STAT 3022 LAB WEEK 2

Loading Data Frame into R

Example 1: Climate of Twin Cities in Summer 2012

7. read.table – Load data frames into R

Download the file summer2012.csv from this course website to your machine, then load it into R.

```
A <- read.table("C:/Users/new/Desktop/STAT30222020spring/LAB1/summer2012.csv",
                header = TRUE, sep = ",")
#Apple: cmd+opt+c; Windows two backslash or slash
```

9. mean – What is the mean temperatures of summer 2012?

```
mean(A$AVG)
```

```
## [1] 75.04348
```

The answer is 75.04 .

9*. tapply(vector, index, function)

```
tapply(A$AVG,A$MN,mean)
```

```
##        6        7        8
## 72.50000 80.32258 72.22581
```

10. sd and var – How about the standard deviation and variance of the daily temperature in summer 2012?

```
sd(A$AVG)
```

```
## [1] 6.717065
```

```
var(A$AVG)
```

```
## [1] 45.11897
```

11. max and min – What are the warmest and coldest daily average temperatures, respectively?

```
max(A$AVG)
```

```
## [1] 91
```

```
min(A$AVG)
```

```
## [1] 59
```

To find the locations that the maximum and minimum occur, use which.max and which.min, respectively.

```
which.max(A$AVG)
```

```
## [1] 34
```

the maximum occurred on the 34th day of summer, e.g. July 4th.

```
which.min(A$AVG)
```

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

the minimum occurred on the 1st day of summer, e.g. June 1st.

To calculate pairwise maximums and minimums, use pmax and pmin, respectively.

```
pmax(c(2, 3, 4), c(5, 2, 6))
```

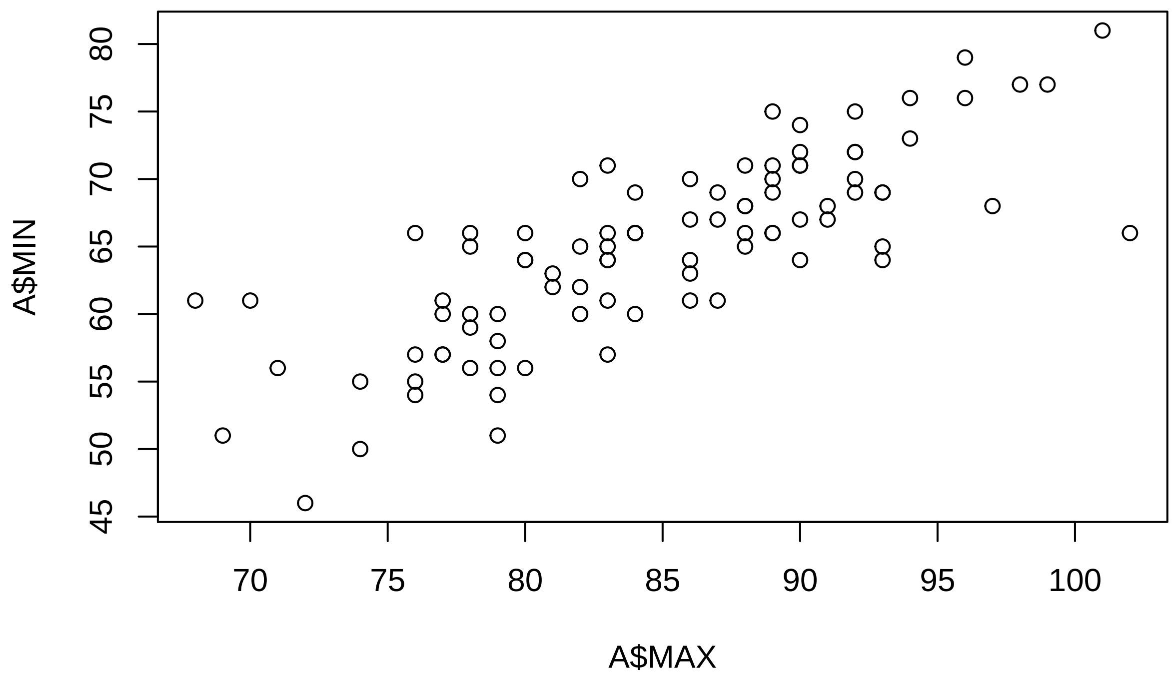
```
## [1] 5 3 6
```

```
pmin(c(2, 3, 4), c(5, 2, 6))
```

```
## [1] 2 2 4
```

