lab 2

2025-10-13

1 Examples

First, download “al4” package.

```
options(repos = c(CRAN = "https://cloud.r-project.org/"))
install.packages("devtools")
```

```
## package 'devtools' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  D:\Temp\RtmpOsMlyF\downloaded_packages
```

```
library(devtools)
```

```
## Loading required package: usethis
```

```
devtools::install_github("cran/alr4")
```

```
## WARNING: Rtools is required to build R packages, but is not currently installed.
##
## Please download and install Rtools 4.5 from https://cran.r-project.org/bin/windows/Rtools/.
```

```
## Skipping install of 'alr4' from a github remote, the SHA1 (2250c241) has not changed since last insta
##   Use 'force = TRUE' to force installation
```

Example 1

“Heights” is a dataset consisting of height of mother “Mheight” and height of daughter “Dheight”. Assume the linear model:$dheight = a + b \times mheight + \epsilon, \epsilon \sim (0, \sigma^2)$.

```
install.packages("alr4")
```

```
## package 'alr4' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  D:\Temp\RtmpOsMlyF\downloaded_packages
```

```
library(alr4)
```

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

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

```
## Loading required package: effects
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
myfit=lm(dheight~mheight,data=Heights)
summary(myfit)
```

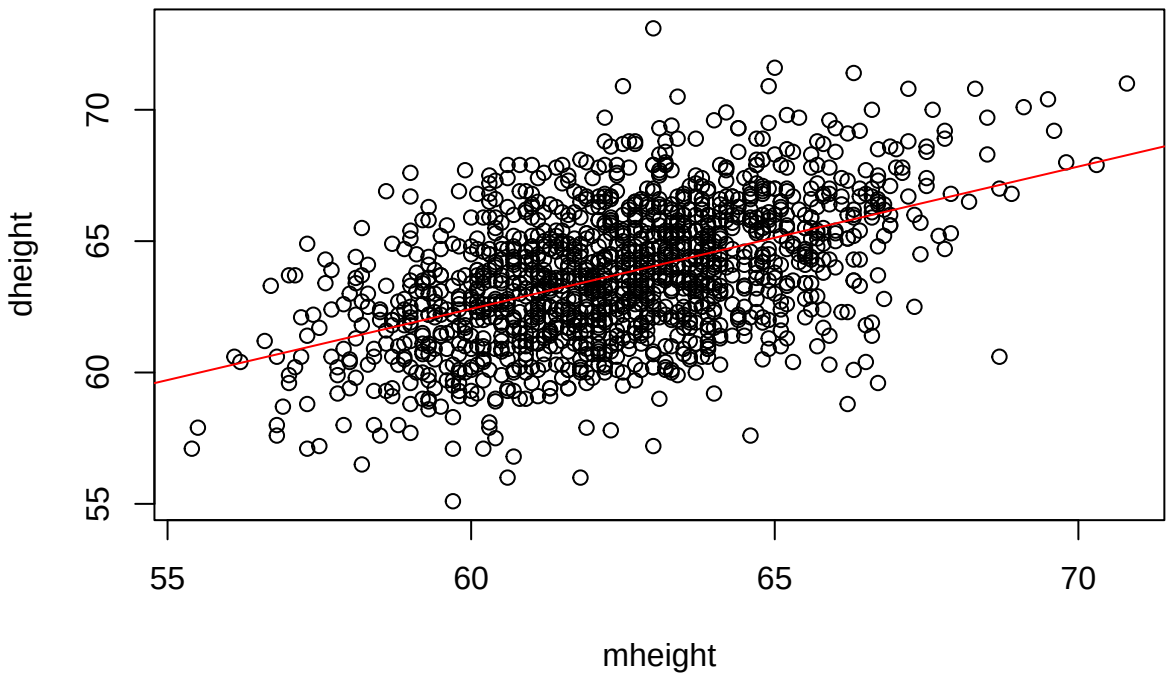
```
##
## Call:
## lm(formula = dheight ~ mheight, data = Heights)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.397 -1.529  0.036  1.492  9.053
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.91744    1.62247   18.44   <2e-16 ***
## mheight      0.54175    0.02596   20.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.266 on 1373 degrees of freedom
## Multiple R-squared:  0.2408, Adjusted R-squared:  0.2402
## F-statistic: 435.5 on 1 and 1373 DF,  p-value: < 2.2e-16
```

```
coefficients(myfit)
```

```
## (Intercept)     mheight
##   29.917437    0.541747
```

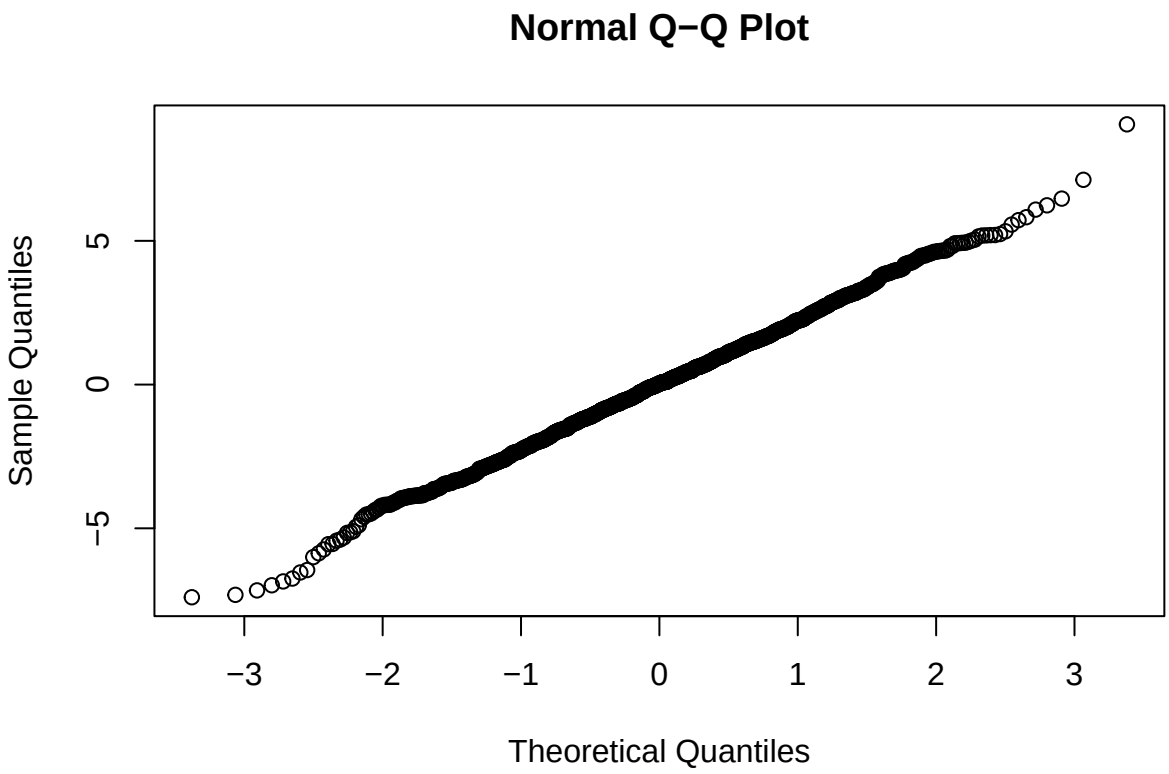
Now we use scatter plot to check:

```
plot(Heights)
abline(myfit,col="red")
```

