Regression_HW3

Warren Geither

10/20/2020

Problem 2

a.)

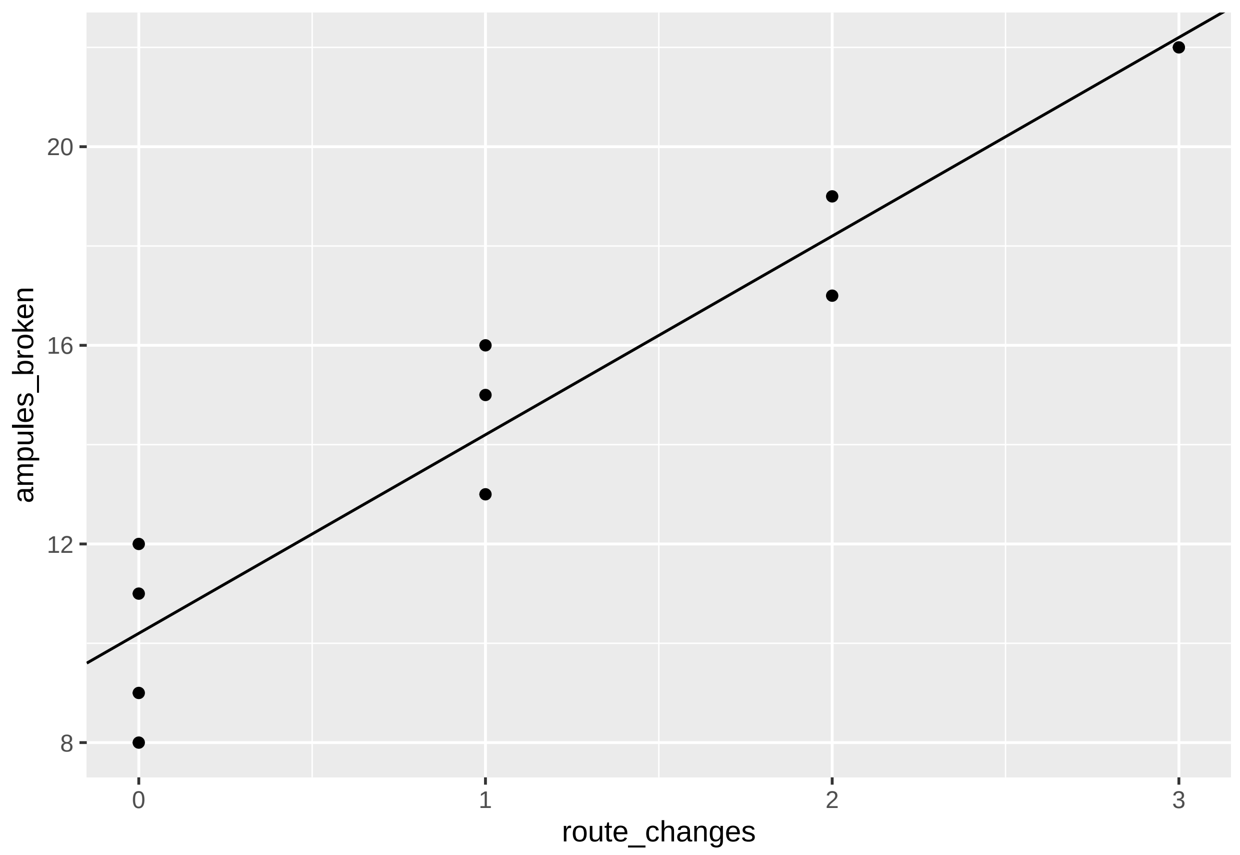
```r
# create dataframe
freight_data_df <- data.frame(route_changes = c(1,0,2,0,3,1,0,1,2,0)
                              , ampules_broken = c(16,9,17,12,22,13,8,15,19,11))

# get values needed to calculate
x <- freight_data_df$route_changes
y <- freight_data_df$ampules_broken
x_bar <- mean(x)
ybar <- mean(y)
n <- length(x)

# get beta hats
beta1hat = sum((x-x_bar)*(y-ybar))/sum((x-x_bar)^2)
beta0hat = ybar-beta1hat*x_bar

# print estimates
print(paste0("betahat0: ", beta0hat))
```

```
## [1] "betahat0: 10.2"
```

```r
print(paste0("betahat1: ", beta1hat))
```

```
## [1] "betahat1: 4"
```

```r
# plot
ggplot(freight_data_df,aes(route_changes, ampules_broken))+
  geom_point()+
  geom_abline(slope=beta1hat, intercept=beta0hat)
```



b.)

```r
# sigma hat^2 = Y^T(I-H)Y/n-2
y_T <- t(y)
I <- diag(10)
X <- cbind(c(rep(1,10)), x)
X_T <- t(X)
H <- X%*%solve(X_T%*%X)%*%X_T

# sig hat
sigma_hat_squared <- ((y_T%*%(I - H)%*%y)/(n-2))

# sxx
sxx <- sum((x-x_bar)^2)

# t-value
crit_val <- qt(0.05/2, 8, lower.tail = FALSE)

# beta0hat C.I
beta0_upper_bound <- beta0hat + crit_val*sqrt(sigma_hat_squared*((1/n)+((x_bar^2)/sxx)))
beta0_lower_bound <- beta0hat - crit_val*sqrt(sigma_hat_squared*((1/n)+((x_bar^2)/sxx)))

# format for print
b0_ci1 <- paste0(round(beta0_lower_bound,3), ",")
b0_ci2 <- paste0(b0_ci1, round(beta0_upper_bound, 3))
b0_ci3 <- paste0("(",b0_ci2)
b0_ci4 <- paste0(b0_ci3, ")")

# beta1hat C.I
beta1_upper_bound <- beta1hat + crit_val*sqrt(sigma_hat_squared/sxx)
beta1_lower_bound <- beta1hat - crit_val*sqrt(sigma_hat_squared/sxx)

# format for print
b1_ci1 <- paste0(round(beta1_lower_bound,3), ",")
b1_ci2 <- paste0(b1_ci1, round(beta1_upper_bound, 3))
b1_ci3 <- paste0("(",b1_ci2)
b1_ci4 <- paste0(b1_ci3, ")")

# print C.I.
print(paste0("95% C.I. for beta0hat: ", b0_ci4))
```

```
## [1] "95% C.I. for beta0hat: (8.67,11.73)"
```

