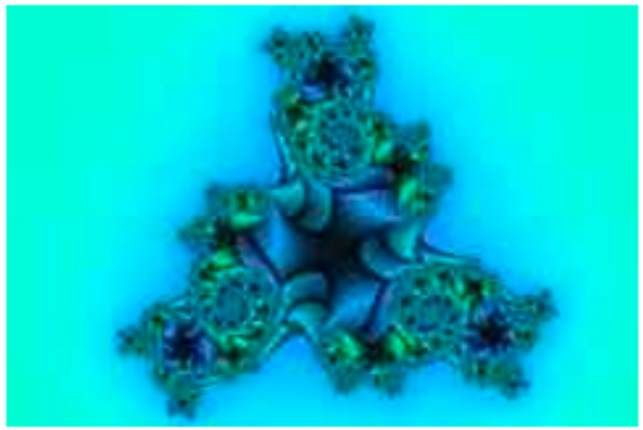
http://www.r-tutor.com/

Simple Linear Regression

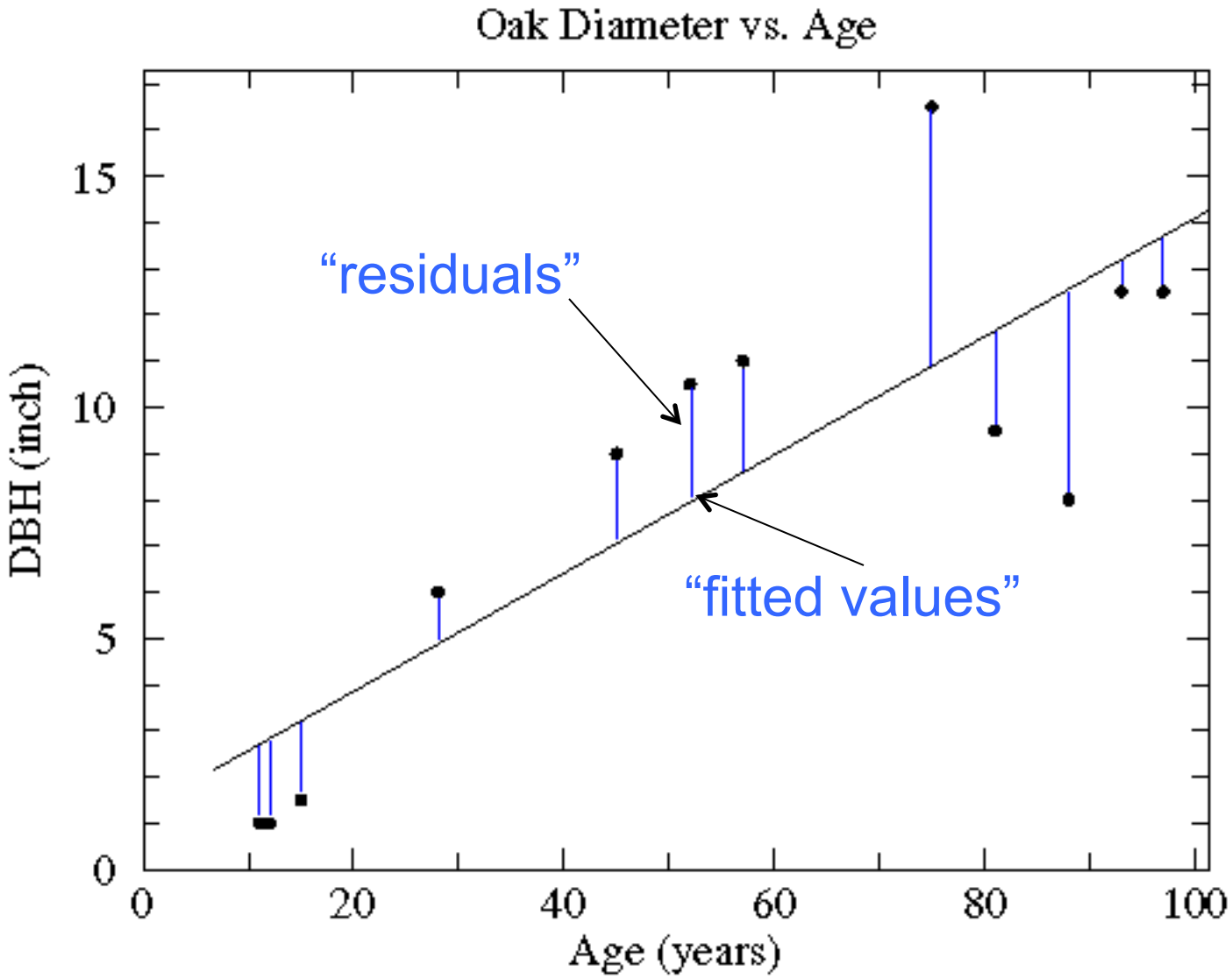


A **simple linear regression model** that describes the relationship between two variables *x* and *y* can be expressed by the following equation. The numbers *α* and *β* are called **parameters**, and *ϵ* is the **error term**.

$$y = \alpha + \beta x + \epsilon$$

For example, in the data set faithful, it contains sample data of two random variables named waiting and eruptions. The waiting variable denotes the waiting time until the next eruptions, and eruptions denotes the duration. Its linear regression model can be expressed as:

$$Eruptions = \alpha + \beta * Waiting + \epsilon$$



Least squares: minimize the sum of the squared residuals

Problem

Apply the simple linear regression model for the data set faithful, and estimate the next eruption duration if the waiting time since the last eruption has been 80 minutes.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)
```

Then we extract the parameters of the estimated regression equation with the coefficients function.

```
> coeffs = coefficients(eruption.lm); coeffs
(Intercept)     waiting
  -1.874016    0.075628
```

We now fit the eruption duration using the estimated regression equation.

```
> waiting = 80           # the waiting time
> duration = coeffs[1] + coeffs[2]*waiting
> duration
(Intercept)
     4.1762
```

Alternative Solution

We wrap the waiting parameter value inside a new data frame named newdata.

```
> newdata = data.frame(waiting=80) # wrap the parameter
```

Then we apply the predict function to eruption.lm along with newdata.

