# lab3

Hoang Chu

2023-11-08

```
library(faraway)
```

## 9.3

```
data(ozone)
```

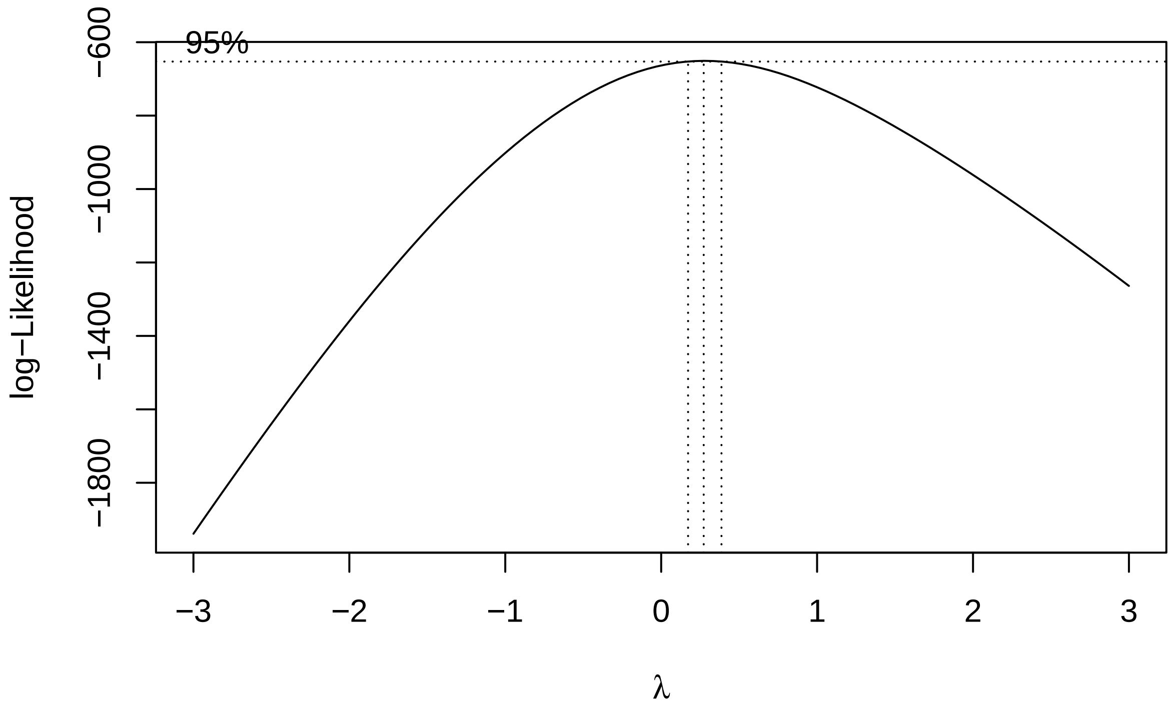
```
model <- lm(O3 ~ temp + humidity + ibh, data = ozone)
summary(model)
```

```
##
## Call:
## lm(formula = O3 ~ temp + humidity + ibh, data = ozone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5291  -3.0137  -0.2249   2.8239  13.9303
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.049e+01  1.616e+00  -6.492 3.16e-10 ***
## temp         3.296e-01  2.109e-02  15.626  < 2e-16 ***
## humidity     7.738e-02  1.339e-02   5.777 1.77e-08 ***
## ibh         -1.004e-03  1.639e-04  -6.130 2.54e-09 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 4.524 on 326 degrees of freedom
## Multiple R-squared:  0.684,  Adjusted R-squared:  0.6811
## F-statistic: 235.2 on 3 and 326 DF,  p-value: < 2.2e-16
```

```
all(ozone$O3 > 0)
```

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

```
library(MASS)
bc <- boxcox(model, lambda = seq(-3,3,0.1))
```



```r
best_lambda <- bc$x[which.max(bc$y)]
```

