Pstat126 hw3

Haocheng Zhang

2023-03-11

```
## Show the table with the given command.
Cereal <- read.csv("cereal.csv",header=T)

str(Cereal)
```

```
## 'data.frame':    77 obs. of  17 variables:
##  $ X       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ name    : chr  "100% Bran" "100% Natural Bran" "All-Bran" "All-Bran with Extra Fiber" ...
##  $ manuf   : chr  "N" "Q" "K" "K" ...
##  $ type    : chr  "cold" "cold" "cold" "cold" ...
##  $ calories: int  70 120 70 50 110 110 110 130 90 90 ...
##  $ protein : int  4 3 4 4 2 2 2 3 2 3 ...
##  $ fat     : int  1 5 1 0 2 2 0 2 1 0 ...
##  $ sodium  : int  130 15 260 140 200 180 125 210 200 210 ...
##  $ fiber   : num  10 2 9 14 1 1.5 1 2 4 5 ...
##  $ carbo   : num  5 8 7 8 14 10.5 11 18 15 13 ...
##  $ sugars  : int  6 8 5 0 8 10 14 8 6 5 ...
##  $ potass  : int  280 135 320 330 -1 70 30 100 125 190 ...
##  $ vitamins: int  25 0 25 25 25 25 25 25 25 25 ...
##  $ shelf   : int  3 3 3 3 3 1 2 3 1 3 ...
##  $ weight  : num  1 1 1 1 1 1 1 1.33 1 1 ...
##  $ cups    : num  0.33 1 0.33 0.5 0.75 0.75 1 0.75 0.67 0.67 ...
##  $ rating  : num  68.4 34 59.4 93.7 34.4 ...
```

```
## Since we are required to focus on 9 varaibles, to make it easier,
##I'll make a new dataset named 'new_cerel' with the selected 9 variables only.
new_cereal <- Cereal[, c("rating", "protein", "fat", "fiber", "carbo",
                         "sugars", "potass", "vitamins", "cups")]
```

```
## (a) FIrstly, remove the observations 5, 21, and 58 as requird.
new_cereal <- new_cereal[-c(5, 21, 58),]


# Secondly, we can run a multiple linear regression model using the lm() function
model <- lm(rating ~ protein + fat + fiber + carbo + sugars + potass +
              vitamins + cups, data = new_cereal)


# Then, we calculate fitted response values and the residuals
fitted_values <- fitted(model)
```

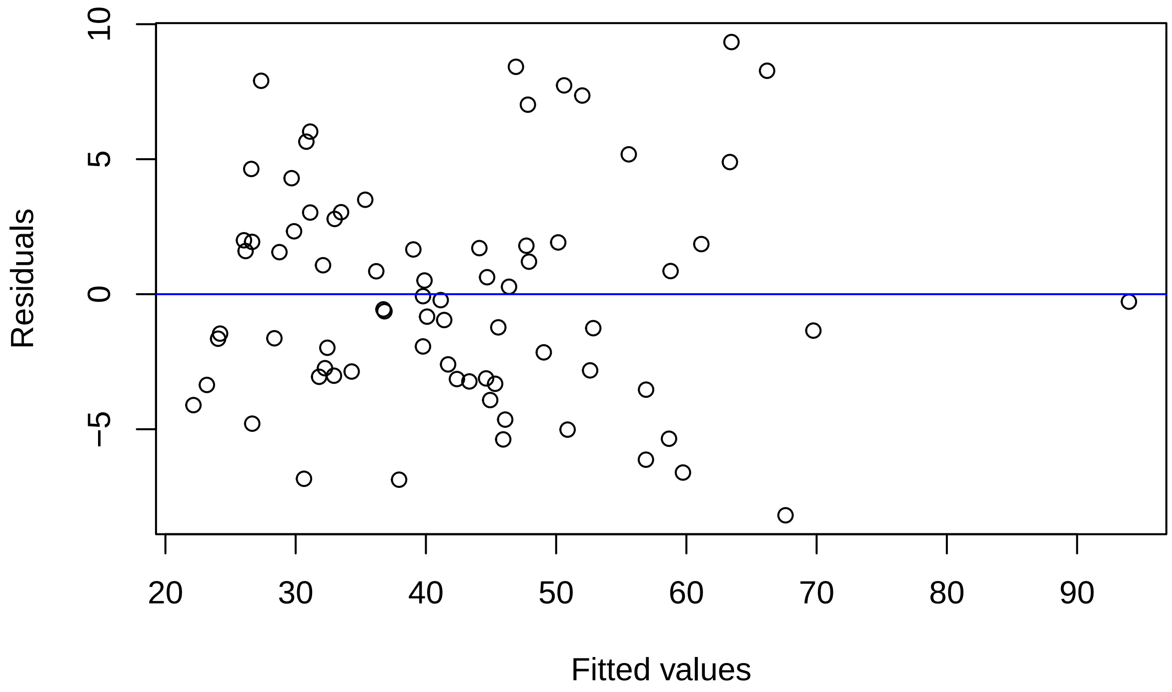
```
residuals <- resid(model)

# Finally, show the first 5 entries using head() function.
head(fitted_values, 5)
```

```
##        1        2        3        4        6
## 69.75066 29.68772 67.61235 93.98080 32.24978
```

```
head(residuals, 5)
```