```
> predict(eruption.lm, newdata)    # apply predict
     1
4.1762
```

The **coefficient of determination** of a linear regression model is the quotient of the variances of the fitted values and observed values of the dependent variable. If we denote y_i as the observed values of the dependent variable, $\bar{y}$ as its mean, and $\hat{y}_i$ as the fitted value, then the coefficient of determination is:

$$r^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2}$$

Problem

Find the coefficient of determination for the simple linear regression model of the data set faithful.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)
```

Then we extract the coefficient of determination from the r.squared attribute of its summary.

```
> summary(eruption.lm)$r.squared
[1] 0.81146
```

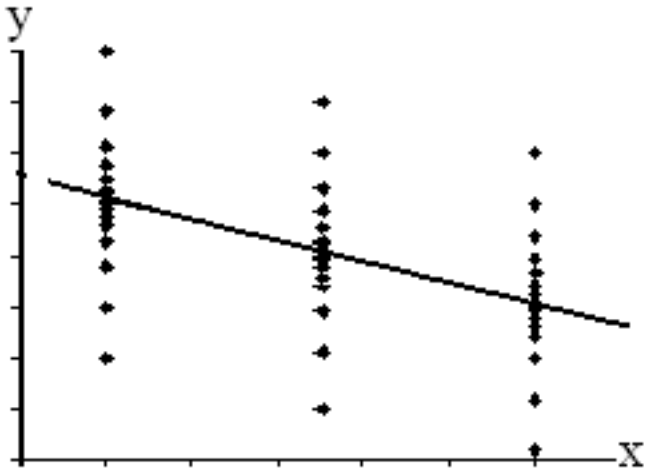
Answer

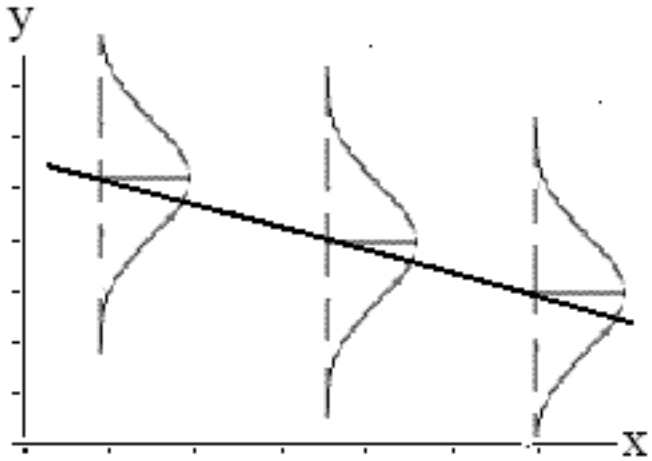
The coefficient of determination of the simple linear regression model for the data set faithful is 0.81146.

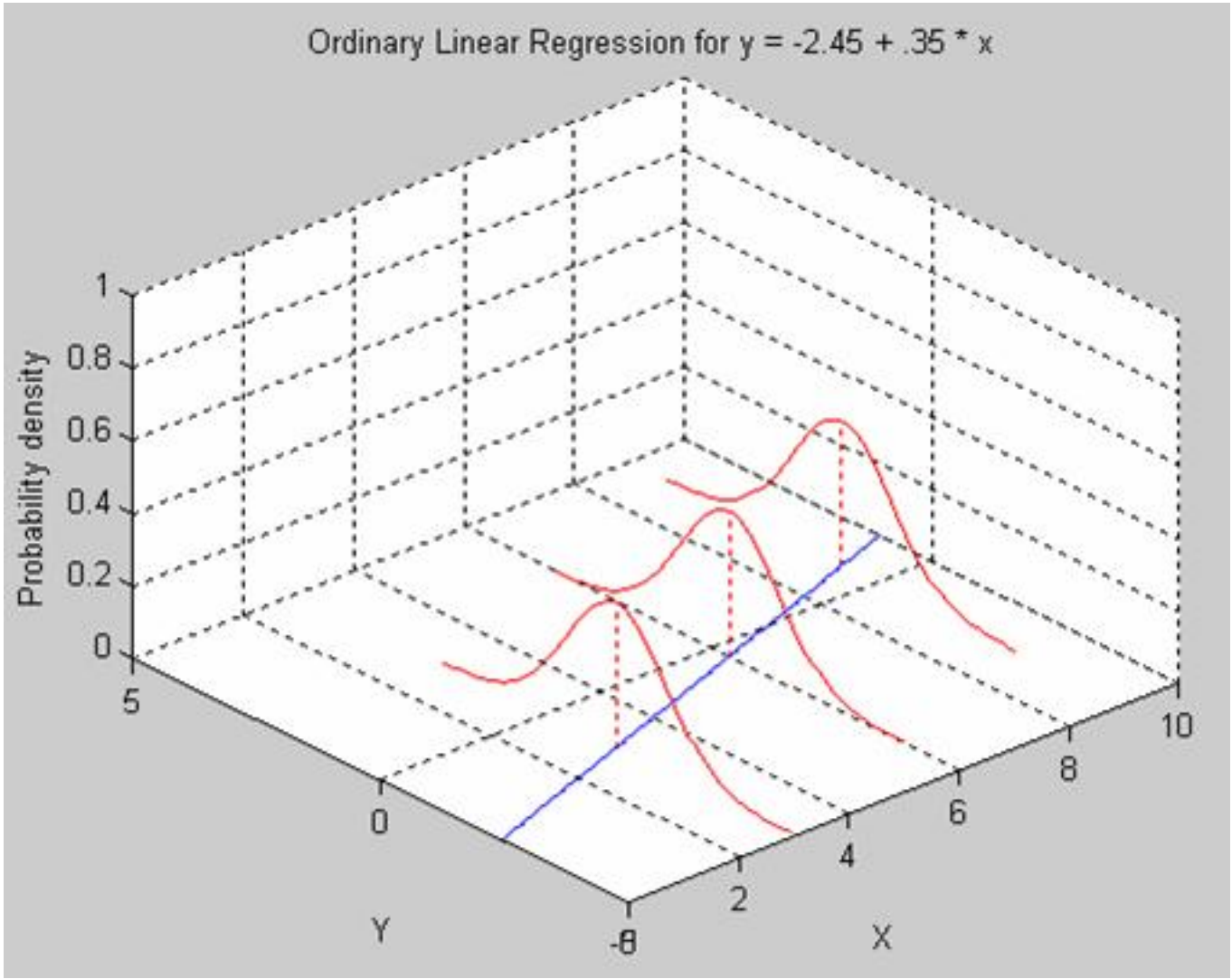
Significance Test for Linear Regression

Assume that the error term ϵ in the linear regression model is independent of x, and is normally distributed, with zero mean and constant variance. We can decide whether there is any **significant relationship** between x and y by testing the null hypothesis that $\beta = 0$.

```
> summary(eruption.lm)

Call:
lm(formula = eruptions ~ waiting, data = faithful)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2992 -0.3769  0.0351  0.3491  1.1933

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.87402    0.16014   -11.7   <2e-16 ***
waiting      0.07563    0.00222    34.1   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.497 on 270 degrees of freedom
Multiple R-squared: 0.811,      Adjusted R-squared: 0.811
F-statistic: 1.16e+03 on 1 and 270 DF,  p-value: <2e-16
```


y
x


y
x


Ordinary Linear Regression for y = -2.45 + .35 * x
Probability density
1
0.8
0.6
0.4
0.2
0
5
0
-5
Y
2
4
6
8
10
X

Problem

In the data set faithful, develop a 95% confidence interval of the mean eruption duration for the waiting time of 80 minutes.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

```
> attach(faithful)     # attach the data frame
> eruption.lm = lm(eruptions ~ waiting)
```

Then we create a new data frame that set the waiting time value.

```
> newdata = data.frame(waiting=80)
```

We now apply the predict function and set the predictor variable in the newdata argument. We also set the interval type as "confidence", and use the default 0.95 confidence level.

```
> predict(eruption.lm, newdata, interval="confidence")
     fit    lwr    upr
1 4.1762 4.1048 4.2476
```

Problem

In the data set faithful, develop a 95% prediction interval of the eruption duration for the waiting time of 80 minutes.