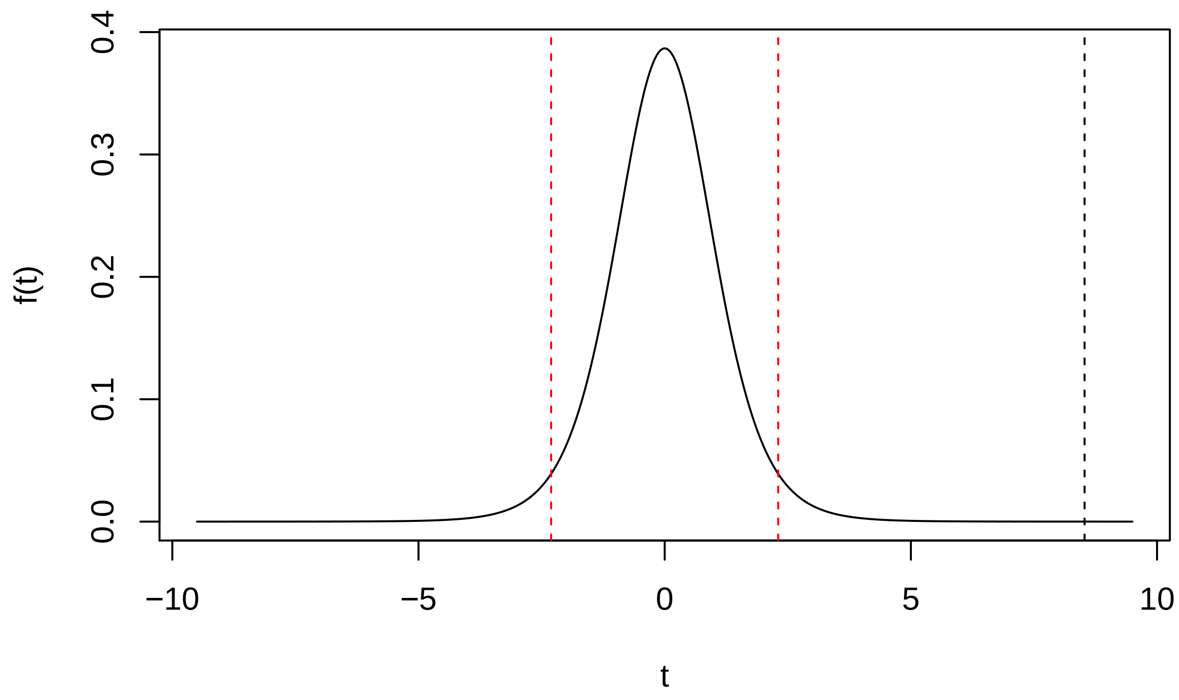
```r
print(paste0("95% C.I. for beta1hat: ", b1_ci4))
```

```
## [1] "95% C.I. for beta1hat: (2.918,5.082)"
```

c.)

$$\text{Hypothesis for linear relationship}$$
$$Ho : \beta_1 = 0$$
$$Ha : \beta_1 \neq 0$$

```r
test_stat <- (beta1hat - 0)/sqrt(sigma_hat_squared/sxx)

# For the plot for t test
dum=seq(-9.5, 9.5, length=10^4)

plot(dum, dt(dum, df=(n-2)), type='l', xlab='t', ylab='f(t)')
abline(v=test_stat, lty=2)
abline(v=crit_val, col='red', lty=2)
abline(v=-crit_val, col='red', lty=2)
```



Because our test statistic of 8.5 is greater than our postive critical value of 2.3, we can reject the null hypothesis at the alpha=0.05 level and say that there is a linear relationship between broken amplues and ship route changes