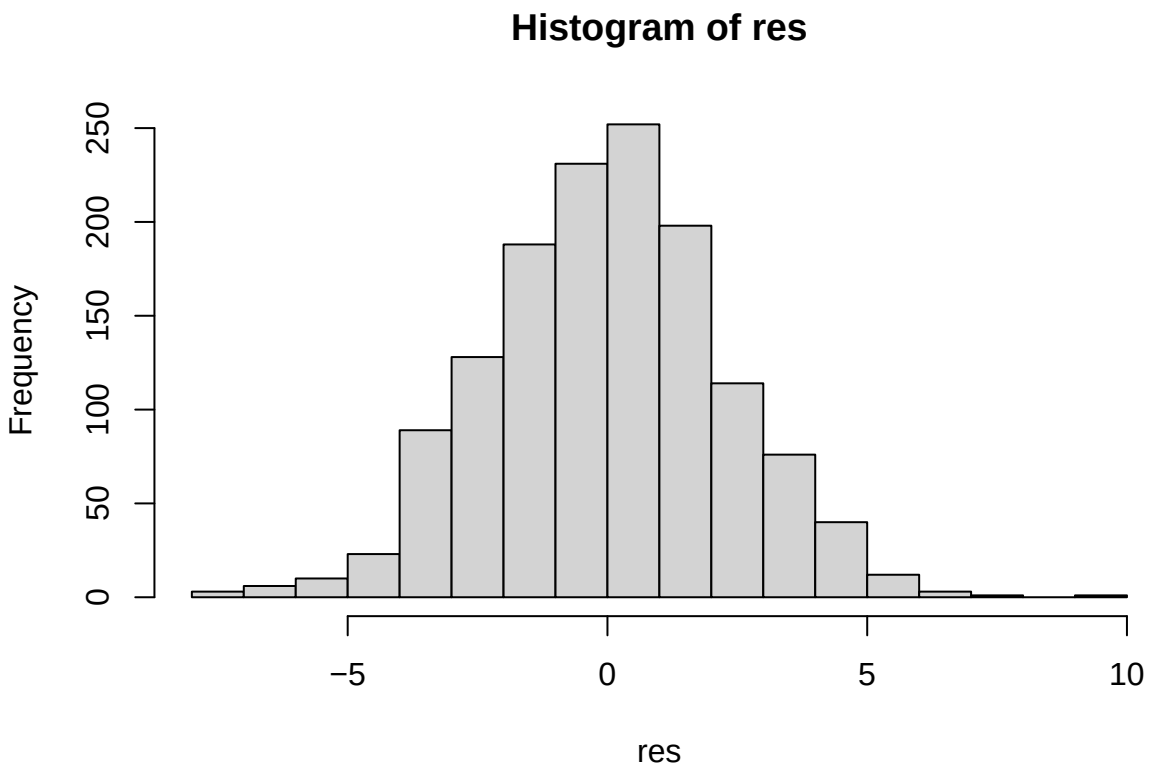


To see whether the model fits the data well, we check whether the residuals follow Gaussian distribution, and if there is a nonlinear relation between residuals and independent variables.

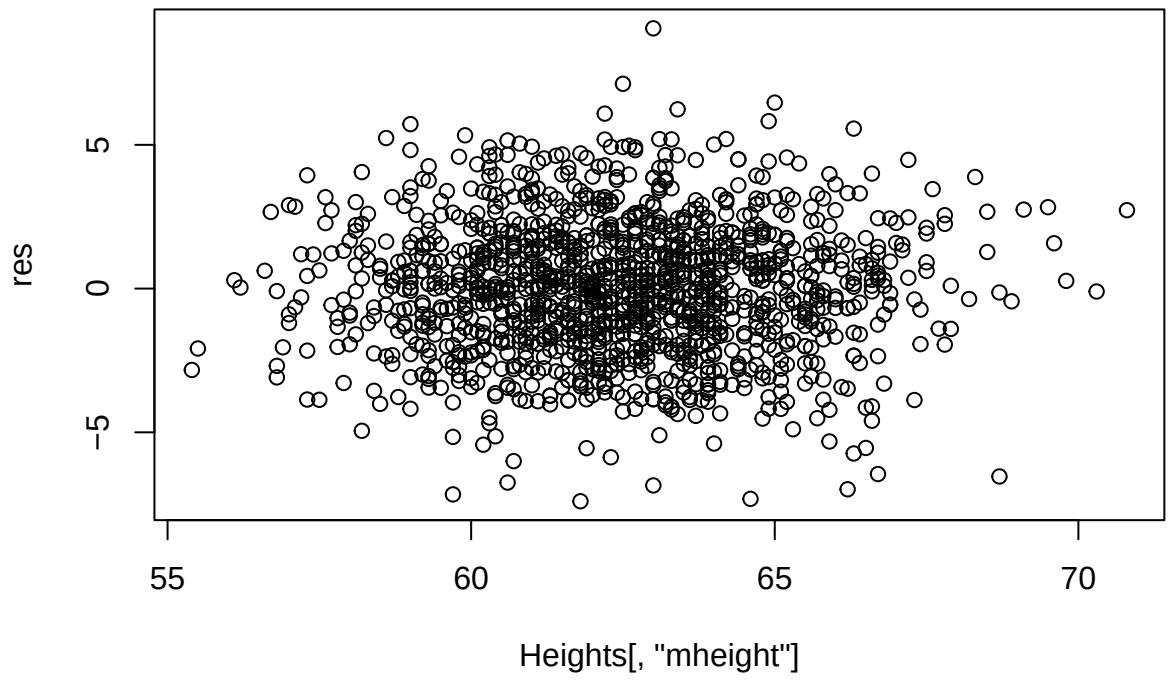
```
res=residuals(myfit)
qqnorm(res)#check if the residuals are normally distributed
```