Solution

We apply the lm function to a formula that describes the variable eruptions by the variable waiting, and save the linear regression model in a new variable eruption.lm.

```
> attach(faithful)     # attach the data frame
> eruption.lm = lm(eruptions ~ waiting)
```

Then we create a new data frame that set the waiting time value.

```
> newdata = data.frame(waiting=80)
```

We now apply the predict function and set the predictor variable in the newdata argument. We also set the interval type as "predict", and use the default 0.95 confidence level.

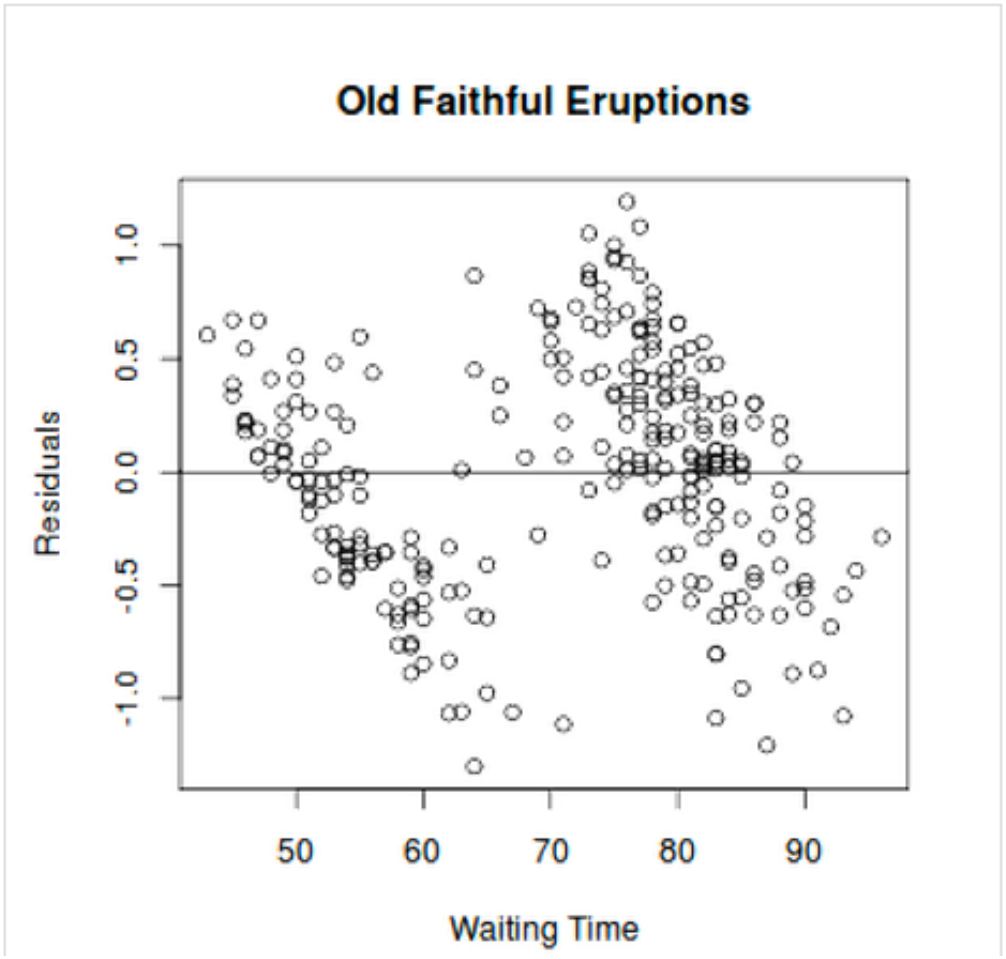
```
> predict(eruption.lm, newdata, interval="predict")
     fit    lwr    upr
1 4.1762 3.1961 5.1564
```

Residual Plot

```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)
> eruption.res = resid(eruption.lm)
```

We now plot the residual against the observed values of the variable waiting.

```
> plot(faithful$waiting, eruption.res,
+     ylab="Residuals", xlab="Waiting Time",
+     main="Old Faithful Eruptions")
> abline(0, 0)                  # the horizon
```