Problem 3

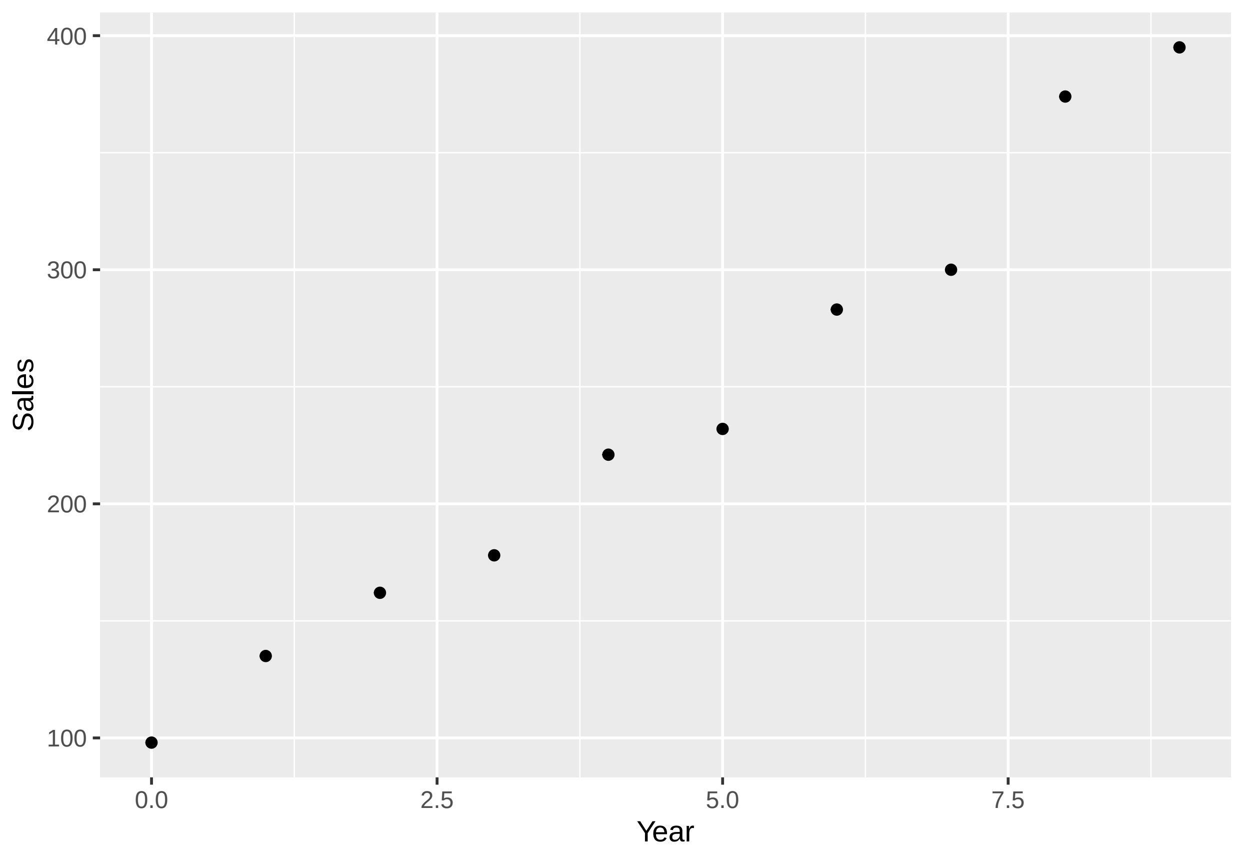
```r
# set seed
set.seed(201547)

# intialize list of estimates
betahat1_vector <- c(rep(NA,100))

# create a loop to bootstrap betahat1
for (i in 1:100){
  # Sample 10 rows from data
  sample_row_nums <- sample(nrow(freight_data_df), n, replace = T)

  # Store sample in dataframe
  sample_rows <- freight_data_df[sample_row_nums, ]

  # get values for betahat1 calculation
  x <- sample_rows$route_changes
  y <- sample_rows$ampules_broken
  x_bar <- mean(x)
  ybar <- mean(y)
  n <- length(x)

  # calc betahat1
  beta1hat = sum((x-x_bar)*(y-ybar))/sum((x-x_bar)^2)

  # store in list
  betahat1_vector[i] <- beta1hat
}

# remove na values
betahat1_vector <- betahat1_vector[!is.na(betahat1_vector)]

# calculate confidence interval
quantile(x = betahat1_vector, probs = c(0.025,0.975))
```

```
##     2.5%    97.5%
## 3.256755 5.866964
```

The intervals are similar since both account for the variance in beta1hat. However the first confidence interval multiples by the critical value for the t-distribution at the alpha level we are using. The bootstrap only picks up the natural variance of the mean of beta1hat from the bootstrapping.

Problem 1 & 4

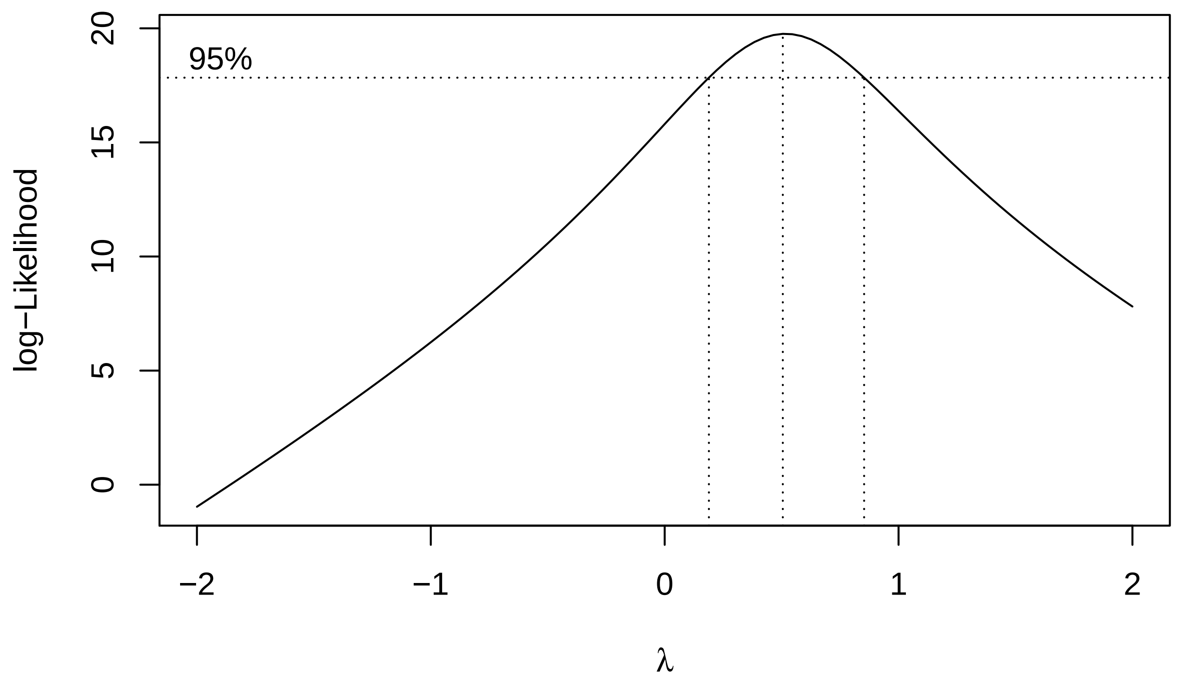
```
# create dataframe
sales_df <- data.frame(year = c(seq(0,9,by=1))
                       , sales = c(98,135,162,178,221,232,283,300,374,395))

# fit model
lmfit <- lm(sales ~ year, sales_df)

# plot line
ggplot(sales_df, aes(x= year, y=sales)) +
  geom_point() +
  #geom_abline(slope= lmfit$coef[2], intercept=lmfit$coef[1]) +
  xlab("Year") +
  ylab("Sales")
```



