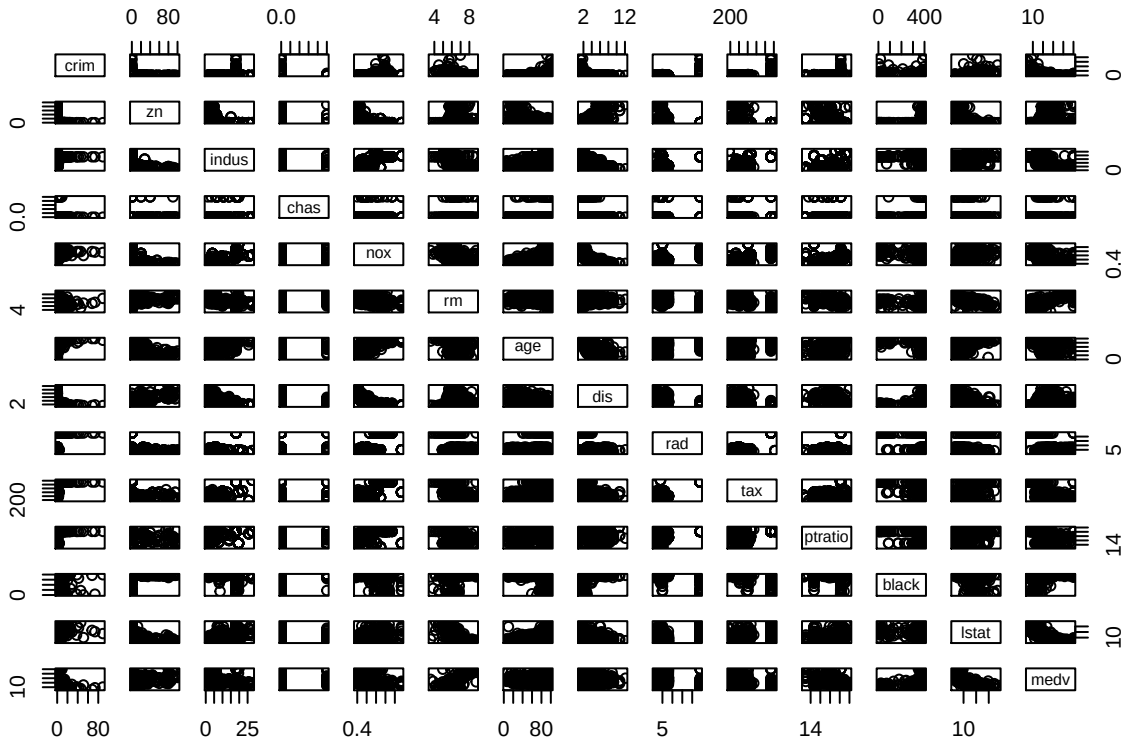
The last 4 observations:

```
tail(Boston,4)
```