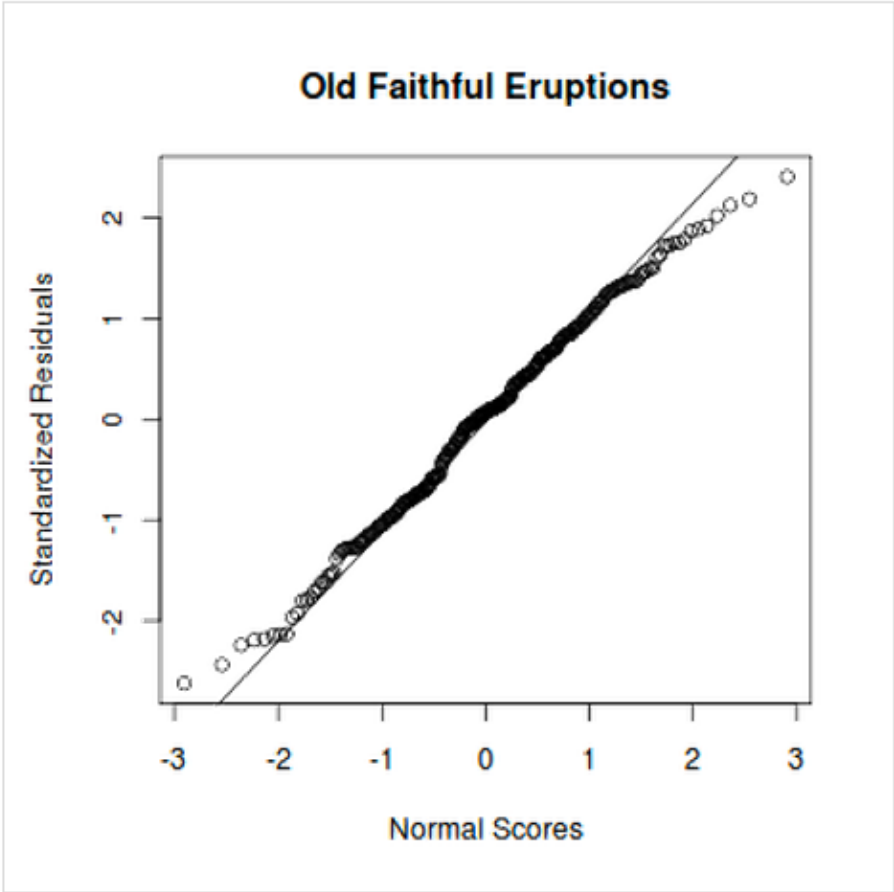


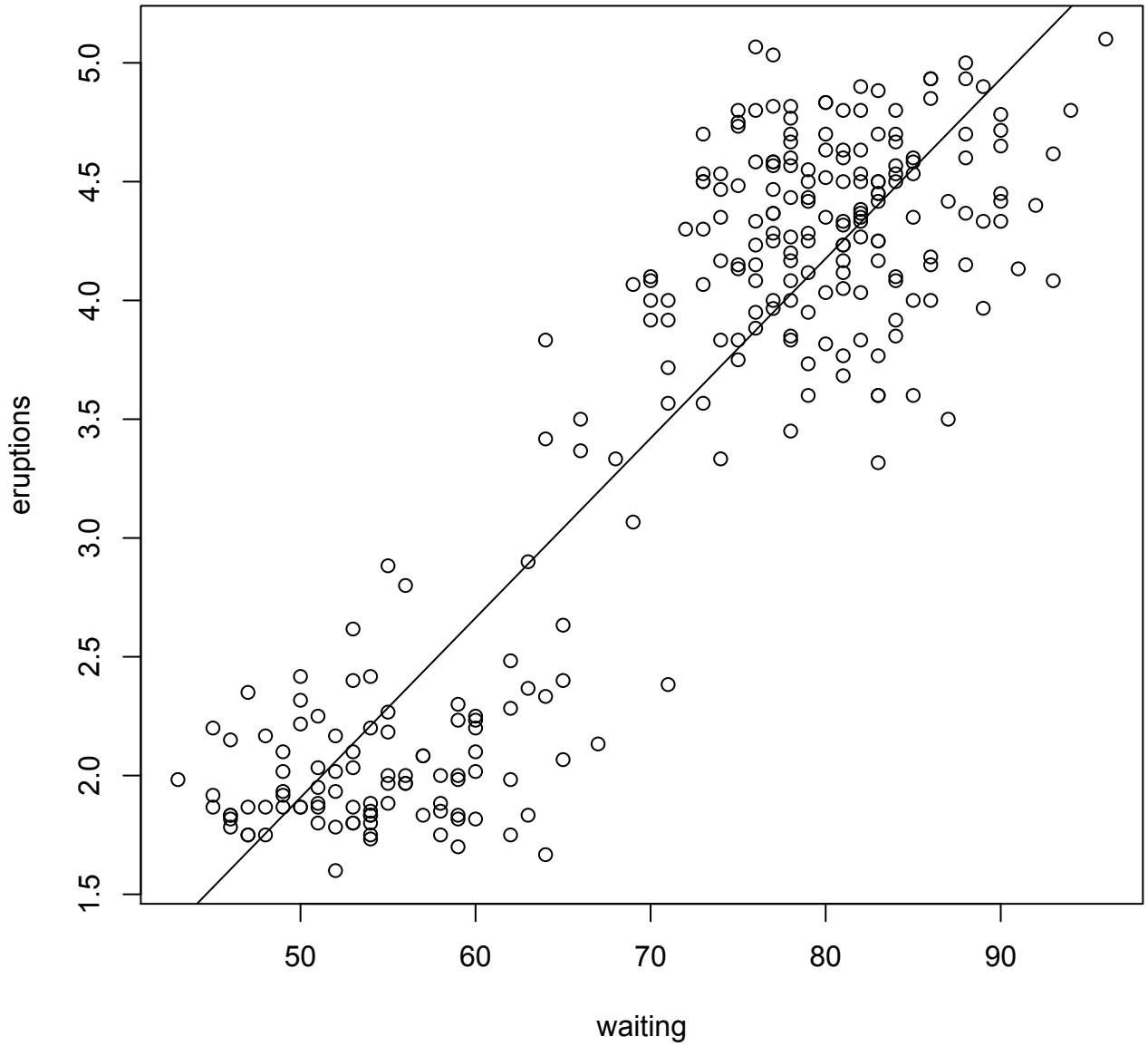
Normal Probability Plot of Residuals

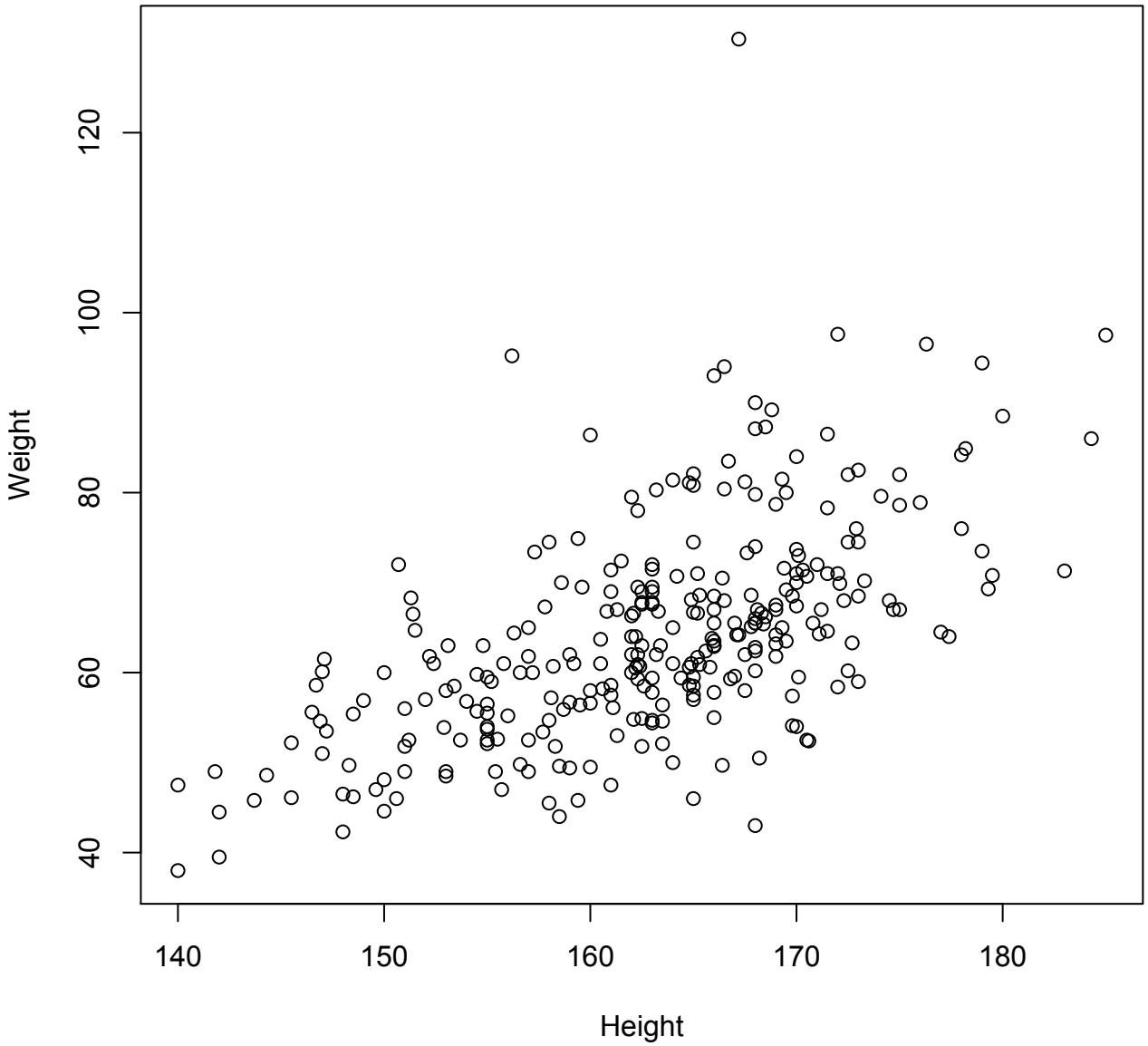
```
> eruption.lm = lm(eruptions ~ waiting, data=faithful)
> eruption.stdres = rstandard(eruption.lm)
```