12. plot – Scatter plot of maximum vs minimum temperatures

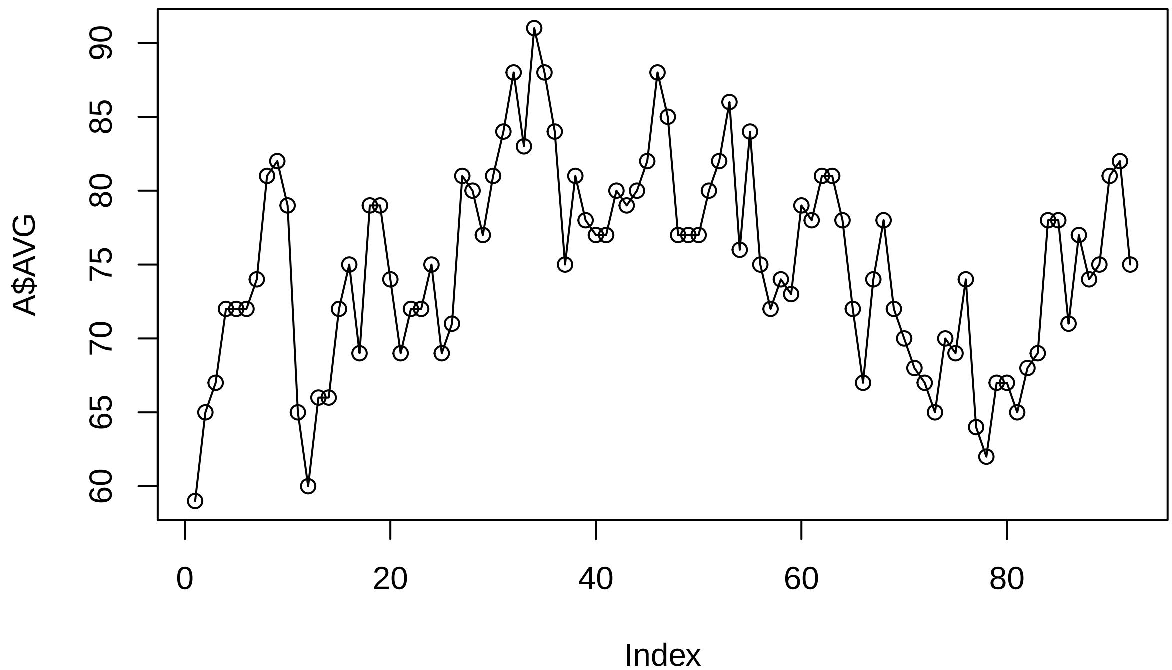
```
plot(A$MAX, A$MIN)
```



Scatter plot of the average temperature by day

13. lines – Connect the dots on the most recently drawn scatter plot, making it a line chart