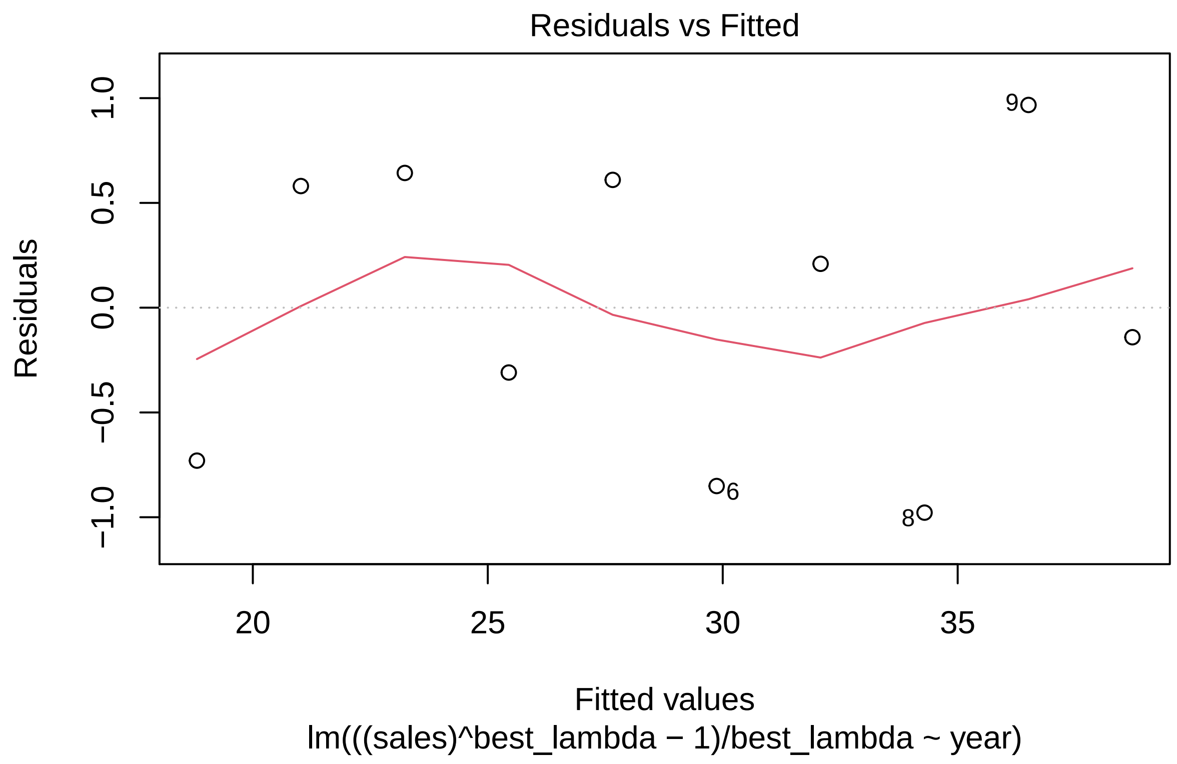
```r
knitr::kable(summary(sales_df))
```