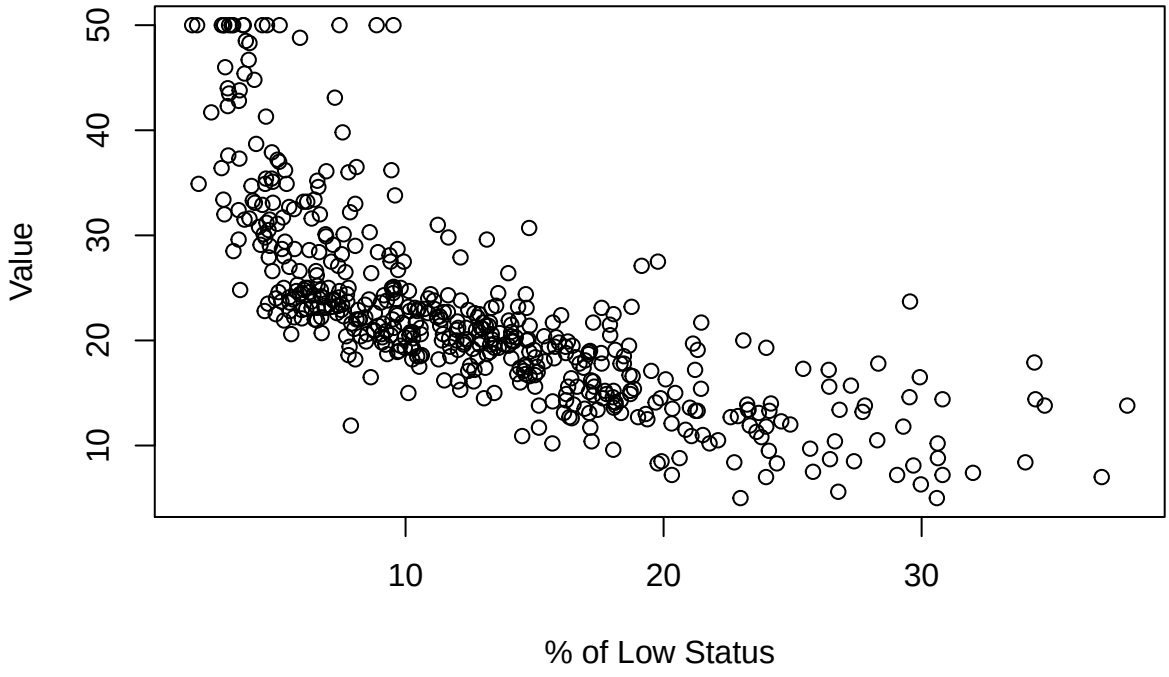
```
##        crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 503 0.04527  0 11.93    0 0.573 6.120 76.7 2.2875   1 273      21 396.90  9.08
## 504 0.06076  0 11.93    0 0.573 6.976 91.0 2.1675   1 273      21 396.90  5.64
## 505 0.10959  0 11.93    0 0.573 6.794 89.3 2.3889   1 273      21 393.45  6.48
## 506 0.04741  0 11.93    0 0.573 6.030 80.8 2.5050   1 273      21 396.90  7.88
##     medv
## 503 20.6
## 504 23.9
## 505 22.0
## 506 11.9
```

Plotting:

```
plot(Boston)
```




```
plot(lstat,medv,xlab="% of Low Status",ylab="Value")
```



Regression

Let's run a regression for medv:

```
reg=lm(medv~lstat+rm,data=Boston)
reg
```

```
##
```

```
## Call:
## lm(formula = medv ~ lstat + rm, data = Boston)
##
## Coefficients:
## (Intercept)        lstat           rm
##     -1.3583      -0.6424       5.0948
```

More info can be obtained by using the summary() command:

```
summary(reg)
```

```
##
## Call:
## lm(formula = medv ~ lstat + rm, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.076  -3.516  -1.010   1.909  28.131
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.35827    3.17283  -0.428    0.669
## lstat       -0.64236    0.04373 -14.689   <2e-16 ***
## rm           5.09479    0.44447  11.463   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.54 on 503 degrees of freedom
## Multiple R-squared:  0.6386, Adjusted R-squared:  0.6371
## F-statistic: 444.3 on 2 and 503 DF,  p-value: < 2.2e-16
```

Or we can get just the coefficients using the command coef():

```
coef(summary(reg))
```

```
##               Estimate Std. Error     t value     Pr(>|t|)
## (Intercept) -1.3582728 3.17282778  -0.4280953 6.687649e-01
## lstat       -0.6423583 0.04373146 -14.6886992 6.669365e-41
## rm           5.0947880 0.44446550  11.4627299 3.472258e-27
```

Or just the standard errors:

```
coef(summary(reg))[,2]
```

```
## (Intercept)       lstat          rm
##  3.17282778  0.04373146  0.44446550
```

```
C <- coef(summary(reg))[c(2,3),c(1,2)]
C
```

```
##         Estimate Std. Error
## lstat -0.6423583 0.04373146
## rm     5.0947880 0.44446550
```

To get heteroskedasticity robust standard errors, use package "sandwich".

```
library(sandwich)
```

Construct the robust variance covariance matrix:

```
rvcv=vcovHC(reg,type="HC")
```

The robust standard errors are on the diagonal:

```
sqrt(diag(rvcv))
```

```
## (Intercept)       lstat          rm
##  5.40374525  0.06374313  0.77111975
```