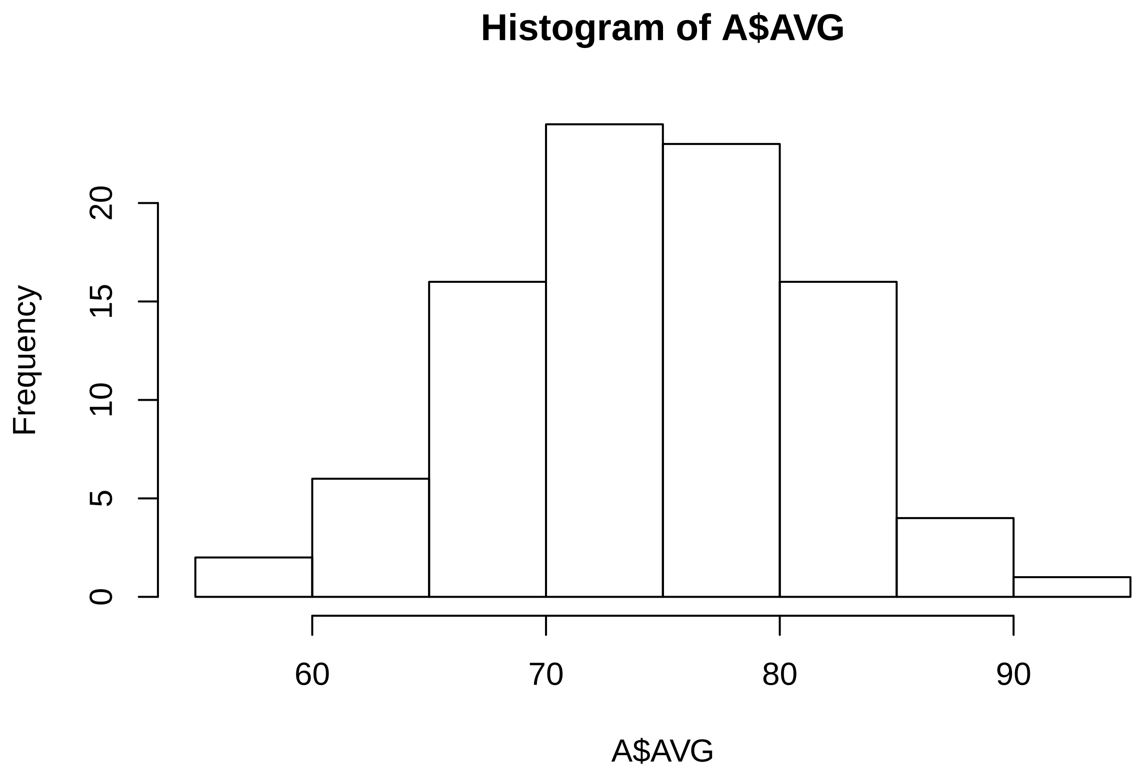
```
plot(A$AVG)
lines(A$AVG)
```



Note: Do not turn off the scatter plot before executing the lines command, or it would not work!

14. hist – Plotting histogram for the daily average temperature

```
hist(A$AVG)
```



SIMPLE LINEAR REGRESSION - AN EXAMPLE USING R

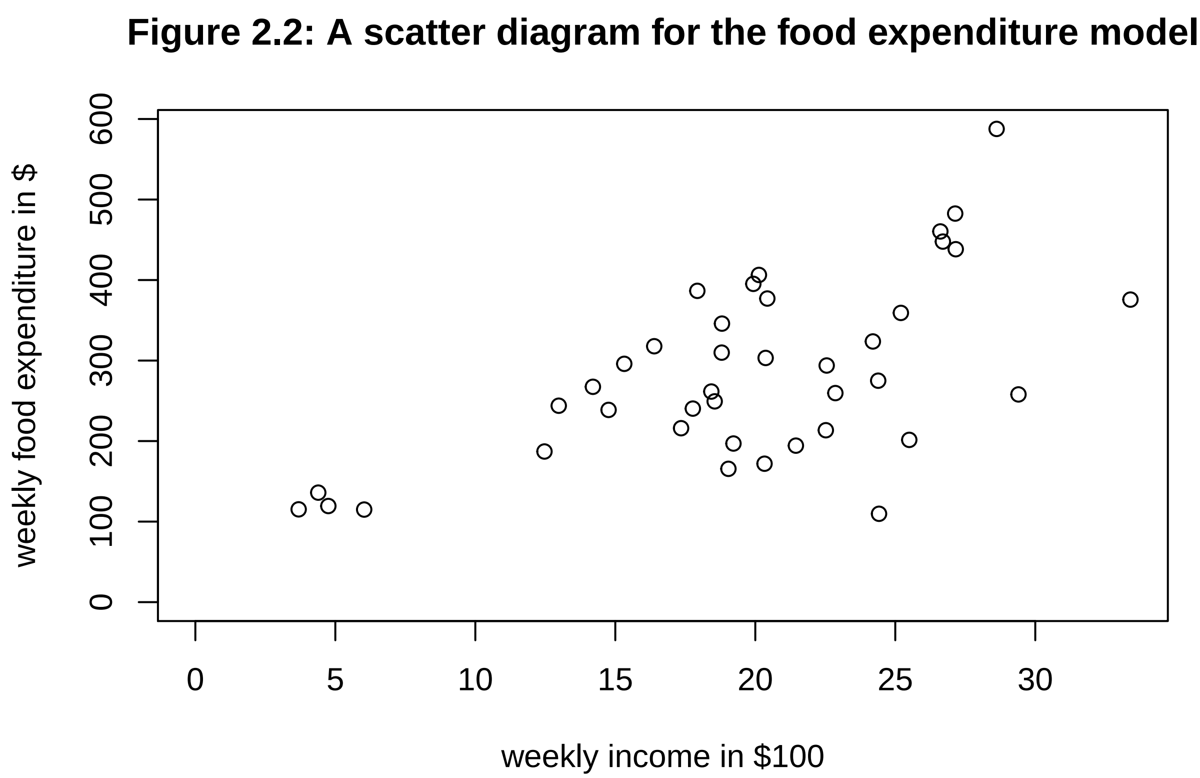
Example 1: Food Expenditure versus Income

```
food=read.table( "C:/Users/new/Desktop/STAT30222020spring/LAB2/food.csv", sep = ",",header = TRUE)
head(food)
```

```
##   food_exp income
## 1   115.22   3.69
## 2   135.98   4.39
## 3   119.34   4.75
## 4   114.96   6.03
## 5   187.05  12.47
## 6   243.92  12.98
```