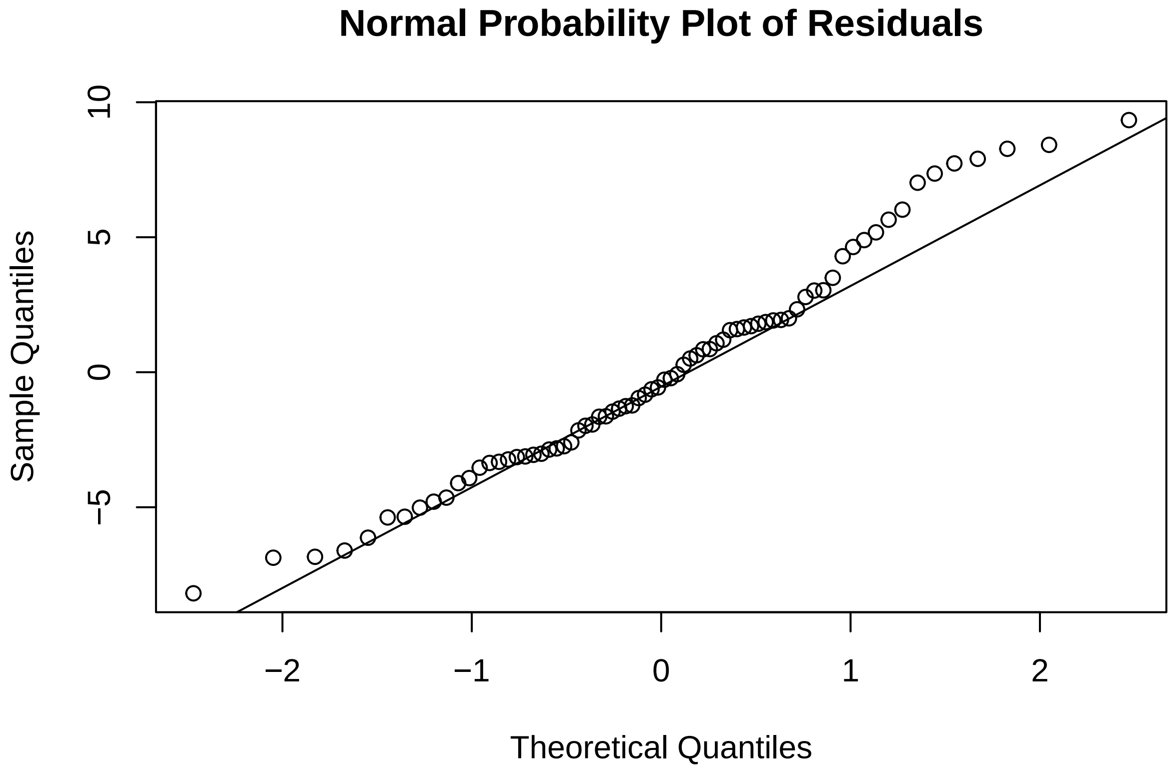
```
##          1          2          3          4          6
## -1.3476910  4.2959597 -8.1868456 -0.2758917 -2.7402368
```

```
## (b)We can use a plot of residuals against fitted values,

plot(model$fitted.values, model$residuals, xlab = "Fitted values", ylab = "Residuals")
abline(h = 0, col = "blue")
```



```
## From the plot above, there is not a clear pattern in the residuals and fitted values
##and the points are randomly scattered around the horizontal line at 0.
##This suggests that the variance of the errors is approximately constant across
##the range of fitted values, and the assumption of constant variance
##is reasonable for this model.
```

```r
## (c)To check if the random errors follow a normal distribution, we can use a normal probability plot.
qqnorm(model$residuals, main="Normal Probability Plot of Residuals")

qqline(model$residuals)
```