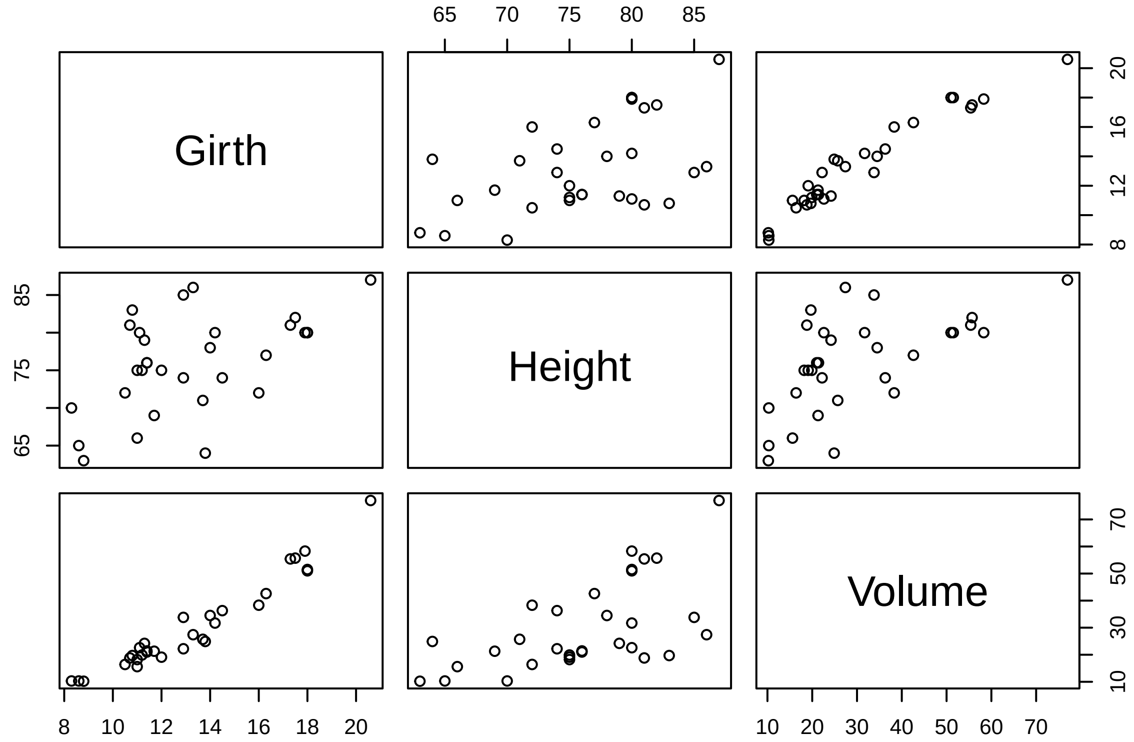
```r
ozone$O3 <- (ozone$O3^best_lambda - 1) / best_lambda
model_transformed <- lm(O3 ~ temp + humidity + ibh, data = ozone)
```

```r
summary(model_transformed)
```

```
##
## Call:
## lm(formula = O3 ~ temp + humidity + ibh, data = ozone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.25617 -0.44974  0.03727  0.52010  2.06666
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.360e-01  2.600e-01  -1.677   0.0945 .
## temp         5.510e-02  3.393e-03  16.238  < 2e-16 ***
## humidity     1.260e-02  2.155e-03   5.847 1.21e-08 ***
## ibh         -2.033e-04  2.636e-05  -7.713 1.50e-13 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.7278 on 326 degrees of freedom
## Multiple R-squared:  0.716,  Adjusted R-squared:  0.7134
## F-statistic:   274 on 3 and 326 DF,  p-value: < 2.2e-16
```

## 9.5

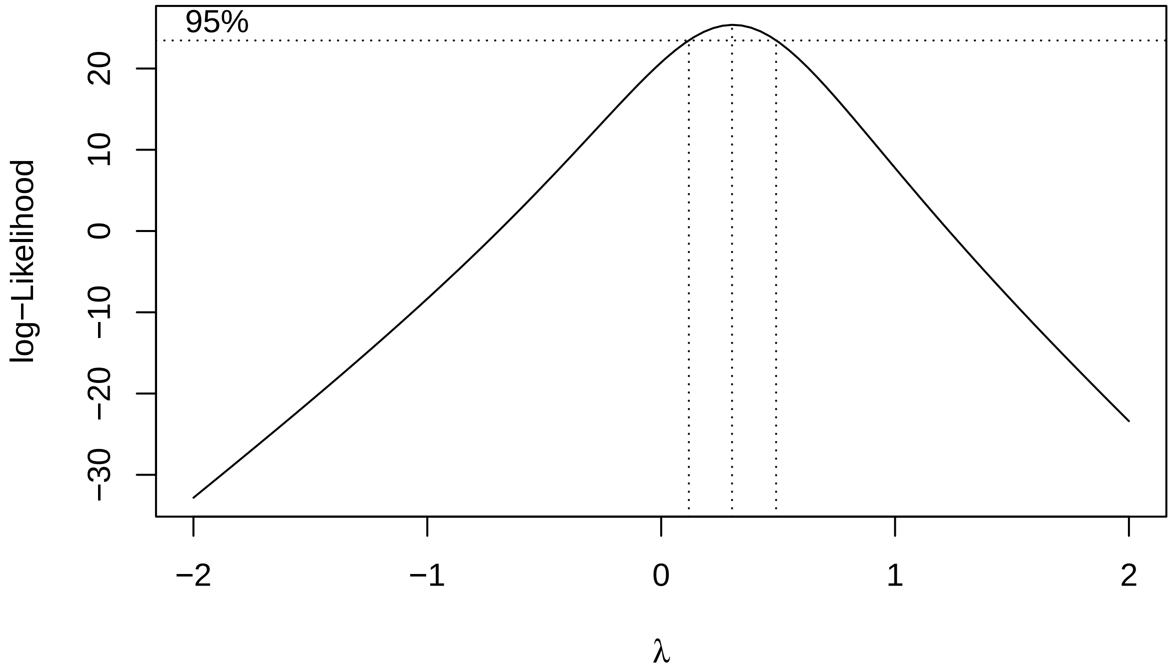
```
data(trees)
pairs(trees)
```



```
all(trees$Volume > 0)
```