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
−5
0
5
−3
−2
−1
0
1
2
3

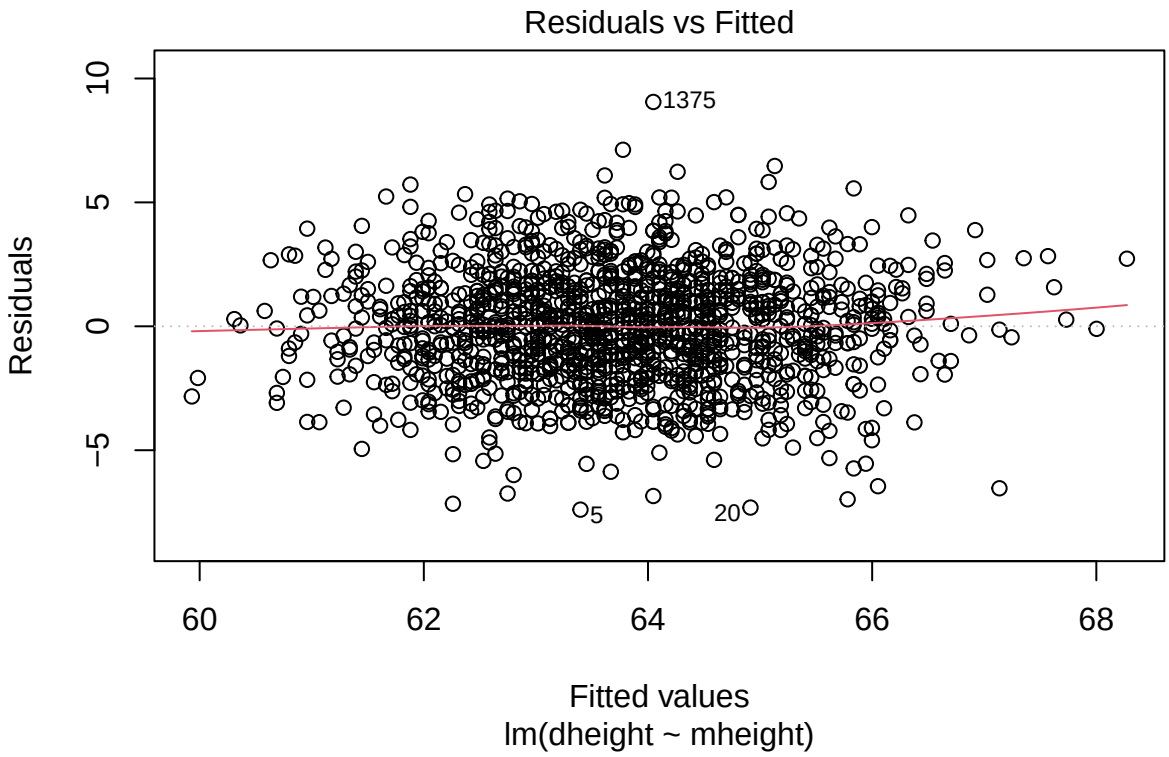
```
hist(res)
```

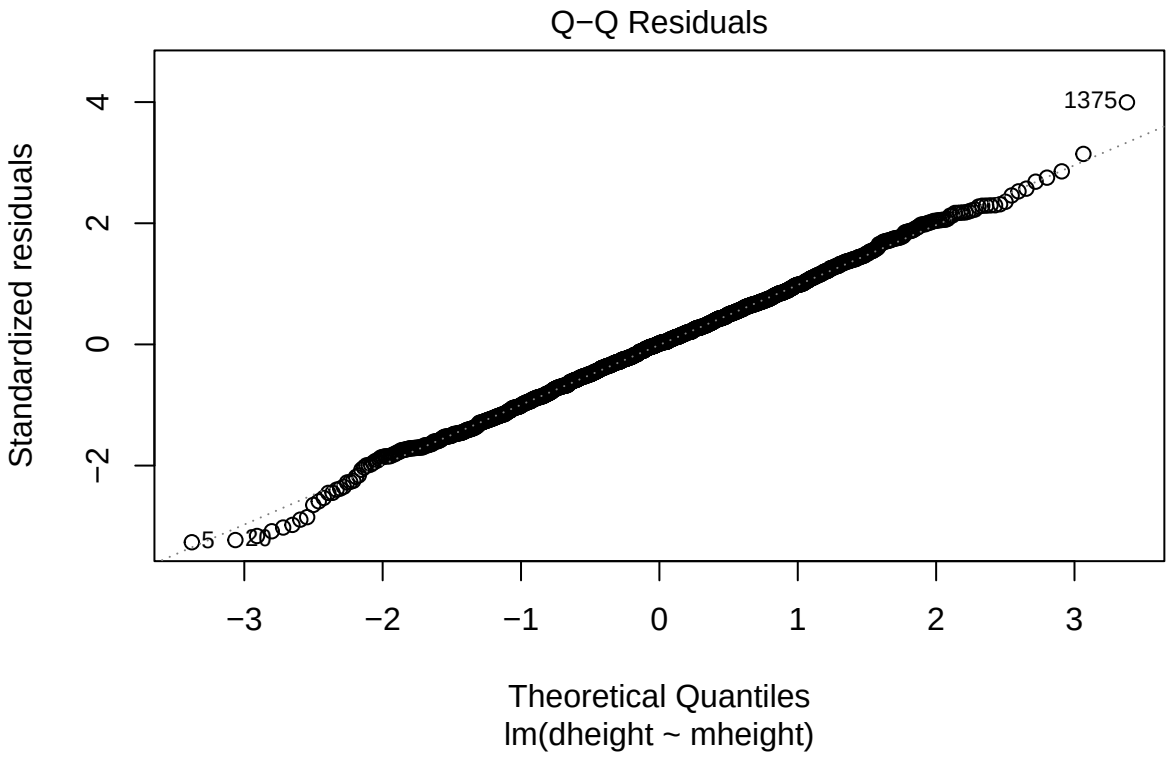


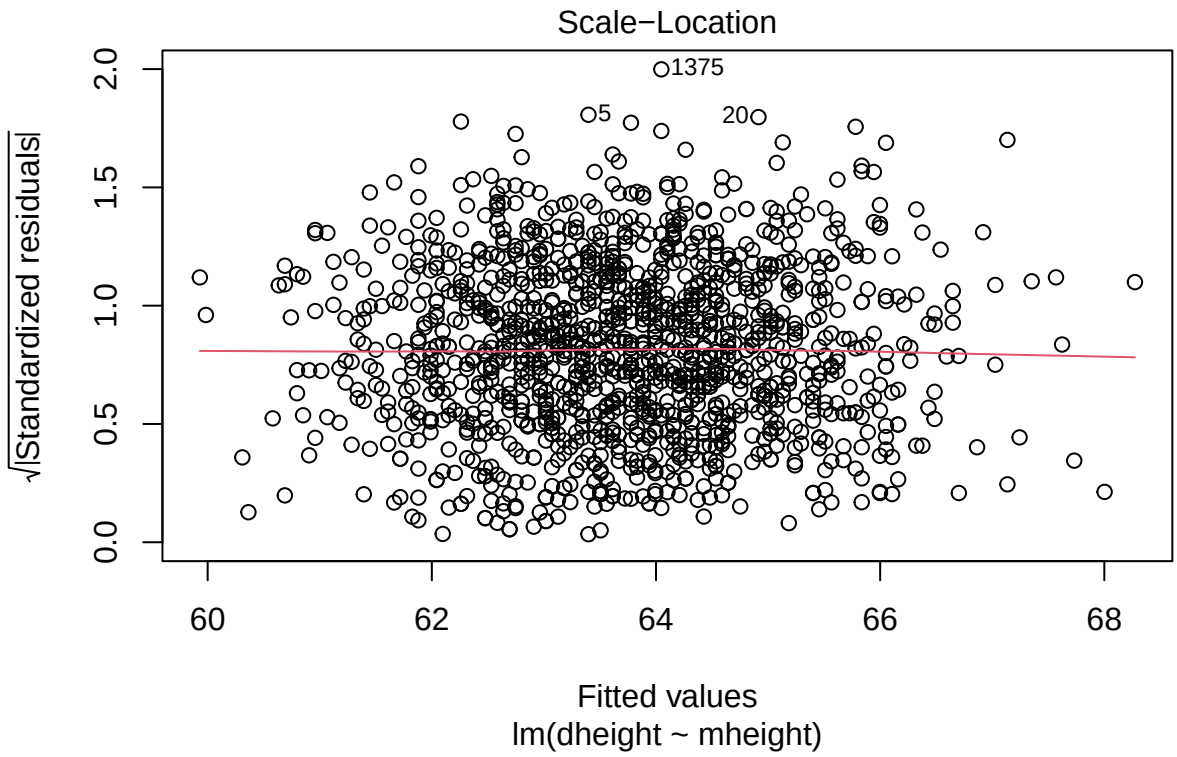

```
plot(Heights[,"mheight"],res)
```

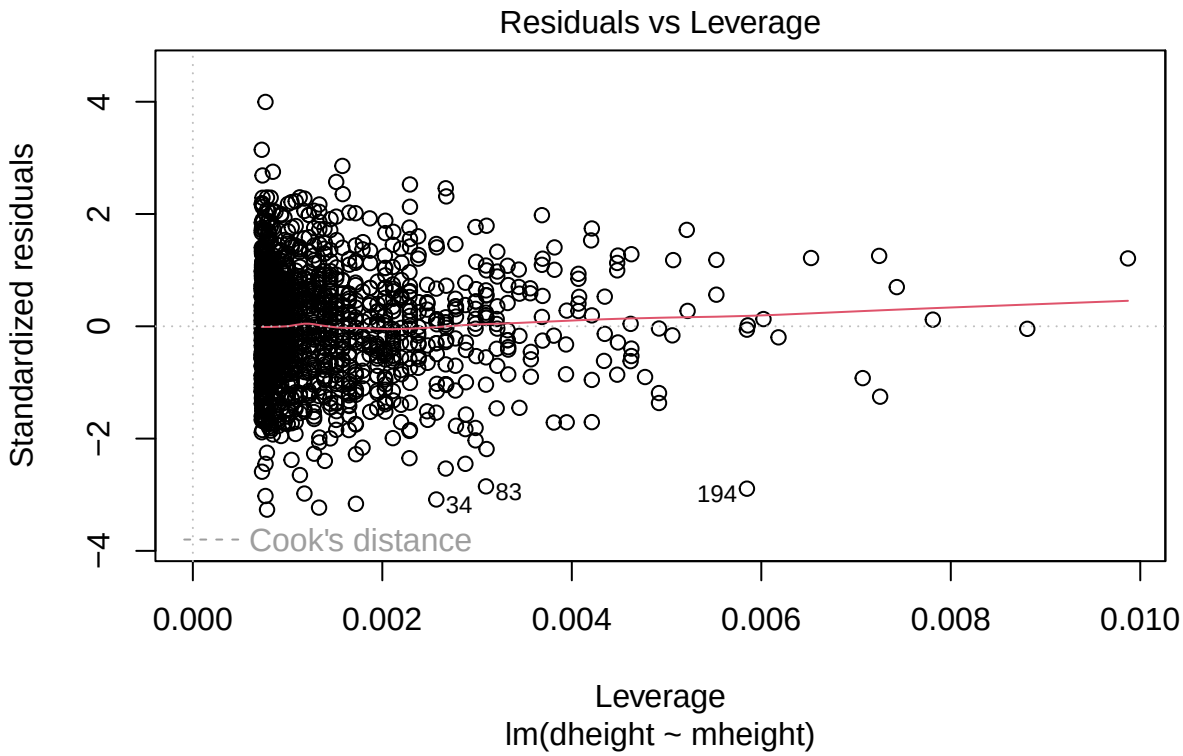



```
plot(myfit)
```


Residuals vs Fitted
Residuals