It is always a good idea to visually inspect the data in a scatter diagram, which can be created using the function plot(). Figure 2.2 is a scatter diagram of food expenditure on income, suggesting that there is a positive relationship between income and food expenditure.

```
plot(food$income, food$food_exp,
     ylim=c(0, max(food$food_exp)),
     xlim=c(0, max(food$income)),
     xlab="weekly income in $100",
     ylab="weekly food expenditure in $",
     main="Figure 2.2: A scatter diagram for the food expenditure model"
     )
```



Estimating a Linear Regression

The R function for estimating a linear regression model is lm(y~x, data) which, used just by itself does not show any output; It is useful to give the model a name, such as mod1, then show the results using summary(mod1). If you are interested in only some of the results of the regression, such as the estimated coefficients, you can retrieve them using specific functions, such as the function coef(). For the food expenditure data, the regression model will be

food_exp=$\beta_0 + \beta_1 income + \epsilon$

```
mod1 <- lm(food_exp ~ income, data = food)
smod1 <- summary(mod1)
smod1
```

```
##
## Call:
## lm(formula = food_exp ~ income, data = food)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -223.025  -50.816   -6.324   67.879  212.044
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   83.416     43.410   1.922   0.0622 .
## income        10.210      2.093   4.877 1.95e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 89.52 on 38 degrees of freedom
## Multiple R-squared:  0.385,  Adjusted R-squared:  0.3688
## F-statistic: 23.79 on 1 and 38 DF,  p-value: 1.946e-05
```

```
b1 <- coef(mod1)[[1]]
b2 <- coef(mod1)[[2]]
```

The function coef() returns a list containing the estimated coefficients, where a specific coefficient can be accessed by its position in the list. For example, the estimated value of β_0 is b1 <- coef(mod1)[[1]], which is equal to 83.416002, and the estimated value of β_1 is b2 <- coef(mod1)[[2]], which is equal to 10.209643.

The intercept parameter, β_0, is usually of little importance in econometric models; we are mostly interested in the slope parameter, β_1. The estimated value of β_1 suggests that the food expenditure for an average family increases by 10.209643 when the family income increases by 1 unit, which in this case is $100. The R function abline() adds the regfression line to the prevoiusly plotted scatter diagram, as Figure 2.3 shows.

```
plot(food_exp ~ income, data = food)
abline(mod1)
```