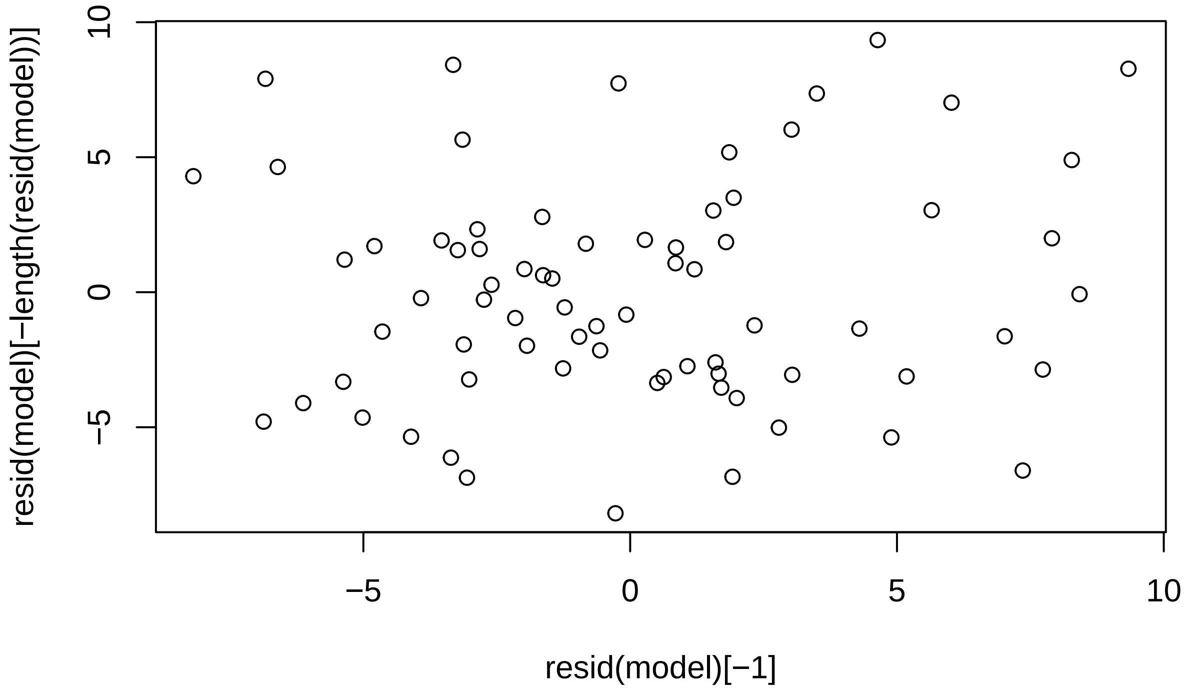
```r
# By observing the line and the plot above, it appears that the residuals are roughly
##normally distributed. The points on the plot follow a straight line fairly closely
##except for some slight deviation at the tails. Therefore, we can conclude that
##the random errors approximately follow a normal distribution.
```

```r
## (d)To run the Shapiro-Wilk test in R for the residuals of the multiple linear
##regression model, we can use the shapiro.test() function.
##And the null hypothesis of the test is that the population is normally distributed.
shapiro.test(model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.97607, p-value = 0.1728
```

```r
# From the output, the p-value is 0.1728, the p-value is greater than the typical
##significance level of 0.05, we fail to reject the null hypothesis that the residuals
##follow a normal distribution. Therefore, we can conclude that there is no significant
##evidence that the residuals deviate from normality.
```

```
## (e)Plot successive pairs of residuals
plot(resid(model)[-1], resid(model)[-length(resid(model))])
```



```
# As we can see, there is no clear pattern or trend in the plot, suggesting that
##there is no significant serial correlation among the residuals,
##which suggests that there is no serial correlation among the observations.
```

```
## (f)Run the Durbin-Watson test
library(lmtest)
```