10
5
0
−5
1375
5
20
60
62
64
66
68
Fitted values
lm(dheight ~ mheight)


Q−Q Residuals
Standardized residuals
4
2
0
−2
1375
5
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(dheight ~ mheight)

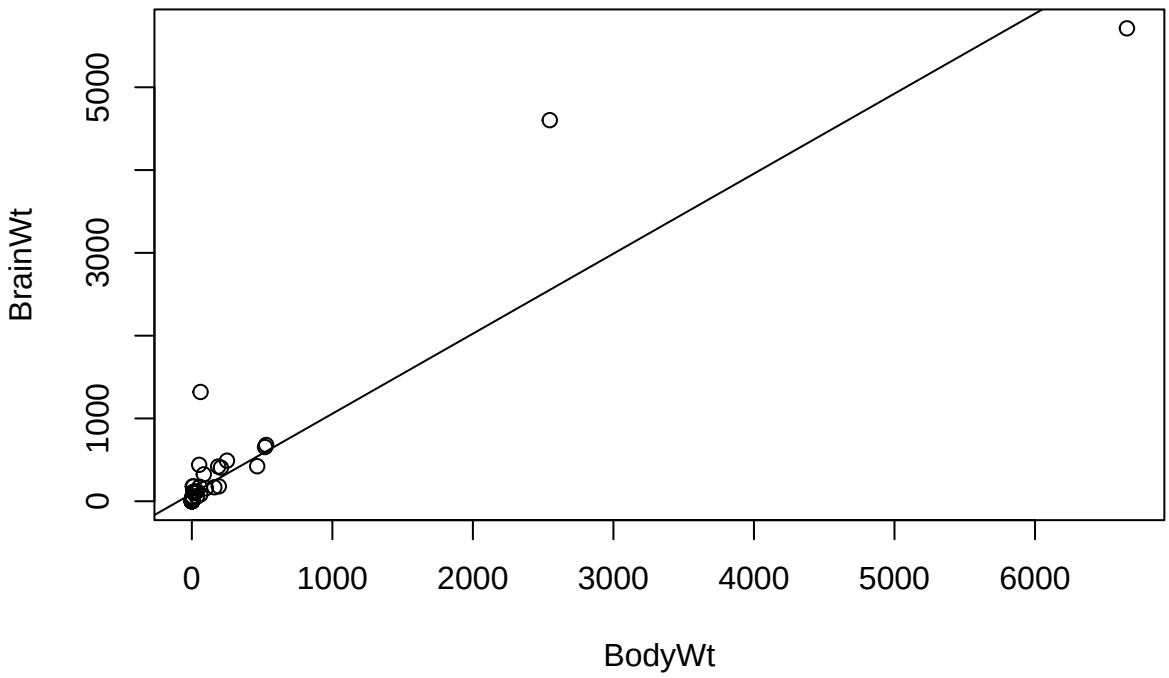

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
1375
5
20
60
62
64
66
68
Fitted values
lm(dheight ~ mheight)