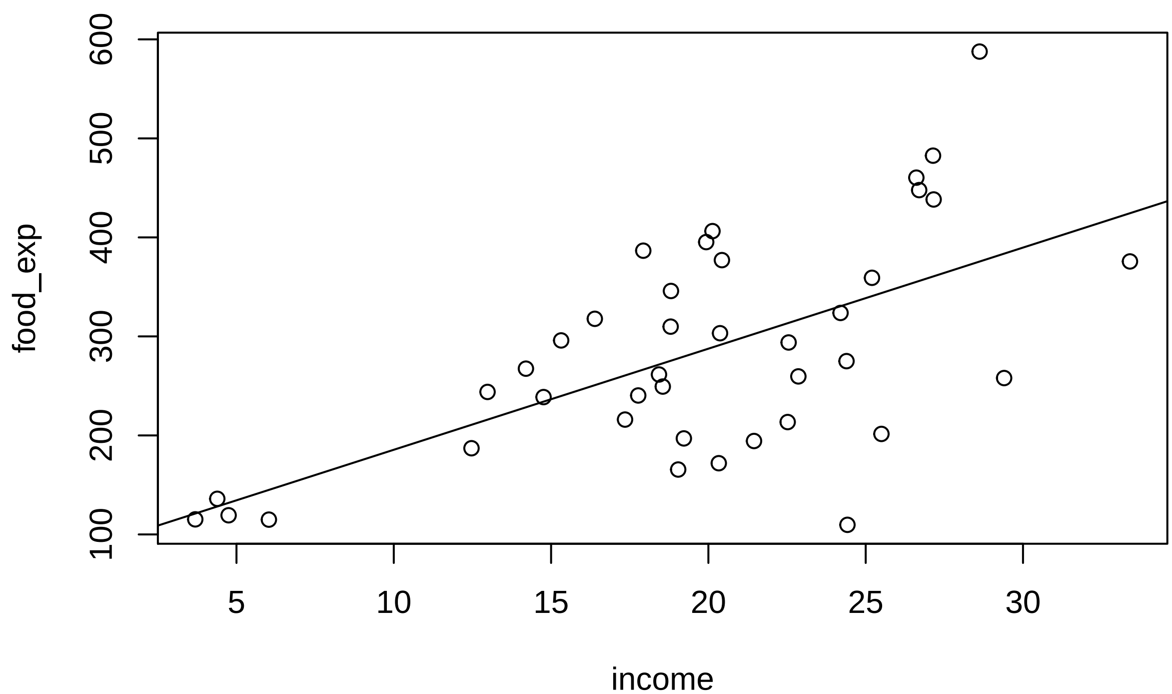


Figure 2.3: Scatter diagram and regression line for the food expenditure model How can one retrieve various regression results? These results exist in two R objects produced by the lm() function: the regression object, such as mod1 in the above code sequence, and the regression summary, which I denoted by smod1. The next code shows how to list the names of all results in each object.

```
names(mod1)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
```

```
## [9] "xlevels"       "call"          "terms"         "model"
```

```r
names(smod1)
```

```
## [1] "call"          "terms"         "residuals"     "coefficients"
## [5] "aliased"       "sigma"         "df"            "r.squared"
## [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

To retrieve a particular result you just refer to it with the name of the object, followed by the $ sign and the name of the result you wish to retrieve. For instance, if we want the vector of coefficients from mod1, we refer to it as mod$coefficients and smod1$coefficients:

```r
mod1$coefficients
```

```
## (Intercept)      income
##    83.41600    10.20964
```

```r
smod1$coefficients
```

```
##             Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 83.41600  43.410163 1.921578 6.218242e-02
## income      10.20964   2.093264 4.877381 1.945862e-05
```

As we have seen before, however, some of these results can be retrieved using specific functions, such as coef(mod1), resid(mod1), fitted(mod1), and vcov(mod1).

Example 2

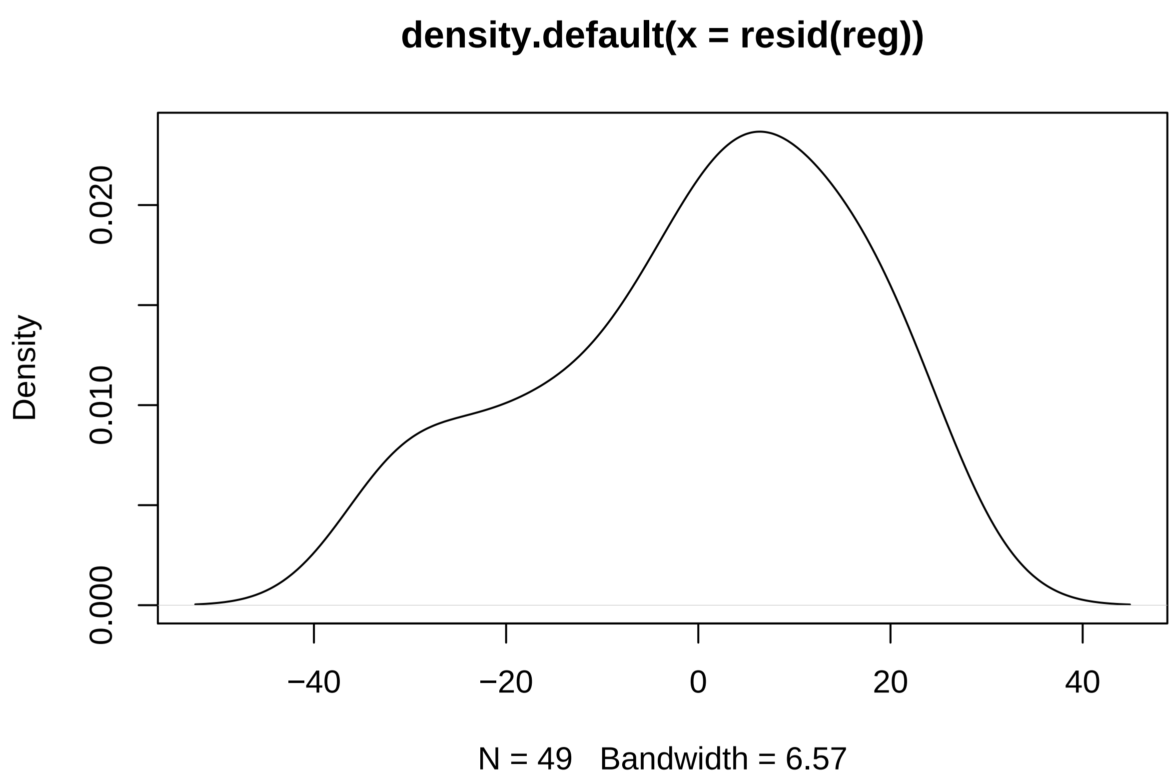
```r
x = c(21,34,6,47,10,49,23,32,12,16,29,49,28,8,57,9,31,10,21,
      26,31,52,21,8,18,5,18,26,27,26,32,2,59,58,19,14,16,9,23,
      28,34,70,69,54,39,9,21,54,26)
y = c(47,76,33,78,62,78,33,64,83,67,61,85,46,53,55,71,59,41,82,