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

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

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

```
dwtest(model)
```

```
##
##  Durbin-Watson test
```

```
##
## data:  model
## DW = 1.8414, p-value = 0.2041
## alternative hypothesis: true autocorrelation is greater than 0
```

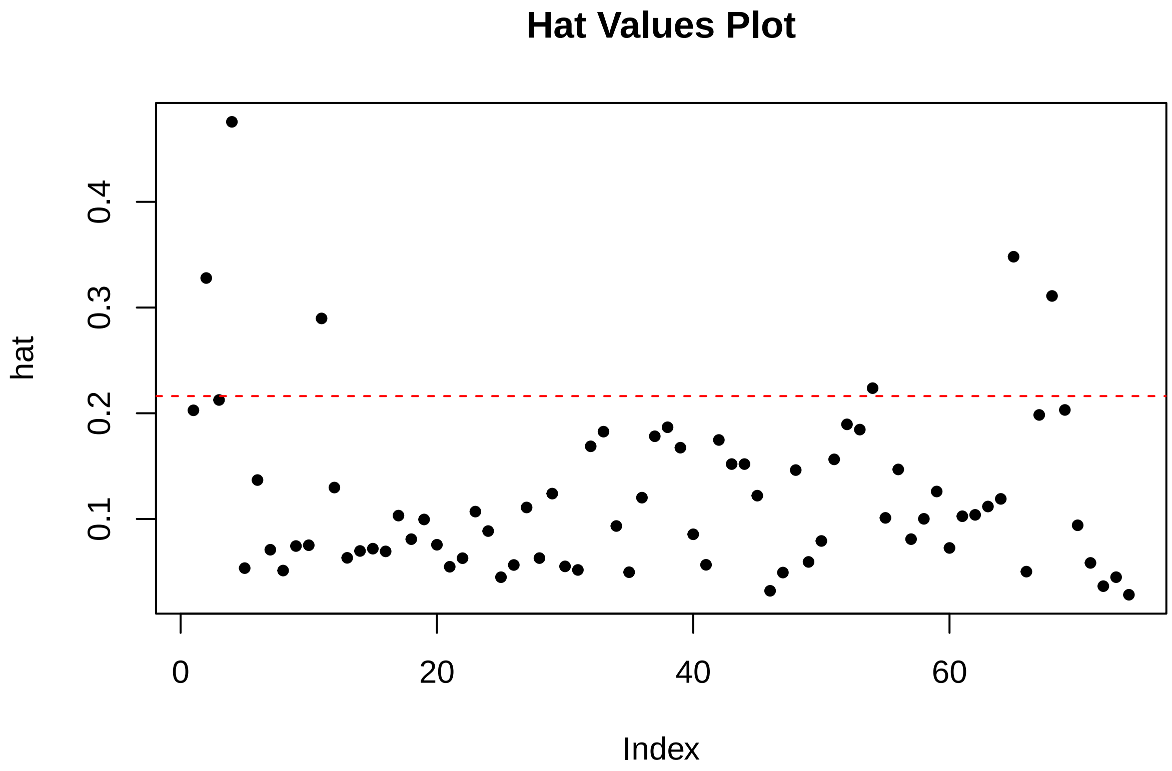
```
# The p-value is 0.2041, indicating that there is no significant evidence of
##autocorrelation in the residuals at the 5% significance level. The null
##hypothesis is that there is no autocorrelation in the residuals,
##and the alternative hypothesis is that there is positive autocorrelation.
##Since the p-value is greater than 0.05, we fail to reject the null hypothesis
##and conclude that there is no evidence of autocorrelation in the residuals.
```

```
## (g) The hat matrix H is defined as H = X(X'X)^(-1)X', so:
X <- model.matrix(model)

H <- X %*% solve(t(X) %*% X) %*% t(X)

# We can sum the diagonal elements of H and compare the result to p + 1:
sum(diag(H))
```

```
## [1] 9
```

```
# The output of the sum is 9, and by hypothesis, the value of p=8, and p+1=9,
##so yes this verifies numerically that the sum of Hii from i=1 to n is
##H_ii = p* = p + 1.
```

```
## (h) The criterion I would use to detect high-leverage points is the hat
##value criterion.
# Calculate hat values
hat <- hatvalues(model)

# Create a plot of hat values
plot(hat, pch = 20, main = "Hat Values Plot")

# Add a horizontal line at the cutoff value
abline(h = 2*8/74, col = "red", lty = 2)
```



```
# By observting the plot and the line, there're a few plots above the line,
##thus we can say yes there are high-leverage points.
```

```
## (i)To compute the standardized residuals, we can use the rstandard() function.
##Then use the criterion to identify outliers that based on standardized residuals
##is that any observation with an absolute standardized residual greater than 3
##may be considered an outlier.

# Compute standardized residuals
std_resid <- rstandard(model)

# Print summary of standardized residuals
summary(std_resid)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.119994 -0.739720 -0.110314  0.001344  0.523617  2.276780
```

```
# By observing the output of the summary above, the minimal value is -2.119994,
##and the max value is 2.276780, and the absolute value of both would be less
##than 3, thus we can say there is no outliers based on the criterion of
##standardized residuals.
```