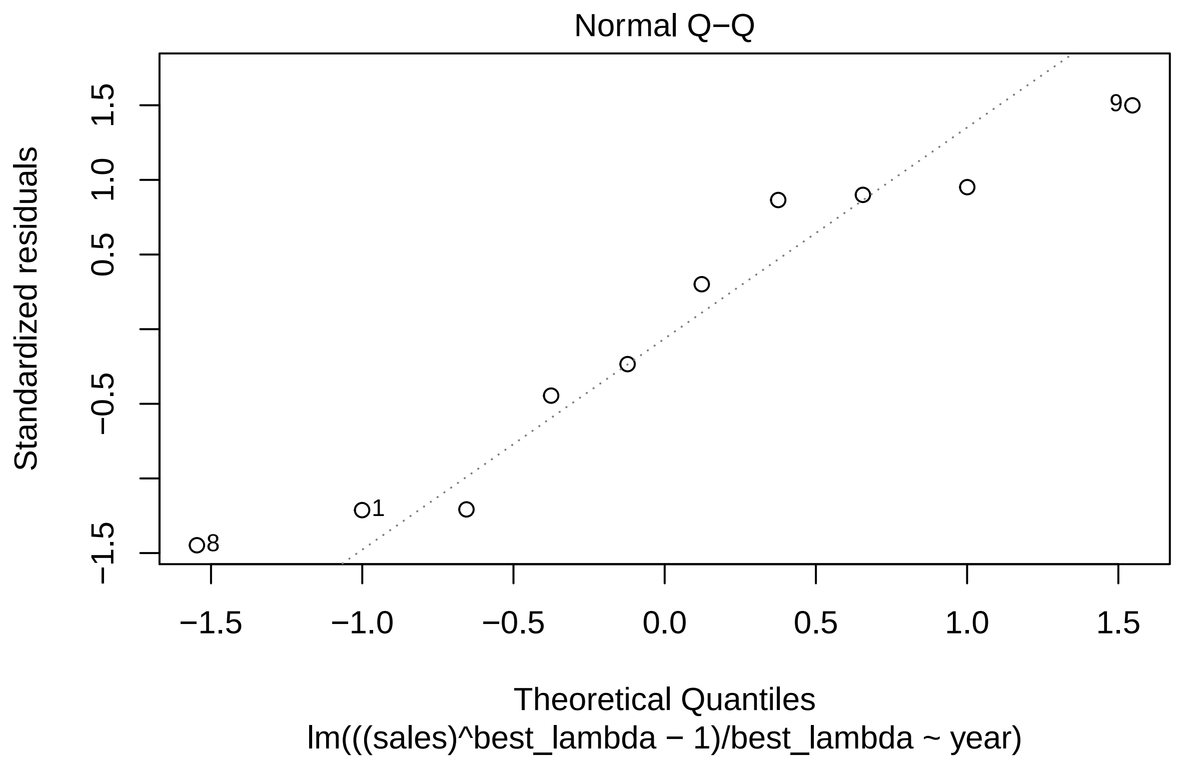
year	sales
Min. :0.00	Min. : 98.0
1st Qu.:2.25	1st Qu.:166.0
Median :4.50	Median :226.5
Mean :4.50	Mean :237.8
3rd Qu.:6.75	3rd Qu.:295.8
Max. :9.00	Max. :395.0

```r
# plot of picking best lambda
boxcox(lmfit,plotit=T)

# storing transform
bc_transform <- boxcox(lmfit)
```