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

```
library(MASS)
# Use the Box-Cox transformation to find optimal lambda for transformation
boxcox_result <- boxcox(Volume ~ Girth + Height, data = trees)
```



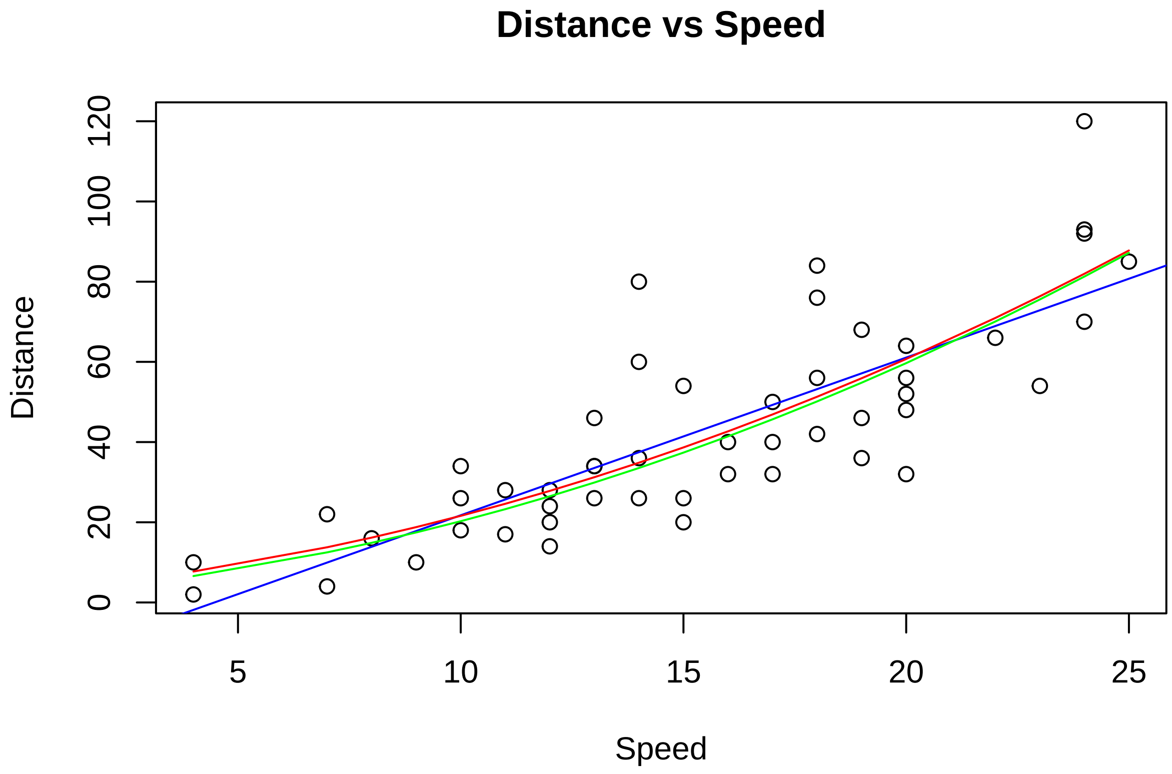
```
lambda <- boxcox_result$x[which.max(boxcox_result$y)]
trees$transformed_volume <- (trees$Volume^lambda - 1) / lambda
```

```
model <- lm(transformed_volume ~ Girth + Height, data = trees)
summary(model)
```