      56,39,89,31,43,29,55, 81,82,82,85,59,74,80,88,29,58,71,60,
      86,91,72,89,80,84,54,71,75,84,79)
reg <- lm(y~x)  #Create a linear model
summary(reg)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.641  -8.865   3.449  12.438  25.213
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  52.8889     4.4944  11.768  1.3e-15 ***
## x             0.4606     0.1351   3.411  0.00134 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.57 on 47 degrees of freedom
## Multiple R-squared:  0.1984, Adjusted R-squared:  0.1814
## F-statistic: 11.63 on 1 and 47 DF,  p-value: 0.00134
```

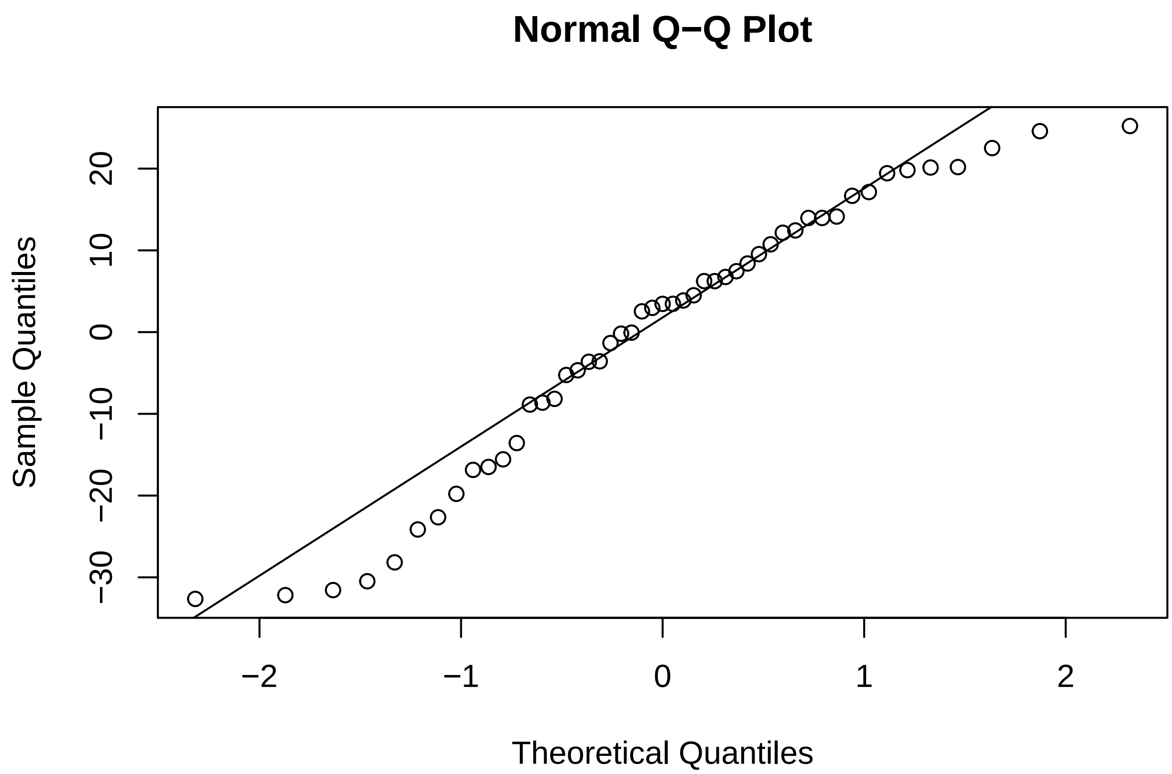
```r
resid(reg) #List of residuals
```