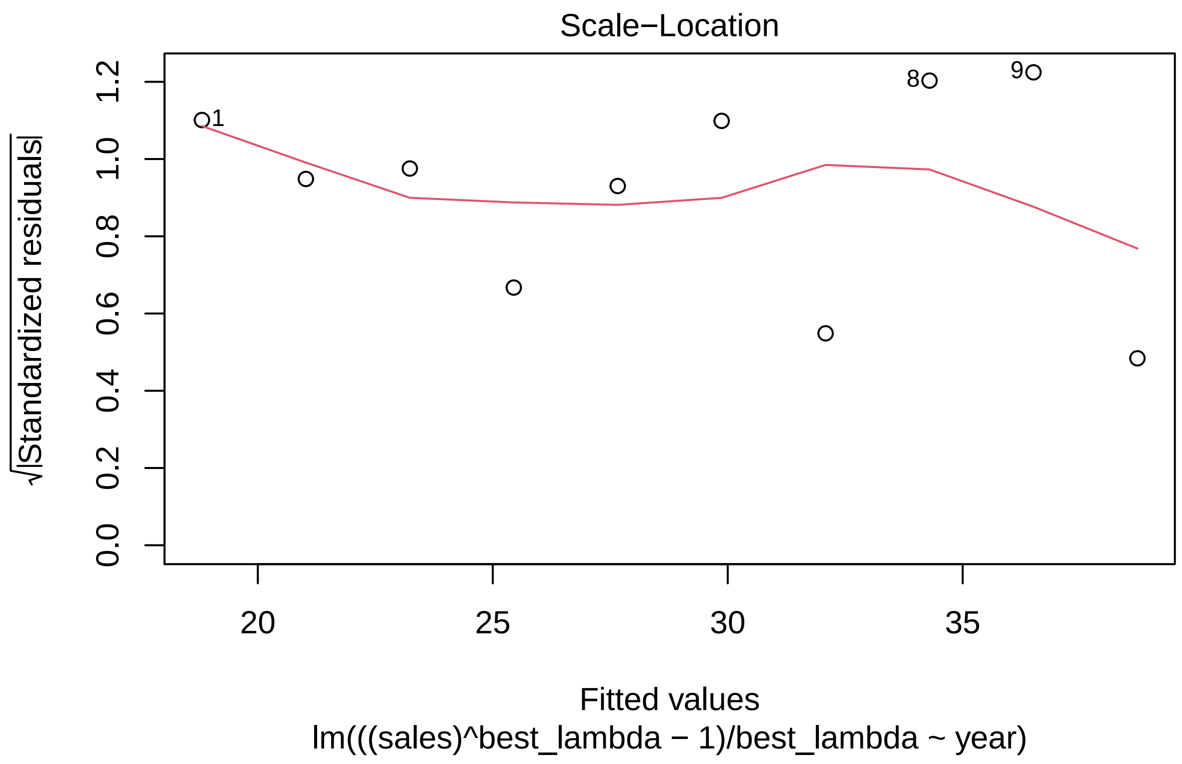


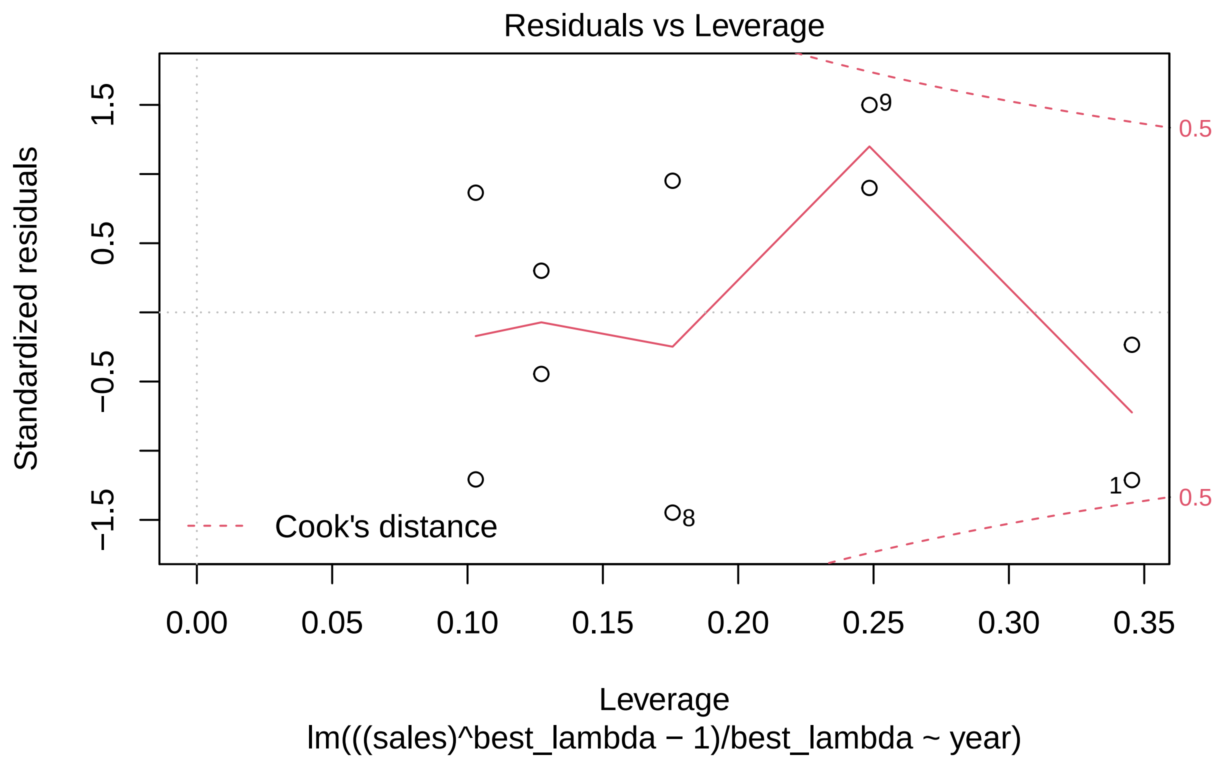
```
# picking lambda that maximimizs the log liklihood
best_lambda <- bc_transform$x[which(bc_transform$y==max(bc_transform$y))]

# fitting new model
bcfit <- lm(((sales)^best_lambda - 1)/best_lambda~year, data = sales_df)

# plot
plot(bcfit)
```


Residuals vs Fitted
Residuals
Fitted values
lm(((sales)^best_lambda − 1)/best_lambda ~ year)
1.0
0.5
0.0
−0.5
−1.0
20
25
30
35
9
6
8



Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(((sales)^best_lambda − 1)/best_lambda ~ year)
−1.5
−0.5
0.5
1.0
1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
8
1
9


Scale−Location
√|Standardized residuals|
0.0
0.2
0.4
0.6
0.8
1.0
1.2
20
25
30
35
1
8
9
Fitted values
lm(((sales)^best_lambda − 1)/best_lambda ~ year)



Problem 4

Randomness Check

$$\text{Hypothesis for Runs Test}$$
$$Ho : \text{Random}$$
$$Ha : \text{Not Random}$$

```
# get residuals
lmfit$residuals
```

```
##          1          2          3          4          5          6
##   6.4363636  10.9393939   5.4424242 -11.0545455  -0.5515152 -22.0484848
##          7          8          9         10
##  -3.5454545 -19.0424242  22.4606061  10.9636364
```

```
bcfit$residuals
```

```
##          1          2          3          4          5          6          7
## -0.7299340  0.5803457  0.6428700 -0.3095120  0.6097647 -0.8511021  0.2093167
##          8          9         10
## -0.9780384  0.9673339 -0.1410445
```

```r
# Durbin-Watson Test
dwtest(lmfit)
```

```
##
##  Durbin-Watson test
##
## data:  lmfit
## DW = 1.8521, p-value = 0.2503
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
dwtest(bcfit)
```

```
##
##  Durbin-Watson test
##
## data:  bcfit
## DW = 2.9688, p-value = 0.9067
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
# Runs Test
runs.test(residuals(lmfit))
```

```
##
##  Runs Test - Two sided
##
## data:  residuals(lmfit)
## Standardized Runs Statistic = -2.0125, p-value = 0.04417
```

```r
runs.test(residuals(bcfit))
```

```
##
##  Runs Test - Two sided
##
## data:  residuals(bcfit)
## Standardized Runs Statistic = 2.0125, p-value = 0.04417
```

Manual Runs Test:

Ho: residuals are random

From TableA30: rL = 2, rU=10, fail to reject when $2 < r < 10$ lmfit: 2 postive run & 1 negative run ==> r = 3 ==> fail to reject null bcfit: 4 positive runs & 5 negative runs ==> r=9 ==> fail to reject null