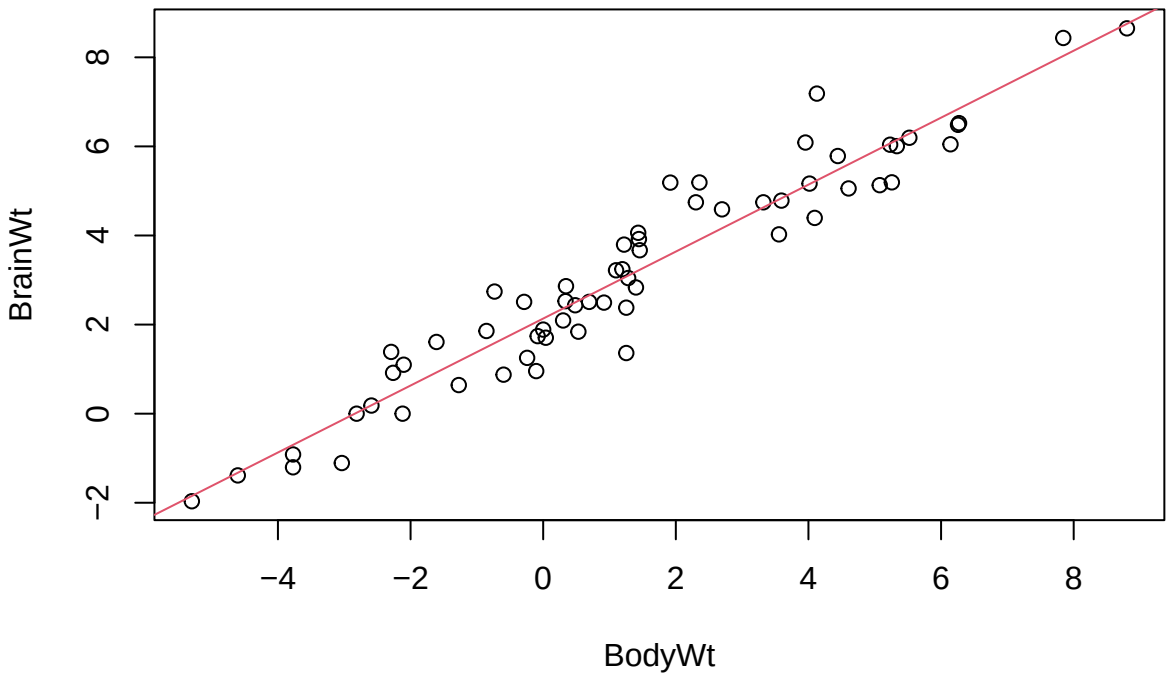
Example 2

Dataset "brains" provides brainweight(independent variable) and weights(response) of various animals.

```
#simple linear model
plot(brains[,2:1])
a=lm(BrainWt~BodyWt,data=brains)
abline(a)
```




```
#fit on log
plot(log(brains[,2:1]))
b=lm(BrainWt~BodyWt,data=log(brains))
abline(b,col=2)
```



We can see that fitting on log(variable) yields a better fit.

Now we predict the brain weight of an animal whose weight is 4kg.

```
x=as.numeric(coefficients(a)[1])
y=as.numeric(coefficients(a)[2])
x_log=as.numeric(coefficients(b)[1])
y_log=as.numeric(coefficients(b)[2])
bw=4
prediction=x+y*bw
print(prediction)
```

```
## [1] 94.87448
```

```
prediction_log=exp(x_log)*bw^y_log
print(prediction_log)
```

```
## [1] 23.97104
```

```
brains[brains[,2]<4.5&brains[,2]>3.5,]
```

```
##                       BrainWt BodyWt
## Vervet                 57.998  4.190
## Yellow-bellied marmot  17.000  4.050
## Rock hyrax2            21.000  3.600
## Raccoon                39.201  4.288
## Red fox                50.400  4.235
```

Apparently, the log prediction is closer to true data.

2 Exercise

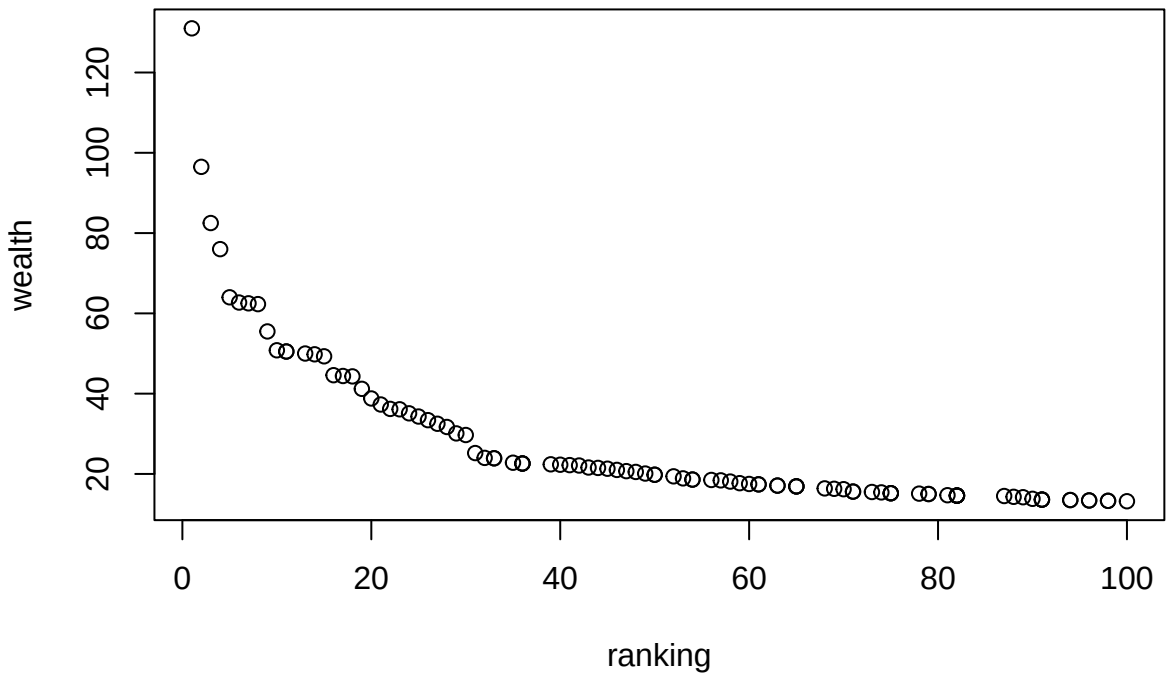
2.1

We would like to find the relation between ranking and wealth(billion).

```
install.packages("readxl")
```