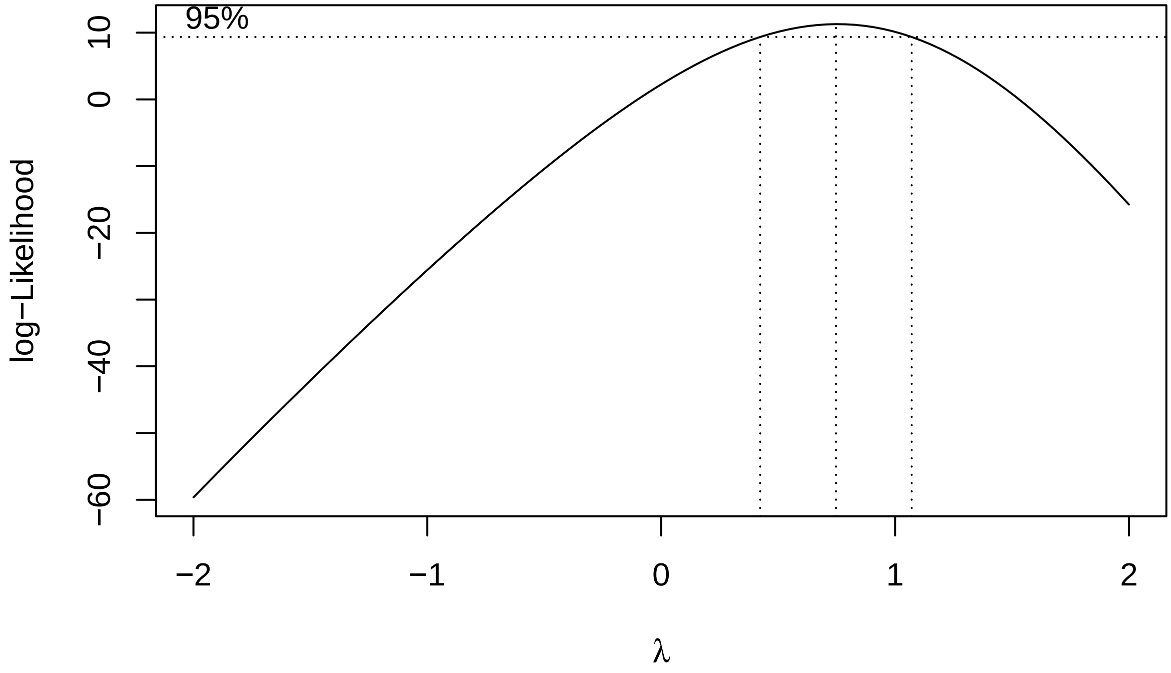
```
## (j) To calculate Cook's distance, we can use the cooks.distance function
```

```
# Calculate Cook's distance
cook_dist <- cooks.distance(model)

# Calculate threshold
threshold <- 4/nrow(new_cereal)

# Count number of observations with Cook's distance greater than threshold
num_observations <- sum(cook_dist > threshold)

show(num_observations)
```