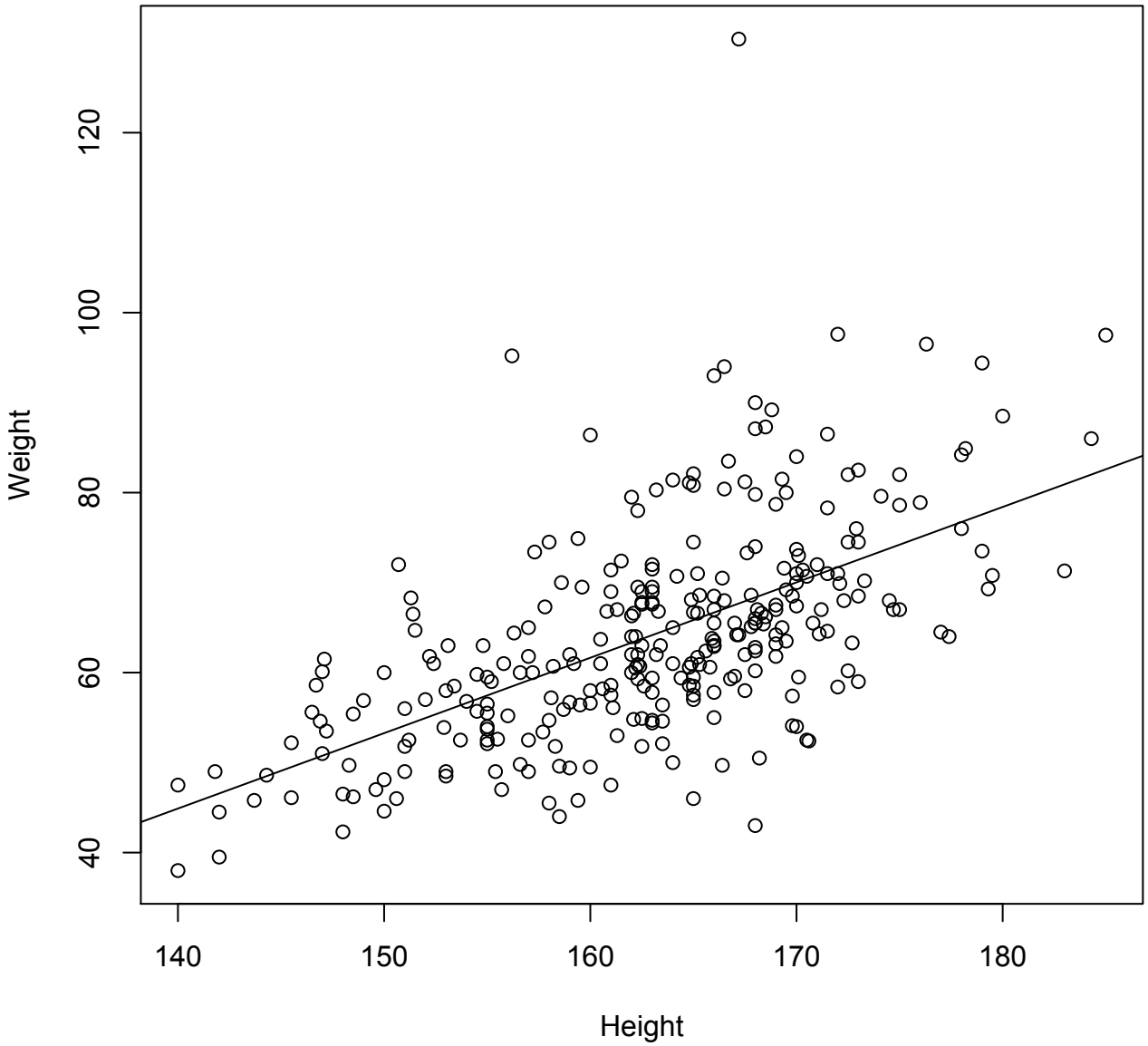
We now create the normal probability plot with the qqnorm function, and add the qqline for further comparison.