Durbin-watson Test:

Ho: autocorrelation = 0 (resdiduals are independent)

lmfit: p-val = 0.2503 ==> Fail to reject bcfit: p-val = 0.9067 ==> Fail to reject

Runs.test():

lmfit: p-val = 0.04417 ==> Reject Ho bcfit: p-val = 0.04417 ==> Reject Ho

Constant Variance Check

BF Test

Hypothesis for BF Test

Ho : Constant Variance

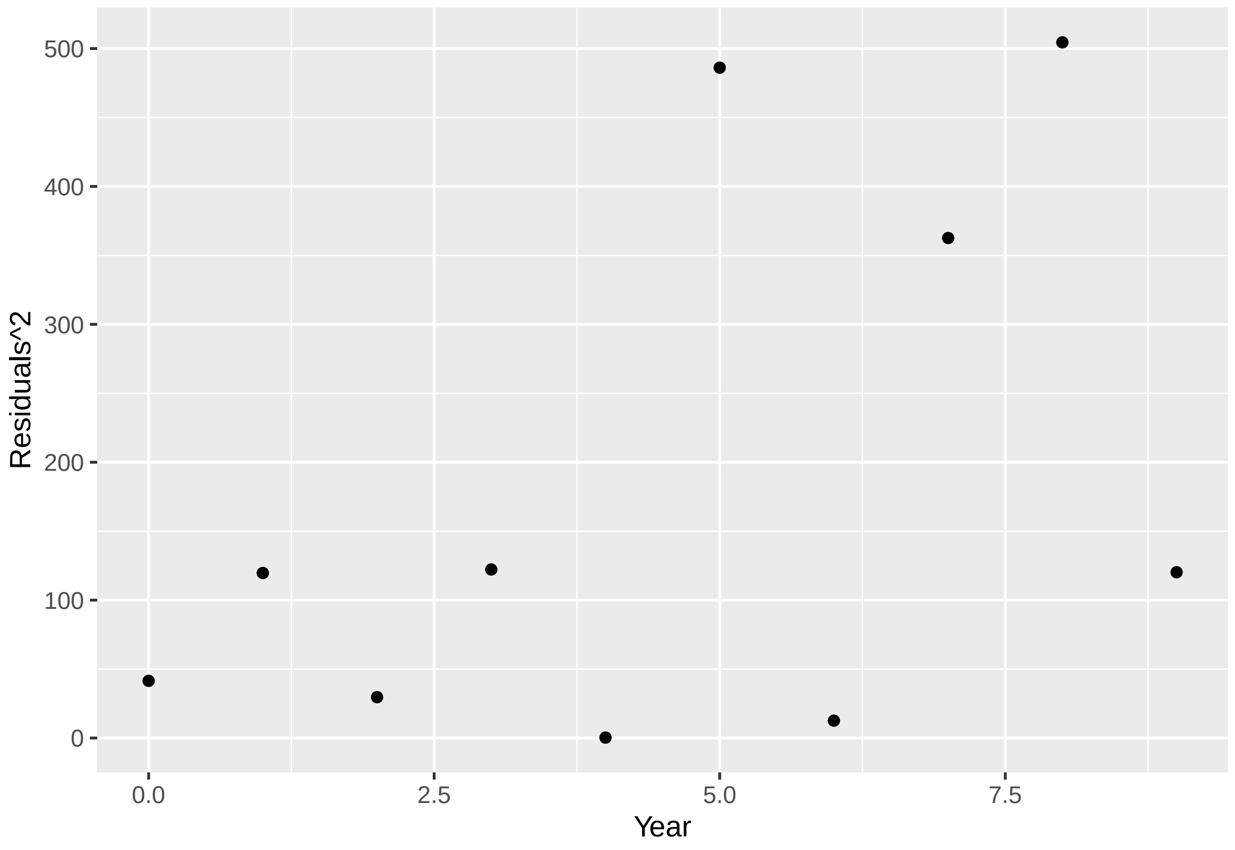
Ha : Not Constant Variance

```r
# set seed
set.seed(346565)

# initialize empty vector for test stats
p_val_matrix <- c(rep(NA, 100))

# loop for different groups (replace lmfit with other model for now)
for(i in 1:100){
  # Store sample in dataframe
  group1 <- sample(lmfit$residuals, 5, replace = FALSE)
  group2 <- lmfit$residuals[!lmfit$residuals %in% group1]

  # calculate n for the 2 groups
  n1 <- length(group1)
  n2 <- length(group2)

  # calculate medians
  median_res1 <- median(group1)
  median_res2 <- median(group2)

  # caluate deviation of residual from median
  d1_vals <- abs(group1 - median_res1)
  d2_vals <- abs(group2 - median_res2)

  s1 <- sd(d1_vals)
  s2 <- sd(d2_vals)

  # mean deviation
  d1_mean <- mean(d1_vals)
  d2_mean <- mean(d2_vals)

  # test statistic
  t_stat <- (d1_mean-d2_mean)/sqrt(((s1^2)/n1)+((s2^2)/n2))

  # simplifies df equation below
  A <- s1^2/n1
  B <- s2^2/n2

  # degrees of freedom for Welch 2-sample t-test
  # https://mse.redwoods.edu/darnold/math15/spring2013/R/Activities/WelchTTest.html
  df <- (A+B)^2/(A^2/(n1-1)+B^2/(n2-1))

  # p-value
  p = 2*pt(t_stat,df)

  # store result
  p_val_matrix[i] <- p
}

# print results
p_val_matrix
```

```
##   [1] 1.0127903 0.6833068 1.0591241 1.7954535 1.4935943 1.1893278 0.4425360
##   [8] 0.3762056 1.7474562 0.9358967 0.9097482 0.5804541 0.5785870 0.8755569
##  [15] 0.3304029 0.6468573 0.7177469 1.0641033 0.9281595 0.6912196 1.4225747
##  [22] 0.4985865 1.1233789 1.7263108 0.3917898 1.5073846 1.8572235 1.1233789
##  [29] 1.1233789 1.7952796 0.6833068 0.4139309 1.4195459 1.6237944 0.2070652
##  [36] 0.4139309 0.9658585 1.5254456 0.6958493 0.3918183 0.7788651 1.1244431
##  [43] 1.5975548 0.4827416 0.7225695 0.6128461 0.4990550 1.5957089 0.6963129
##  [50] 0.4915082 0.4982457 0.8100386 1.5968269 1.0817940 1.1292898 0.6916434
##  [57] 0.1432145 0.2070652 1.7785345 1.7949001 1.5289658 1.2609880 1.3531427
##  [64] 0.6468573 0.3304029 1.4635847 0.2045465 0.3304029 0.9408759 1.6092114
##  [71] 1.0718405 0.8755569 1.2223233 1.2772179 1.3346527 0.2736892 1.4441517
##  [78] 0.8100386 1.8039635 0.9110114 1.1004406 1.3193656 1.7929348 1.0127903
##  [85] 0.6128461 1.7954535 0.4092329 1.5372811 1.2472777 0.4425360 0.4140226
##  [92] 0.4915082 0.8282045 1.8045752 1.2223233 1.4214130 1.5860691 1.3041507
##  [99] 0.9033070 0.8472066
```