```
##           1           2           3           4           5
## -15.56228406   7.44941046 -22.65270081   3.46110498   4.50474366
##           6           7           8           9          10
##   2.53982722 -30.48356182  -3.62931177  24.58346589   6.74091036
##          11          12          13          14          15
##  -5.24739512   9.53982722 -19.78675624  -3.57397858 -24.14528385
##          16          17          18          19          20
##  13.96538254  -8.16867289 -16.49525634  19.43771594  -8.86547847
##          21          22          23          24          25
## -28.16867289  12.15791057 -31.56228406 -13.57397858 -32.18036741
##          26          27          28          29          30
##  -0.19206193  19.81963259  17.13452153  16.67388265  20.13452153
##          31          32          33          34          35
##  -8.62931177  20.18985472  -0.06656161   8.39407727 -32.64100629
##          36          37          38          39          40
##  -1.33781187  10.74091036   2.96538254  22.51643818  25.21324376
##          41          42          43          44          45
##   3.44941046   3.86641067  -4.67295044   6.23663280 -16.85378395
##          46          47          48          49
##  13.96538254  12.43771594   6.23663280  14.13452153
```

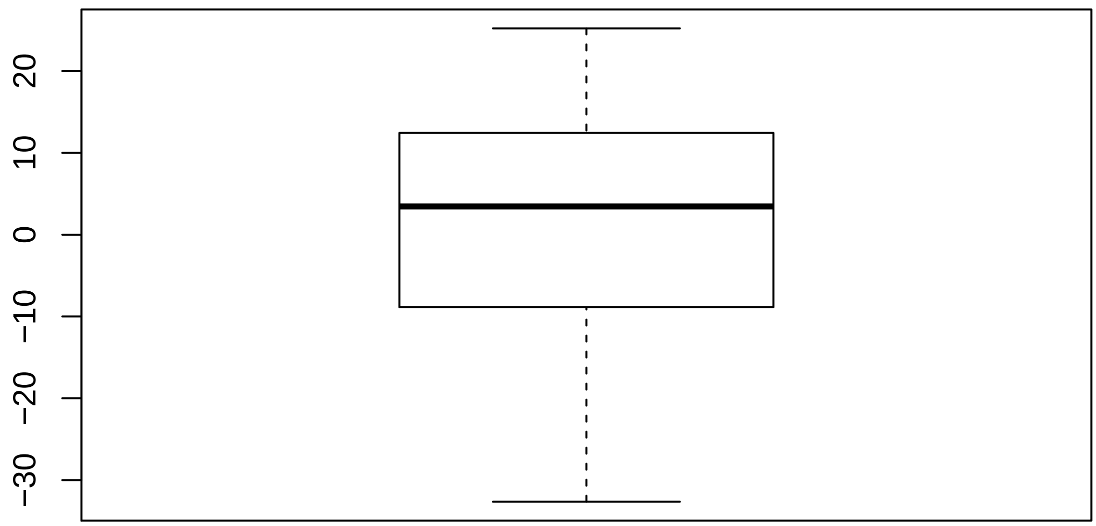
```r
plot(density(resid(reg))) #A density plot
```



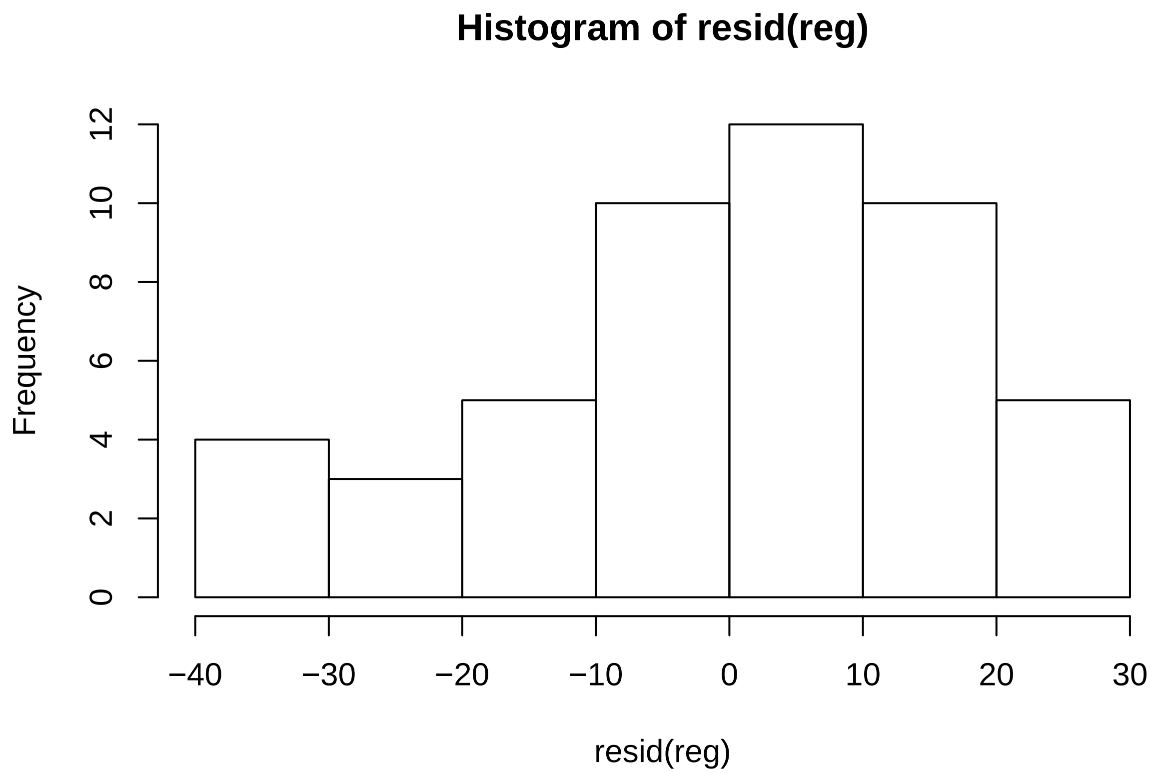

```r
qqnorm(resid(reg)) # A quantile normal plot - good for checking normality
qqline(resid(reg))
```




```
boxplot(resid(reg)) #A Boxplot plot
```




```r
hist(resid(reg)) #A Histogram plot
```


Histogram of resid(reg)
Frequency
0
2
4
6
8
10
12
−40
−30
−20
−10
0
10
20
30
resid(reg)