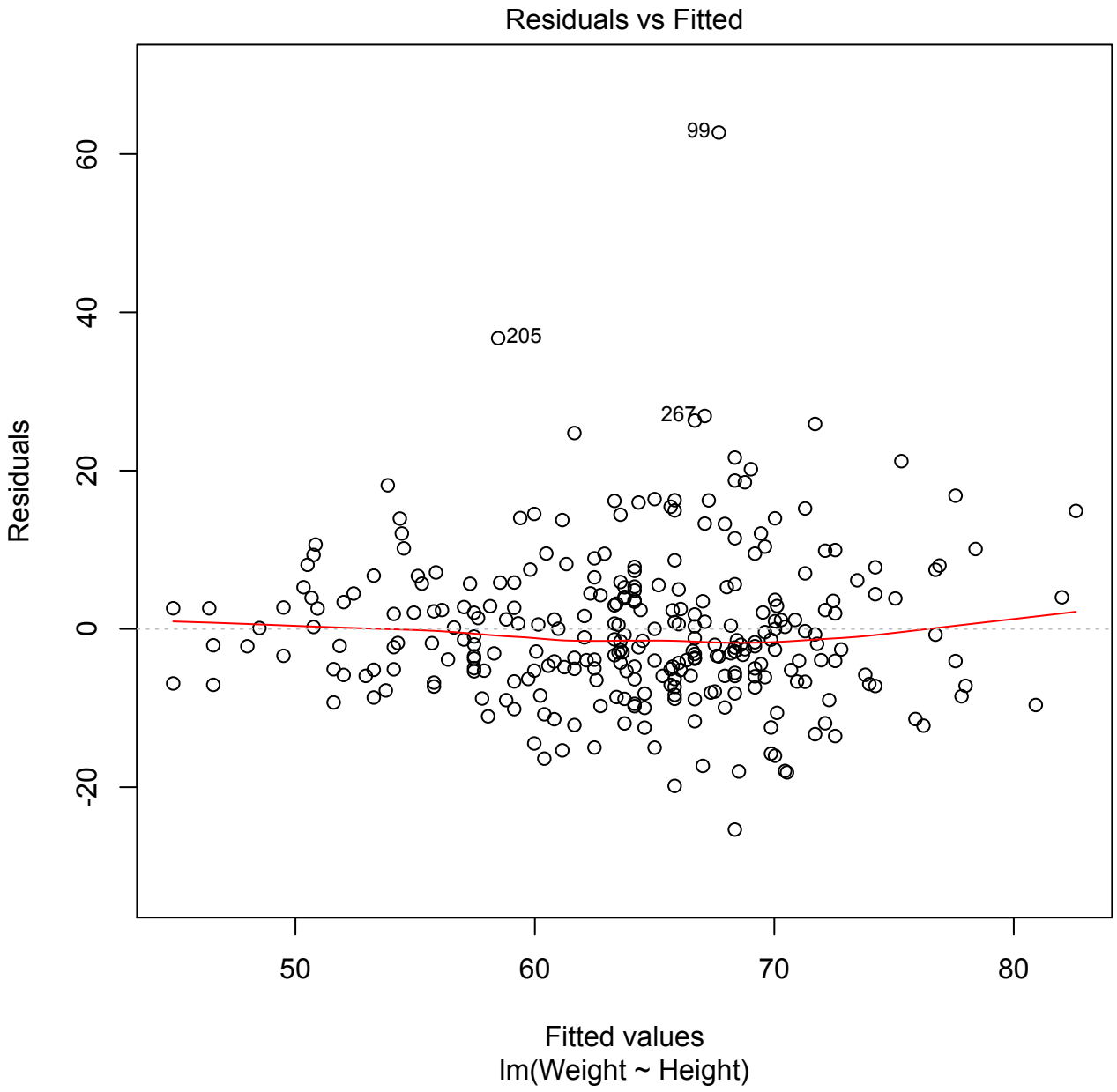
```
> qqnorm(eruption.stdres,
+      ylab="Standardized Residuals",
+      xlab="Normal Scores",
+      main="Old Faithful Eruptions")
> qqline(eruption.stdres)
```

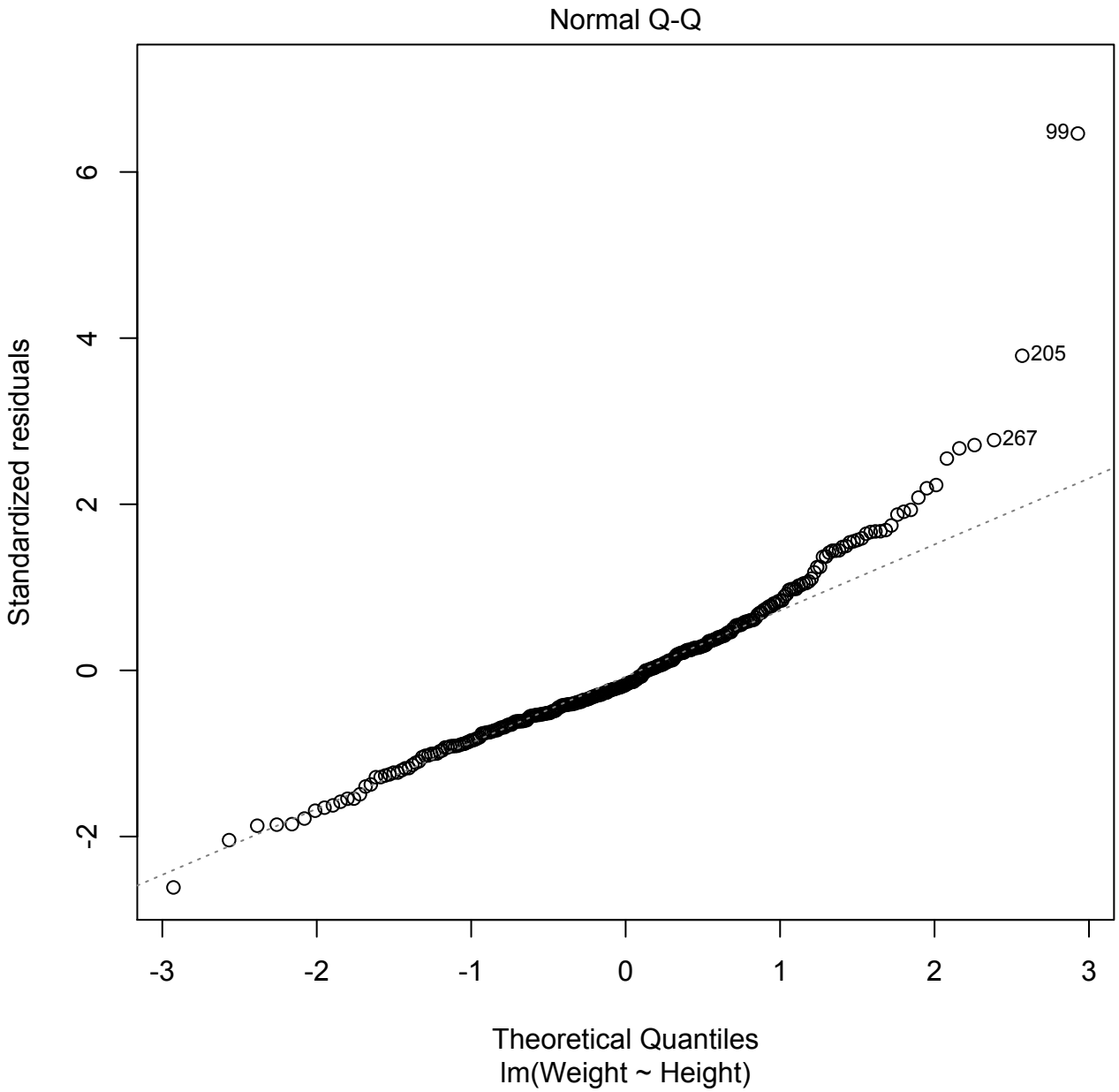



eruptions
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
50
60
70
80
90
waiting


Weight
120
100
80
60
40
140
150
160
170
180
Height


Weight
120
100
80
60
40
140
150
160
170
180
Height


Residuals vs Fitted
99
205
267
Residuals
60
40
20
0
-20
50
60
70
80
Fitted values
lm(Weight ~ Height)