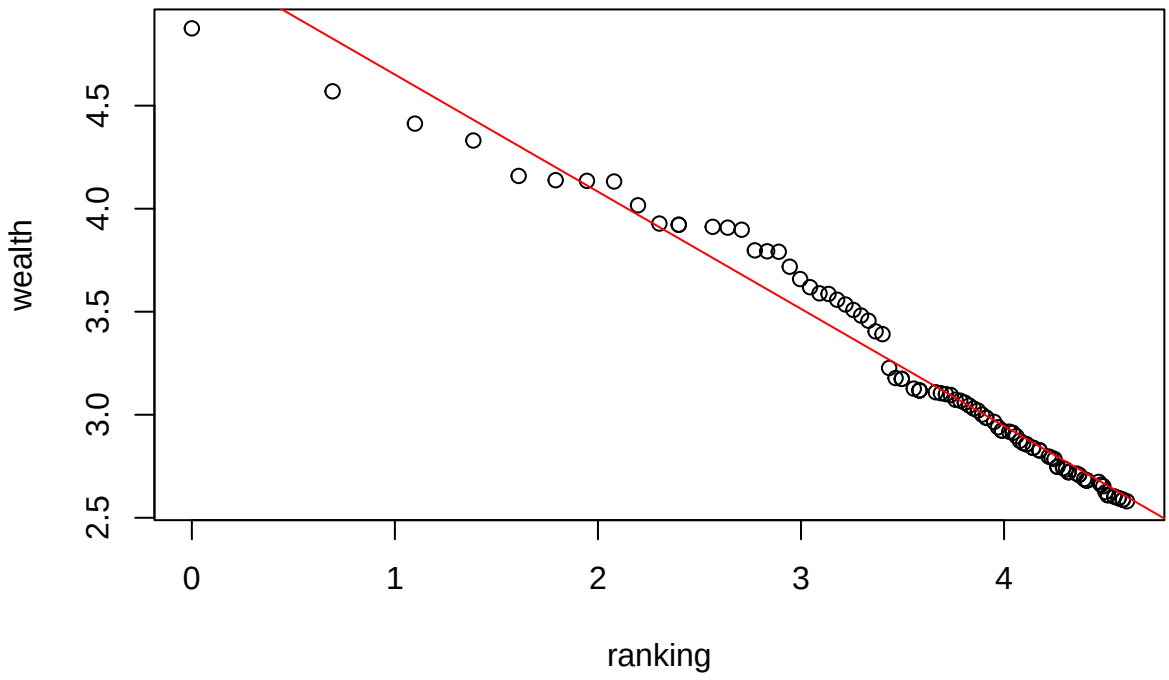
```
## package 'readxl' successfully unpacked and MD5 sums checked

## Warning: cannot remove prior installation of package 'readxl'

## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## D:\360Downloads\Software\R-4.5.1\library\00LOCK\readxl\libs\x64\readxl.dll to
## D:\360Downloads\Software\R-4.5.1\library\readxl\libs\x64\readxl.dll: Permission
## denied

## Warning: restored 'readxl'

##
## The downloaded binary packages are in
##  D:\Temp\RtmpOsMlyF\downloaded_packages
```

```
library(readxl)
data=read_excel("forbes2019.xlsx")
analysis <- data.frame(
  ranking= data[[1]],
  wealth= data[[4]]
)
plot(analysis)
```



From the plot, we can see that wealth and ranking cannot fit linear relation well. Therefore, we can first take log.

```
fit=lm(wealth~ranking,data=log(analysis))
plot(log(analysis))
abline(fit,col="red")
```




```
summary(fit)
```

```
##
## Call:
## lm(formula = wealth ~ ranking, data = log(analysis))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34423 -0.03443 -0.02314  0.00228  0.21810
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.219425   0.036374  143.49   <2e-16 ***
## ranking     -0.568528   0.009708  -58.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08949 on 98 degrees of freedom
## Multiple R-squared:  0.9722, Adjusted R-squared:  0.9719
## F-statistic:  3430 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
coefficients(fit)
```

```
## (Intercept)     ranking
##   5.2194250  -0.5685282
```

Eventually, we can fit the data by $wealth = e^{5.22} ranking^{-0.57}$.

2.2

We would like to set up a standard for obesity.

(a)First, let's find the LS estimate for a,b and σ.

```
data <- read.table("height-weight.txt", header = TRUE)
df<-data.frame(
  sex=data[[1]],
  weight=data[[2]],
  height=data[[3]]
)

result <- lm(weight ~ height, data = log(df))

coef(result)
```

```
## (Intercept)      height
##    2.598779    2.929655
```

```
summary(result)
```

```
##
## Call:
## lm(formula = weight ~ height, data = log(df))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.27563 -0.07883 -0.00480  0.07677  0.45833
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.59878    0.08843   29.39   <2e-16 ***
## height       2.92966    0.16521   17.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1216 on 197 degrees of freedom
## Multiple R-squared:  0.6148, Adjusted R-squared:  0.6129
## F-statistic: 314.4 on 1 and 197 DF,  p-value: < 2.2e-16
```

$\hat{a} = 2.598779, \hat{b} = 2.929655, \hat{sigma} = 0.1216$.

```
a_hat=as.numeric(coefficients(result)[1])
b_hat=as.numeric(coefficients(result)[2])
sigma_hat=summary(result)$sigma
epsilon<-function(a_hat,b_hat,sigma_hat,W,H){
  return((log(W)-a_hat-b_hat*log(H))/sigma_hat)
}
is_obesity<-function(coefficient){
```

```
  return ((coefficient)>1.645)
}
#check my weight
W=73
H=1.78
my_coefficient=epsilon(a_hat=a_hat,b_hat=b_hat,sigma_hat=sigma_hat,W=W,H=H)
print(my_coefficient)
```

```
## [1] 0.01974951
```

```
print(is_obesity(my_coefficient))
```

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

```
#check the proportion whose coefficient is larger than mine
proportion=pnorm(my_coefficient, lower.tail = FALSE)
print(proportion)
```

```
## [1] 0.4921216
```

My weight coefficient is about 0.02, and I have not exceeded the standard. There are around 50% people whose coefficients exceed me.