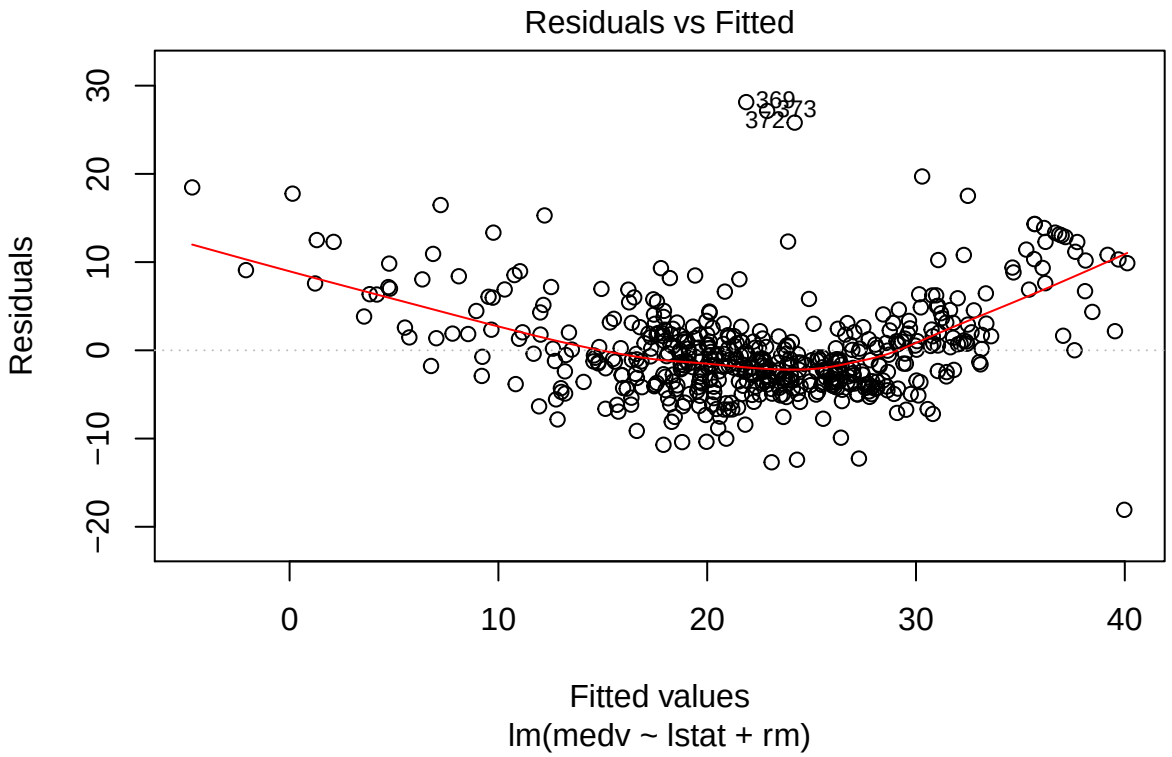
Let's compare with the non-robust:

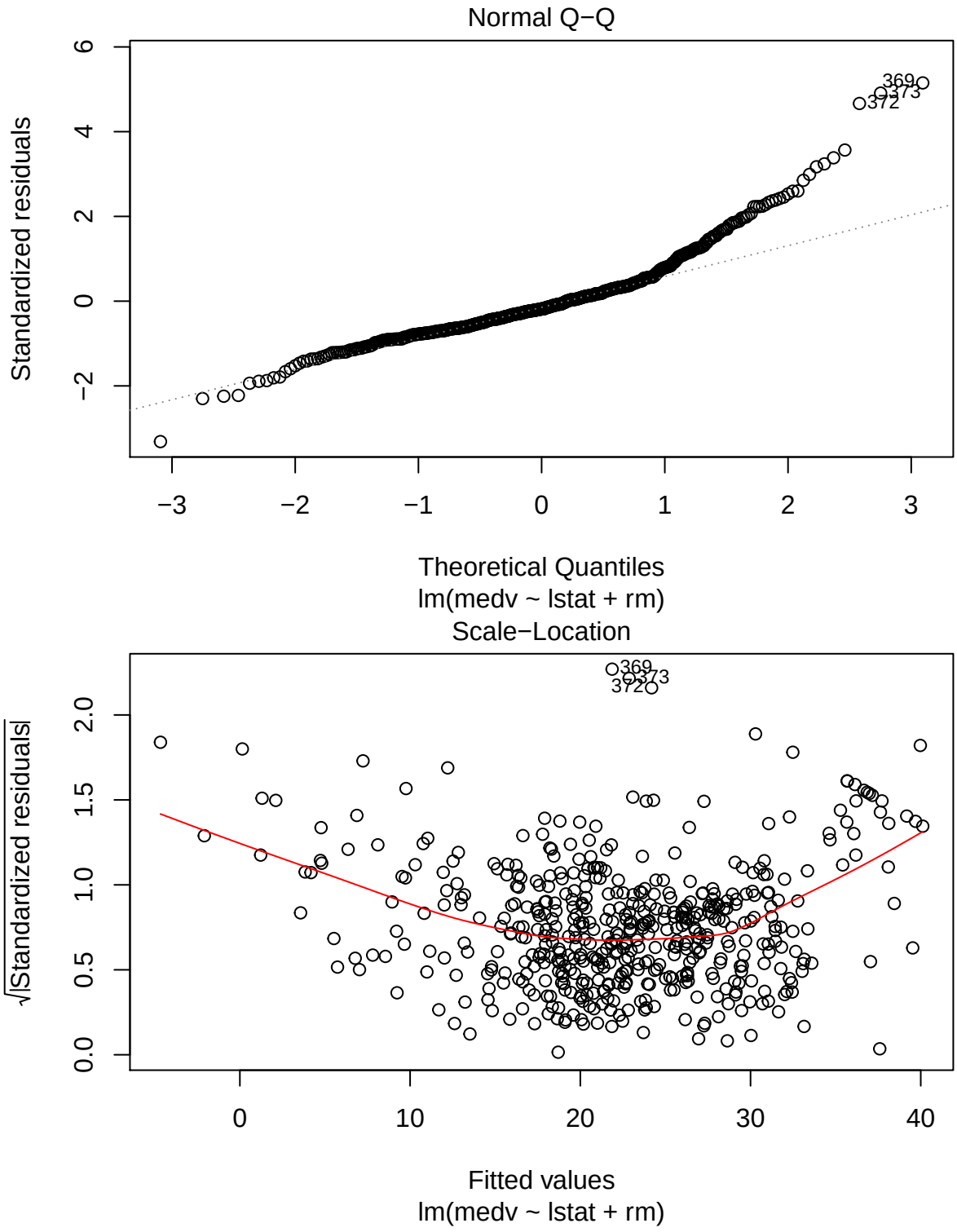
```
cbind("non-robust"=coef(summary(reg))[,2],"robust"=sqrt(diag(rvcv)))
```

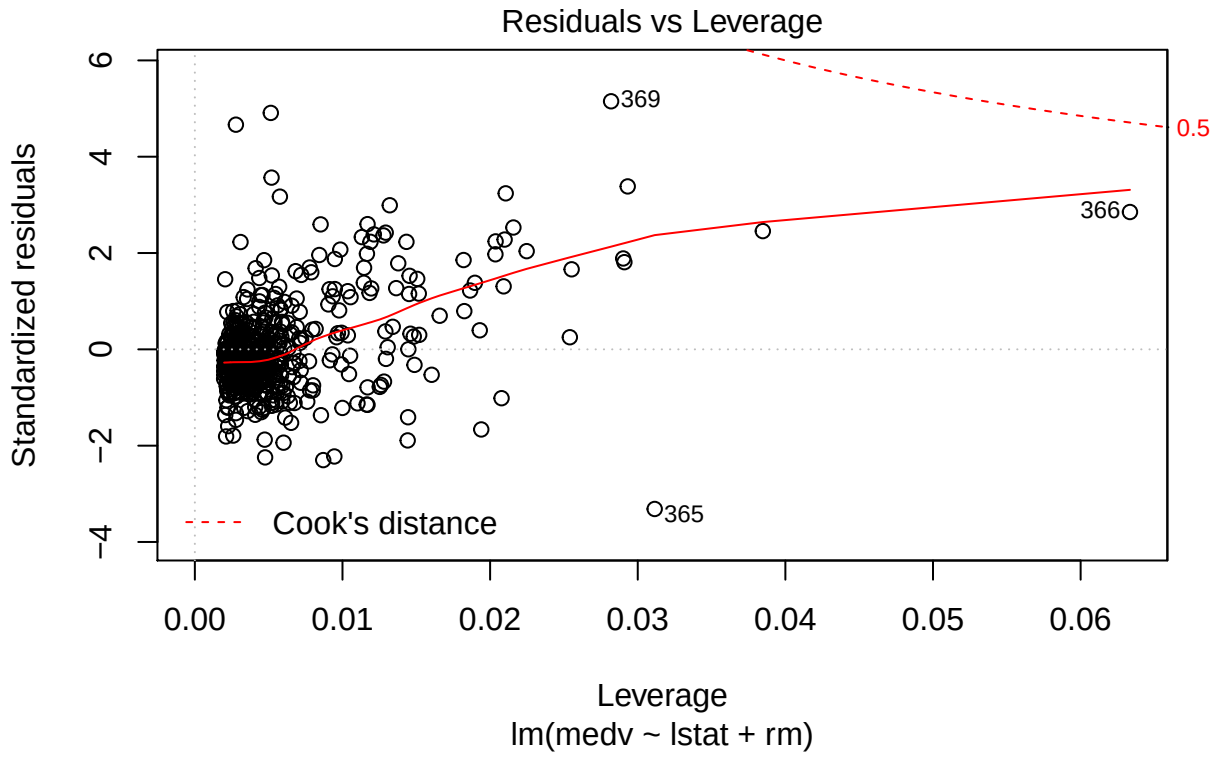
```
##             non-robust     robust
## (Intercept) 3.17282778 5.40374525
## lstat       0.04373146 0.06374313
## rm          0.44446550 0.77111975
```