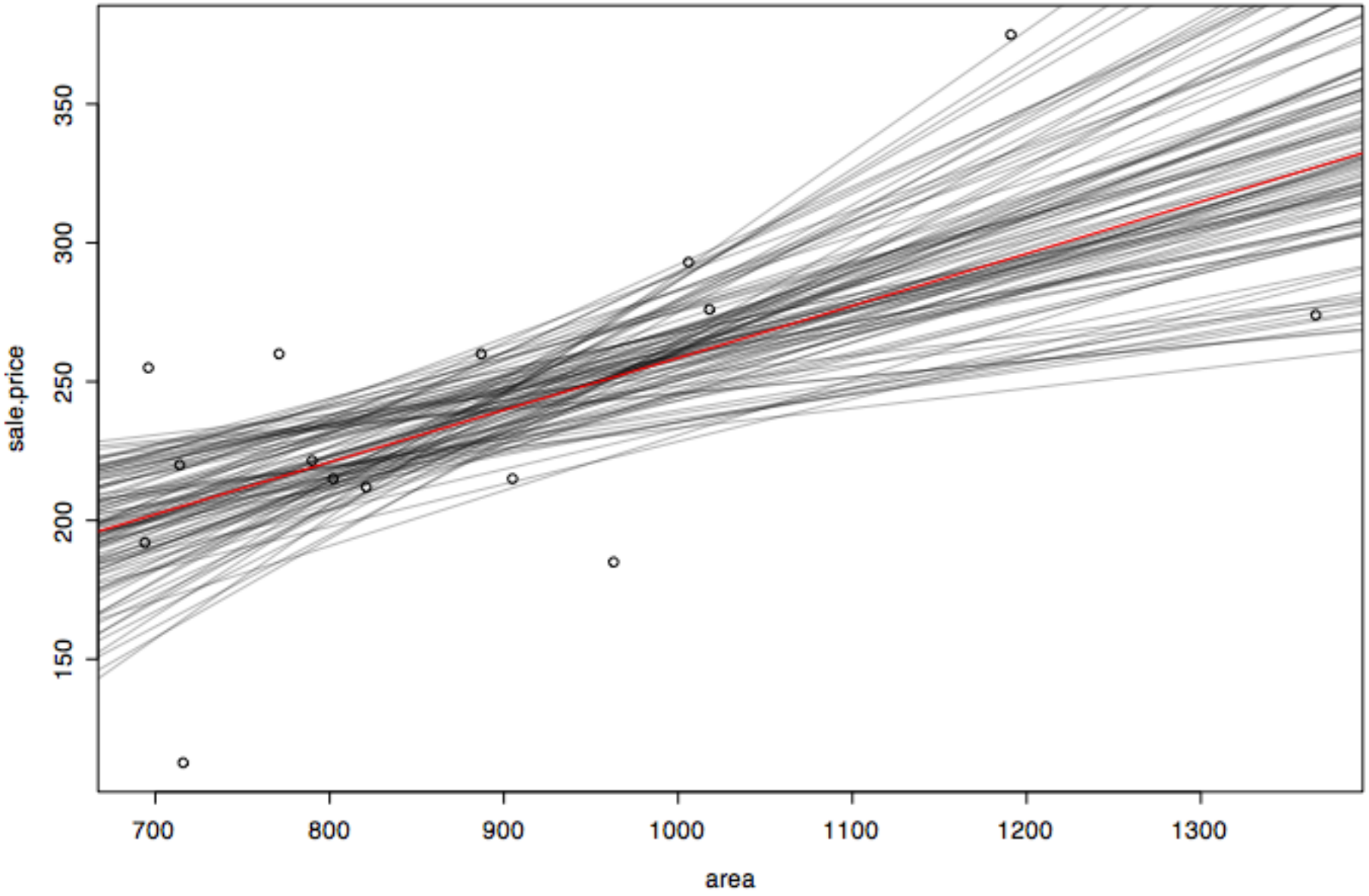

Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(Weight ~ Height)
99
205
267
-3
-2
-1
0
1
2
3
-2
0
2
4
6

The Bootstrap

```
foo <- rnorm(100)
t.test(foo)

b_means<-rep(0,1000)
for (i in 1:1000) {b_means[i] <- mean(sample(foo,replace=TRUE))}
b_means <- sort(b_means)
b_means[25]
b_means[975]
```

bootstrapping the linear model

```
plot(sale.price ~ area, data=houseprices)
houseprices.lm <- lm (sale.price ~ area, data=houseprices)

houseprices.fn <- function(houseprices, index) {
 house.resample <- houseprices[index,]
 house.lm <- lm(sale.price ~ area, data=house.resample)
 abline(house.lm,lwd=0.3)
 coef(house.lm)[2]
}

houseprices.boot <- boot(houseprices, R=99, statistic=houseprices.fn)
abline(houseprices.lm,col="red")
```