(b)If we directly regress on the data without taking log:

```
result2=lm(weight~height,data=df)
coef(result2)
```

```
## (Intercept)      height
##   -130.7470    114.9222
```

```
summary(result2)
```

```
##
## Call:
## lm(formula = weight ~ height, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.650  -5.419  -0.576   4.857  42.887
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -130.747     11.563  -11.31   <2e-16 ***
## height       114.922      6.769   16.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.523 on 197 degrees of freedom
## Multiple R-squared:  0.594,  Adjusted R-squared:  0.592
## F-statistic: 288.3 on 1 and 197 DF,  p-value: < 2.2e-16
```

$\hat{a} = -130.7470, \hat{b} = 114.9222, \hat{\sigma} = 8.523$

```
a_hat2=as.numeric(coef(result2)[1])
b_hat2=as.numeric(coef(result2)[2])
sigma_hat2=summary(result2)$sigma
eta<-function(a,b,sigma,H,W){
  return((W-a-b*H)/sigma)
}
is_obesity2<-function(coefficient){
  return (coefficient>1.645)
}
#recheck my weight
W=73
H=1.78
my_coefficient2=eta(a=a_hat2,b=b_hat2,sigma=sigma_hat2,H=H,W=W)
print(my_coefficient2)
```

```
## [1] -0.09556973
```

```
print(is_obesity2(my_coefficient2))
```

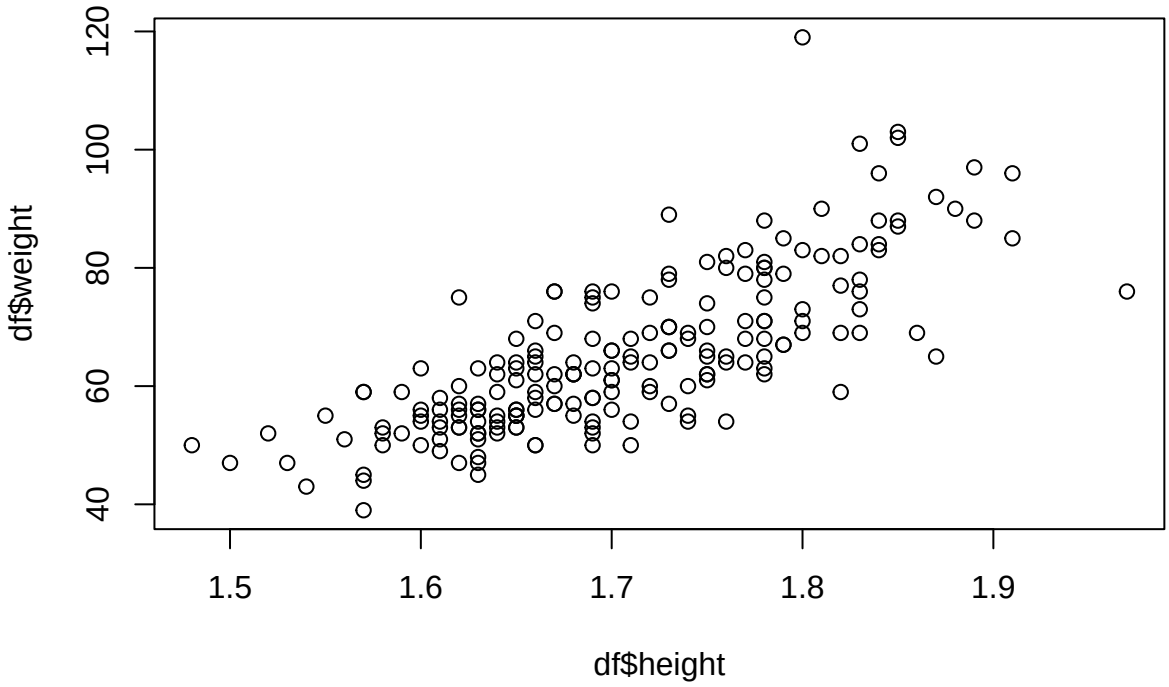
```
## [1] FALSE
```

```
proportion=pnorm(my_coefficient2, lower.tail = FALSE)
print(proportion)
```

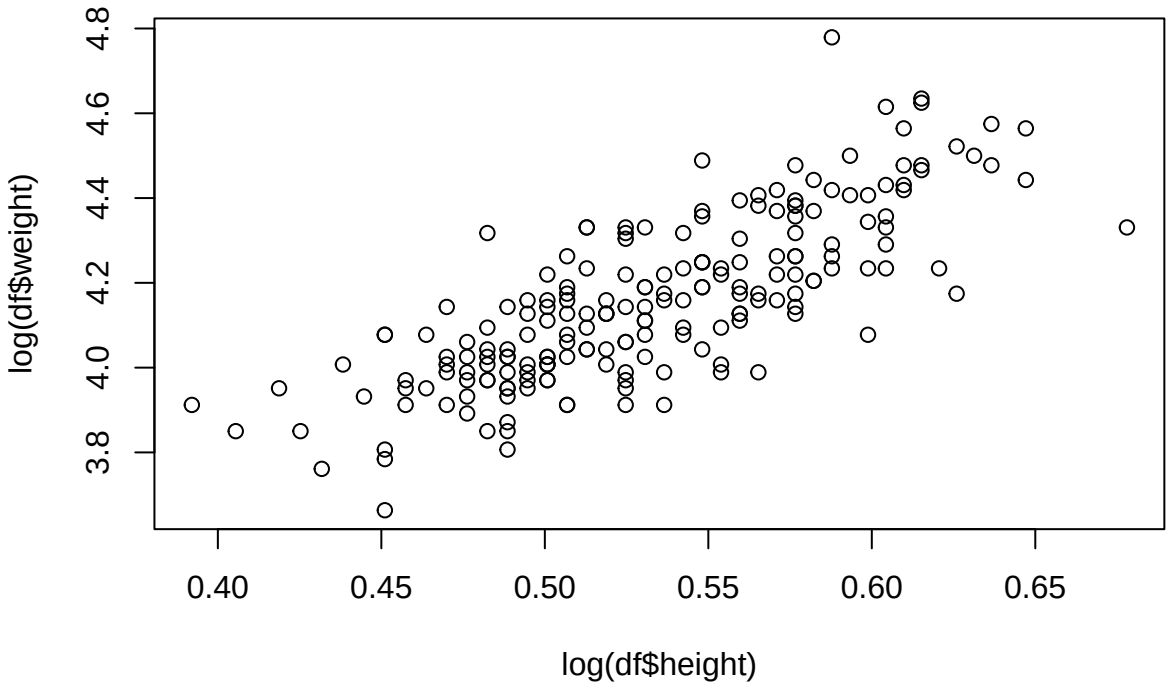
```
## [1] 0.5380688
```

The results from (a) and (b) are close. To see which one is more reasonable, we can check the plot of weight and height to see whether they follow a linear relation:

```
plot(df$height,df$weight)
```