The plot() command applied to the estimated regression object:

```
plot(reg)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(medv ~ lstat + rm)
Scale−Location
√|Standardized residuals|
Fitted values
lm(medv ~ lstat + rm)
369
373
372




```
names(reg)
```

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

Let's add the regression line to the plot of medv agains lstat, but we need to adjust the intercept for the average value of the rm variable, since we exclude rm from the plot.

- Pick the intercept and the coeff. on lstat:

```
betas=reg$coefficients[1:2]
betas
```

```
## (Intercept)       lstat
##  -1.3582728  -0.6423583
```

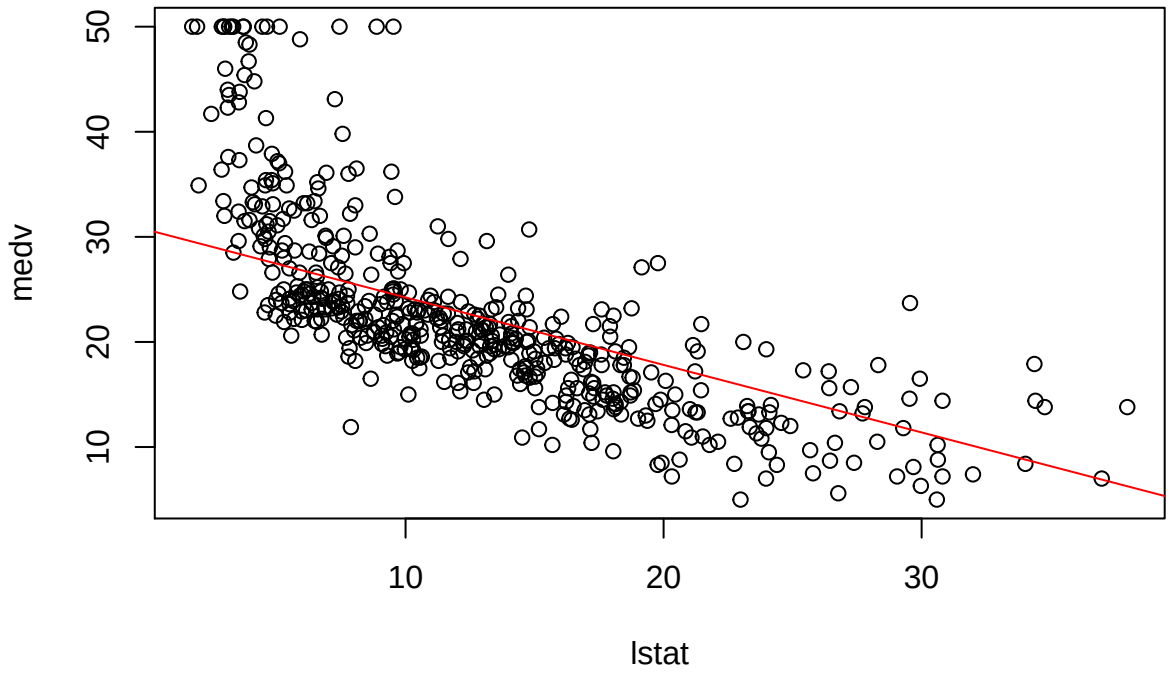
** If plot here, then no line added because control var rm is not included ** * Shift the intercept using the average of rm and its coeff.