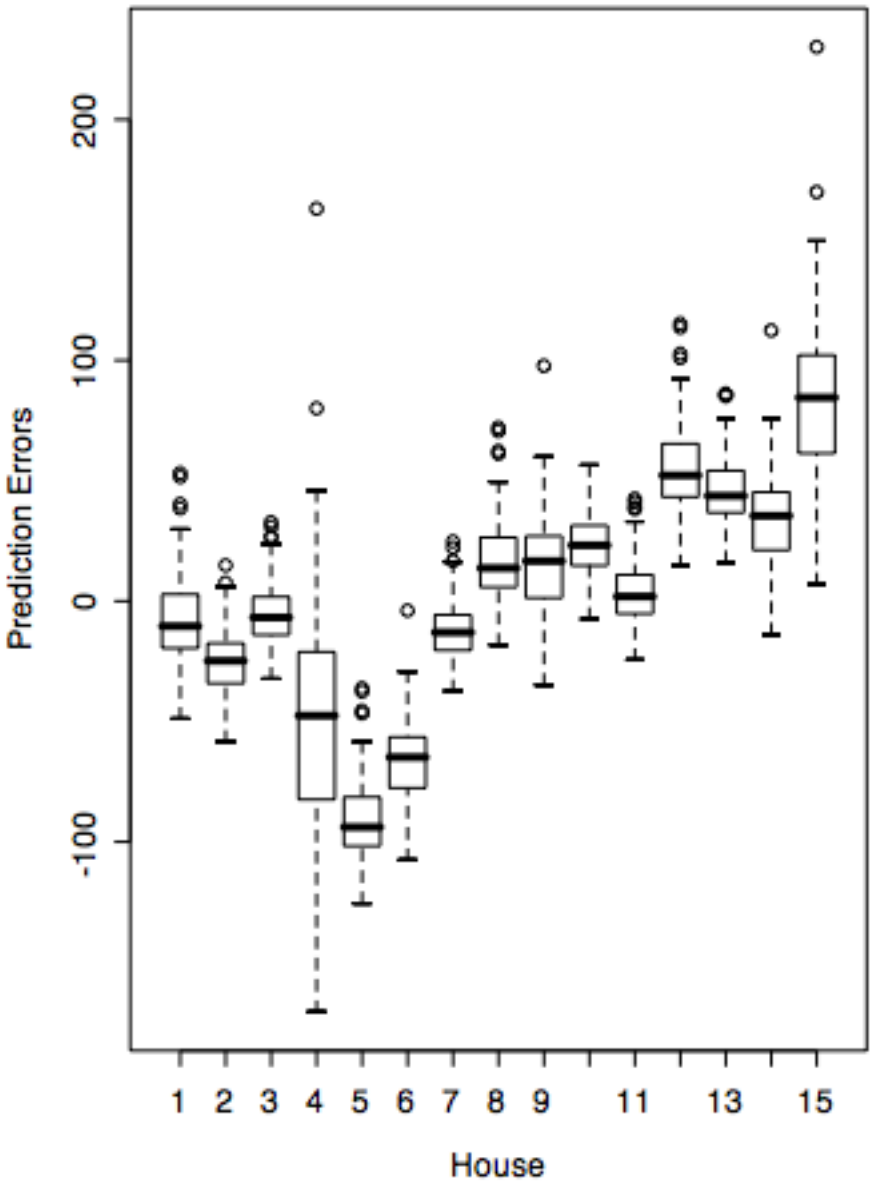

sale.price
150
200
250
300
350
700
800
900
1000
1100
1200
1300
area

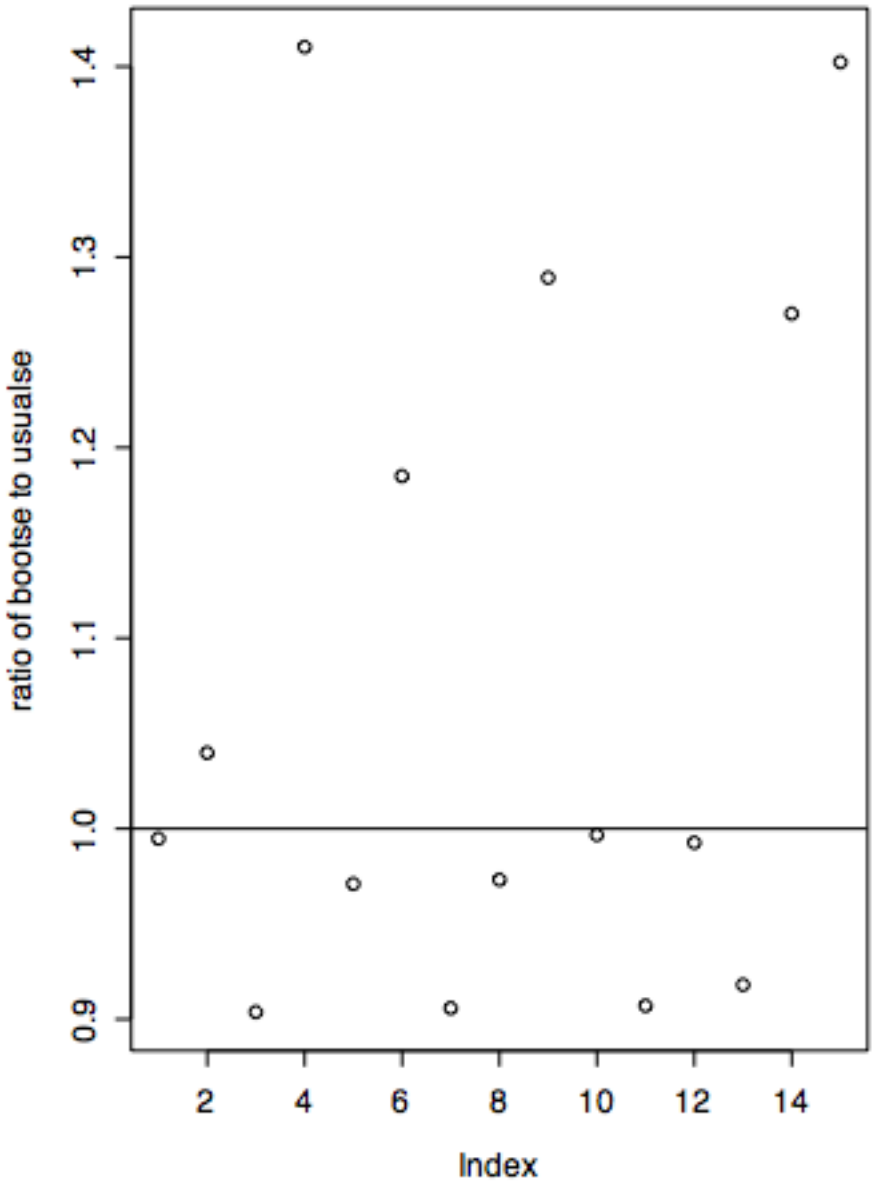
```
> houseprices.boot

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = houseprices, statistic = houseprices.fn, R = 99)


Bootstrap Statistics :
     original     bias    std. error
t1* 0.1877769 0.01020854  0.07837381



> summary(houseprices.lm)
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  70.7504    60.3477   1.172   0.2621
area          0.1878     0.0664   2.828   0.0142 *
```

```
houseprices.fn <- function(houseprices, index) {
 house.resample <- houseprices[index,]
 house.lm <- lm(sale.price ~ area, data=house.resample)
 predict(house.lm, newdata=data.frame(area=1200))
}

houseprices.boot <- boot(houseprices, R=99, statistic=houseprices.fn)

boot.ci(houseprices.boot, type="perc")
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 99 bootstrap replicates

CALL :
boot.ci(boot.out = houseprices.boot, type = "perc")

Intervals :
Level     Percentile
95%   (247.4, 363.1 )
Calculations and Intervals on Original Scale
Some percentile intervals may be unstable
```

```
houseprices2.fn <- function(houseprices, index) {
 house.resample <- houseprices[index,]
 house.lm <- lm(sale.price ~ area, data=house.resample)
 houseprices$sale.price - predict(house.lm,houseprices)
}

R<-200
houseprices2.boot <- boot(houseprices, R=R, statistic=houseprices2.fn)
par(mfrow=c(1,2))
n <- length(houseprices$area)
house.fac <- factor(rep(1:n,rep(R,n)))
plot(house.fac,as.vector(houseprices2.boot$t),ylab="Prediction Errors",
xlab="House")

bootse <- apply(houseprices2.boot$t,2,sd)
usualse <- predict(houseprices.lm,houseprices, se.fit=TRUE)$se.fit
plot(bootse/usualse,ylab="ratio of bootse to usualse")
abline(h=1)
```


Prediction Errors
-100
0
100
200
1 2 3 4 5 6 7 8 9 11 13 15
House


ratio of bootse to usualse
0.9
1.0
1.1
1.2
1.3
1.4
2 4 6 8 10 12 14
Index