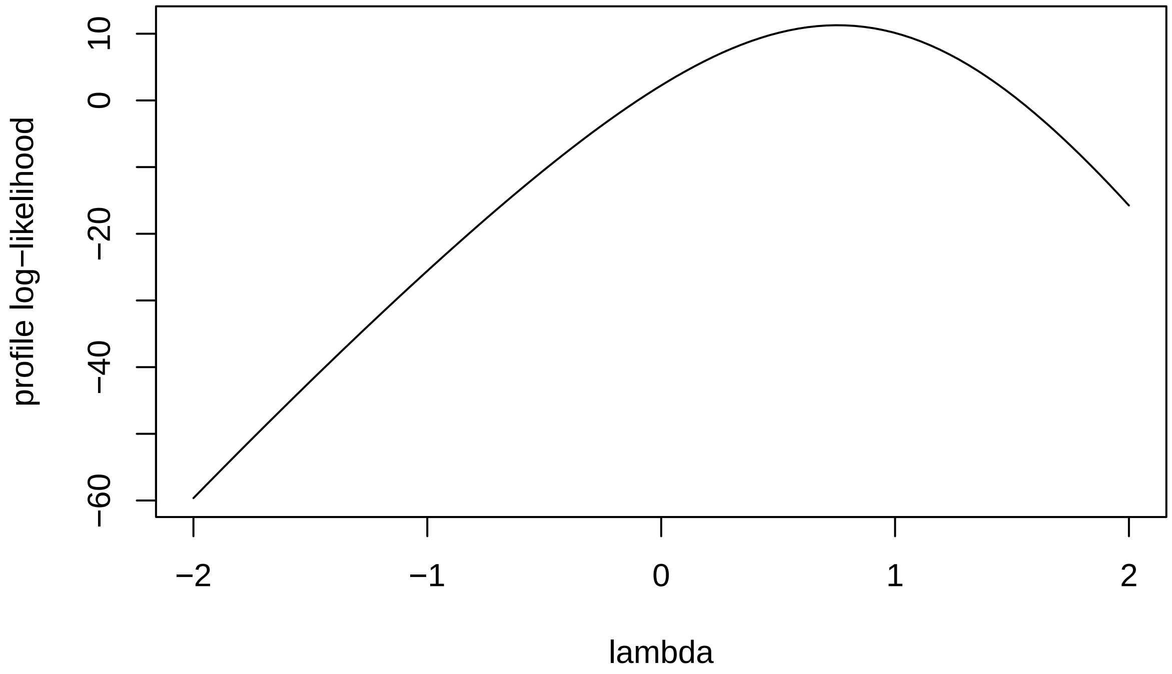
```
## [1] 7
```

```
# From the outcome, there're 7 observations in this data set have a
##Cook’s distance that is greater than 4/n.
```

```
## (k)To check whether the response needs a Box-Cox transformation,
##we can use the boxcox function
library(MASS)

# Fit the Box-Cox transformation
boxcox_model <- boxcox(model)
```



```r
# Plot the profile log-likelihood and the recommended lambda value
plot(boxcox_model$x, boxcox_model$y, type = "l", xlab = "lambda", ylab = "profile log-likelihood")

abline(v = boxcox_model$lambda, lty = 2)
```



```r
# From the plot above, output of the boxcox function is around 1.0, it suggests
##that a Box-Cox transformation is not necessary, and that a linear regression
##model is appropriate for the data.
```