```
betas[1]=betas[1]+reg$coefficients[3]*mean(rm)
betas[1]
```

```
## (Intercept)
##    30.66061
```

- Plot:

```
plot(lstat,medv)
abline(betas,col="red")
```



IV regression

Package: AER (Applied Econometrics with R):

```
library(AER)
```

```
## Loading required package: car
```

```
## Loading required package: carData
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: survival
```

```
?ivreg
```

Let's illustrate with data on cigarette consumption in CigaretteSW data set included with AER:

```
?CigarettesSW
data(CigarettesSW)
attach(CigarettesSW)
```

```
## The following object is masked from Boston:
##
##     tax
```

The command data() is used to load data sets.

OLS first:

```
summary(lm(packs~price))
```

```
##
## Call:
## lm(formula = packs ~ price)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.926  -9.621  -0.967   7.596  70.369
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 164.35649    6.90885  23.789  < 2e-16 ***
## price        -0.38463    0.04608  -8.348 5.92e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.71 on 94 degrees of freedom
## Multiple R-squared:  0.4257, Adjusted R-squared:  0.4196
## F-statistic: 69.68 on 1 and 94 DF,  p-value: 5.923e-13
```

Command ivreg() used without specifying instruments estimates OLS:

```
summary(ivreg(packs~price))
```

```
##
## Call:
## ivreg(formula = packs ~ price)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -53.9258  -9.6210  -0.9668   7.5961  70.3695
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 164.35649    6.90885  23.789  < 2e-16 ***
## price        -0.38463    0.04608  -8.348 5.92e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.71 on 94 degrees of freedom
## Multiple R-Squared: 0.4257,  Adjusted R-squared: 0.4196
## Wald test: 69.68 on 1 and 94 DF,  p-value: 5.923e-13
```

Robust standard errors:

```
summary(ivreg(packs~price),vcov=sandwich)
```

```
##
## Call:
## ivreg(formula = packs ~ price)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -53.9258  -9.6210  -0.9668   7.5961  70.3695
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 164.35649    6.62974  24.791  < 2e-16 ***
## price        -0.38463    0.04073  -9.444  2.8e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.71 on 94 degrees of freedom
## Multiple R-Squared: 0.4257,  Adjusted R-squared: 0.4196
## Wald test: 89.19 on 1 and 94 DF,  p-value: 2.798e-15
```

IV regression with income and tax as instruments:

```
summary(ivreg(packs~price | income+tax), vcov=sandwich, diagnostics = TRUE)
```

```
##
## Call:
## ivreg(formula = packs ~ price | income + tax)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -55.216 -10.367  -1.462   7.971  68.509
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 169.92173    7.54344  22.526  < 2e-16 ***
## price        -0.42342    0.04486  -9.439 2.87e-15 ***
##
## Diagnostic tests:
##                  df1 df2 statistic p-value
## Weak instruments   2  93   207.463  <2e-16 ***
## Wu-Hausman         1  93     2.790  0.0982 .
## Sargan             1  NA     1.541  0.2145
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.78 on 94 degrees of freedom
## Multiple R-Squared: 0.4214,  Adjusted R-squared: 0.4152
## Wald test: 89.09 on 1 and 94 DF,  p-value: 2.872e-15
```

The diagnostics part:

- "Weak instruments" is actually just the F-test for the null hypothesis that the first-stage coefficients on the IVs are equal to zero. Thus, this is not a test for weak instruments, and rejecting H_0 does not mean that the instruments are strong. Strictly speaking, one cannot test if instruments are weak. Unfortunately, the label "Weak instruments" used here is very misleading.
- Wu-Hausman: This is the Hausman specification test, which tests if the regressors (price in this case) are exogenous. Rejecting H_0 implies that the OLS and IV estimates are significantly different, which suggests endogeneity of the regressors.
- Sargan: This is the overidentifying restrictions specification test, which tests exogeneity of IVs (the null hypothesis). Rejecting the null hypothesis suggests that the instruments are endogenous (invalid).