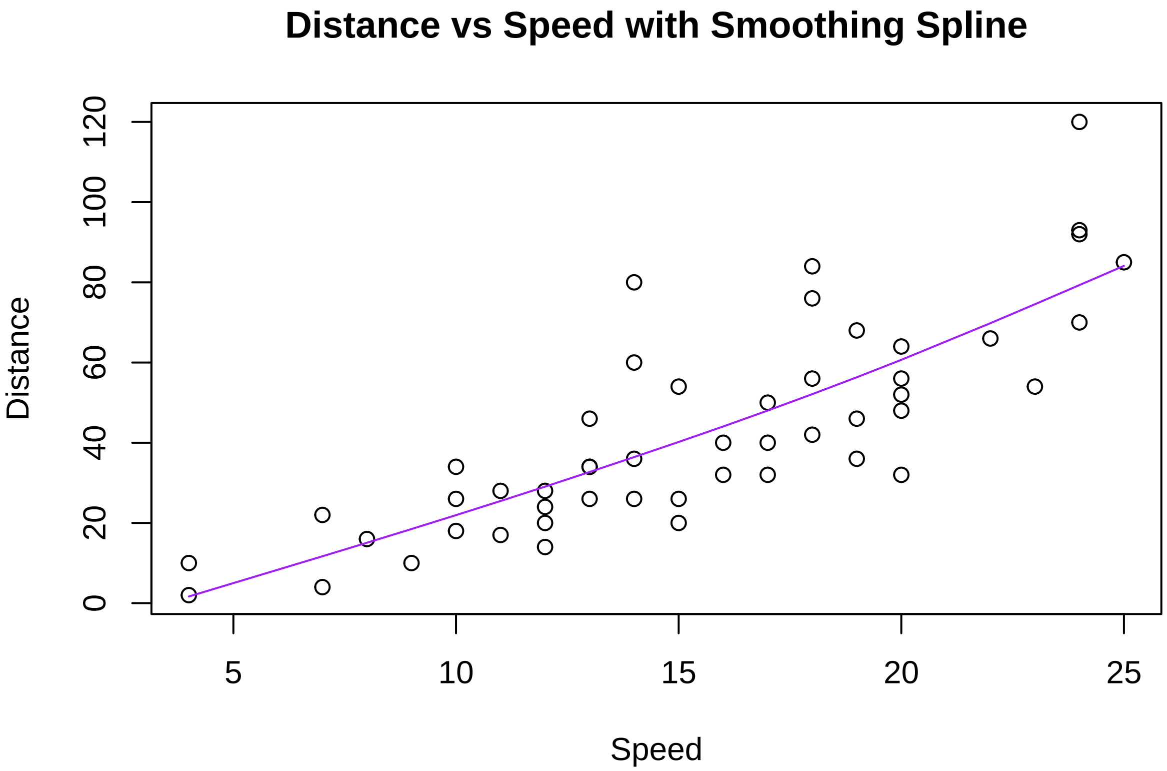
```
##
## Call:
## lm(formula = transformed_volume ~ Girth + Height, data = trees)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42600 -0.14274 -0.01468  0.18705  0.36851
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.733542   0.500080  -5.466 7.77e-06 ***
## Girth        0.409448   0.015299  26.764  < 2e-16 ***
## Height       0.039685   0.007535   5.267 1.34e-05 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.2247 on 28 degrees of freedom
## Multiple R-squared:  0.9775, Adjusted R-squared:  0.9759
## F-statistic: 609.6 on 2 and 28 DF,  p-value: < 2.2e-16
```

## 9.8

```r
# Load the cars dataset
data(cars)

# (a) Plot distance against speed
plot(cars$speed, cars$dist, main = "Distance vs Speed", xlab = "Speed", ylab = "Distance")

# (b) Show a linear fit to the data on the plot
abline(lm(dist ~ speed, data = cars), col = "blue")

# (c) Show a quadratic fit to the data on the plot
model_quad <- lm(dist ~ speed + I(speed^2), data = cars)
lines(cars$speed, predict(model_quad), col = "red")

# (d) Now use sqrt(dist) as the response and fit a linear model. Show the fit on the same plot
model_sqrt <- lm(sqrt(dist) ~ speed, data = cars)
lines(cars$speed, predict(model_sqrt)^2, col = "green")
```



```r
# (e) Compute the default smoothing spline fit to the plot and display on a fresh plot of the data
smooth_spline_fit <- smooth.spline(cars$speed, cars$dist)
plot(cars$speed, cars$dist, main = "Distance vs Speed with Smoothing Spline", xlab = "Speed", ylab = "Di
lines(smooth_spline_fit, col = "purple")
```


Distance vs Speed with Smoothing Spline
Distance
0
20
40
60
80
100
120
Speed
5
10
15
20
25