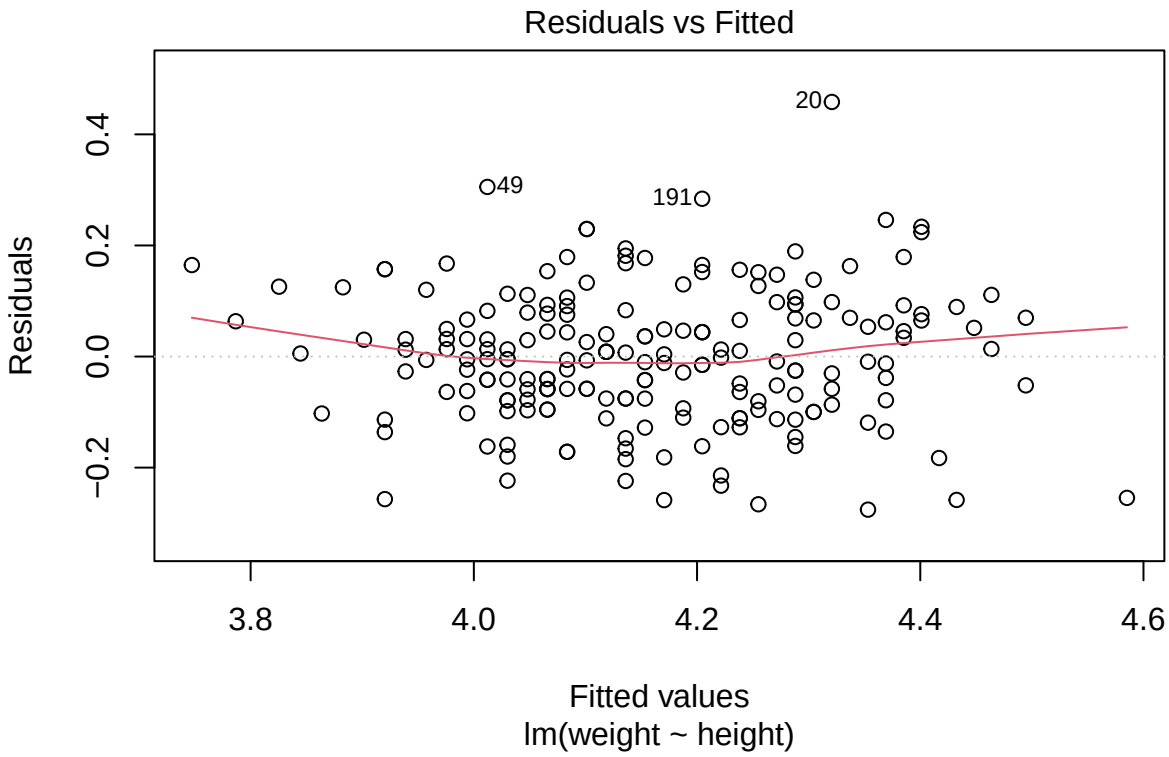



```
plot(log(df$height),log(df$weight))
```

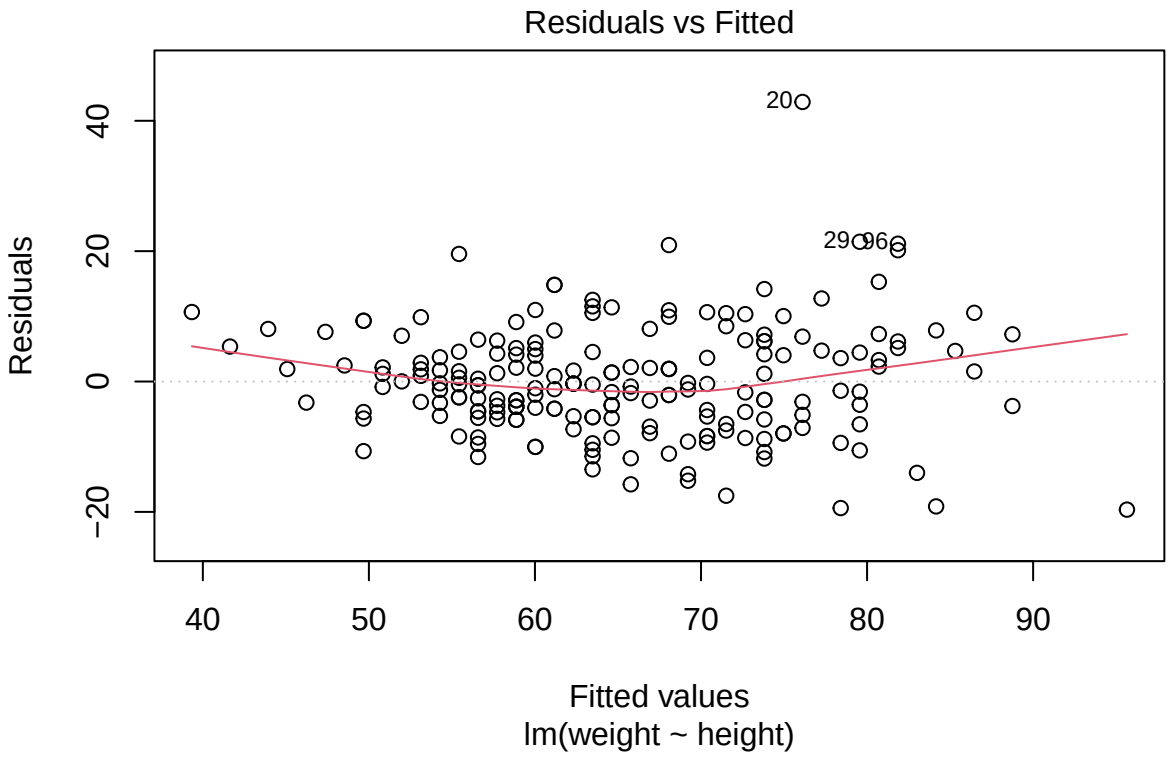


Still, the scatter plots look similar. Let's inspect the residuals of each regression:

```
#log regression
plot(result,which=1)
```




```
#simple regression
plot(result2,which=1)
```



Unfortunately, it is still hard to distinguish which one is significantly better. However, since the R^2 of log linear regression is larger than simple linear regression, we should conclude that (a)log linear regression is slightly better.

Theoretically, using log linear regression is more reasonable, since weight and height ought to follow power law. Nevertheless, the reason why its performance is not significantly better than linear regression is probably that the range of height in the data is narrow.

(c)Now we add the factor of sex into the regresssion:

```
result3=lm(log(weight)~log(height)+sex,data=df)
coef(result3)
```

```
## (Intercept) log(height)         sex
##   3.0087053   2.0571556   0.1240784
```

We can see that $\hat{b} \approx 2$.

2.3

We would like to check Benford law by using Monte Carlo method.

```
set.seed(666)
#generate n samples
n=1000000
```

```r
z=rnorm(n)
x=exp(z)
#find the first nonzero digit
transform<-function(x){
  return (x/10^floor(log10(x)))
}#transform the number such that the integer part is one digit
first_nonzero_digit <- function(x) {
  if (x == 0) return(0)


  transformed <- transform(x)

  first_digit <- floor(transformed)

  return(first_digit)
}
#result
first_digits <- sapply(x, first_nonzero_digit)
table(first_digits)/n
```

```
## first_digits
##        1        2        3        4        5        6        7        8
## 0.307945 0.170555 0.118611 0.093803 0.079311 0.068227 0.059750 0.053604
##        9
## 0.048194
```

```r
i<-c(1,2,3,4,5,6,7,8,9)
Benford<-log10(1+1/i)
Benford
```

```
## [1] 0.30103000 0.17609126 0.12493874 0.09691001 0.07918125 0.06694679 0.05799195
## [8] 0.05115252 0.04575749
```

We can see that with n=1000000 samples, the simulation is very close to Benford law.