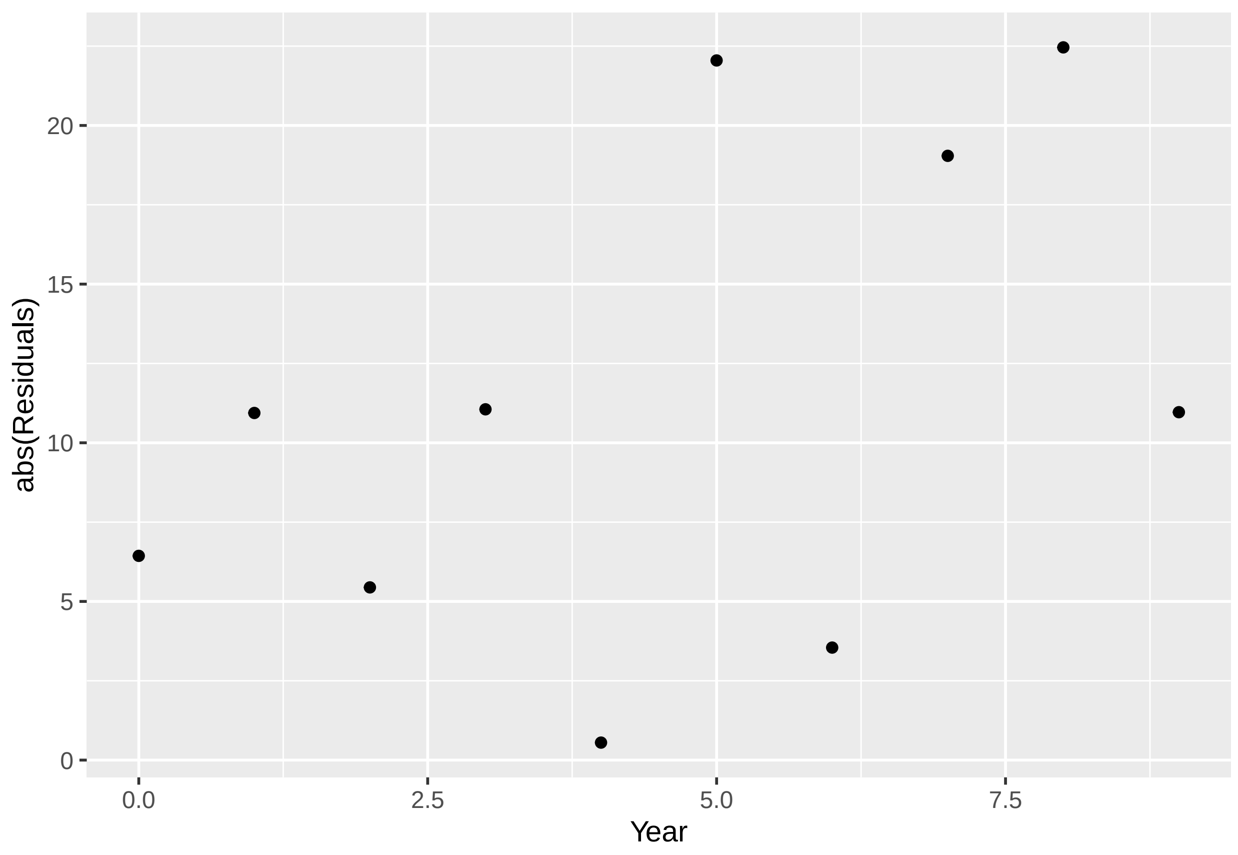
lmfit: Fail to Reject 100/100 times bcfit: Fail to Reject 100/100 times

```r


# plots for bp test
ggplot(sales_df, aes(x= year, y=residuals(lmfit)^2)) +
  geom_point() +
  xlab("Year") +
  ylab("Residuals^2")
```

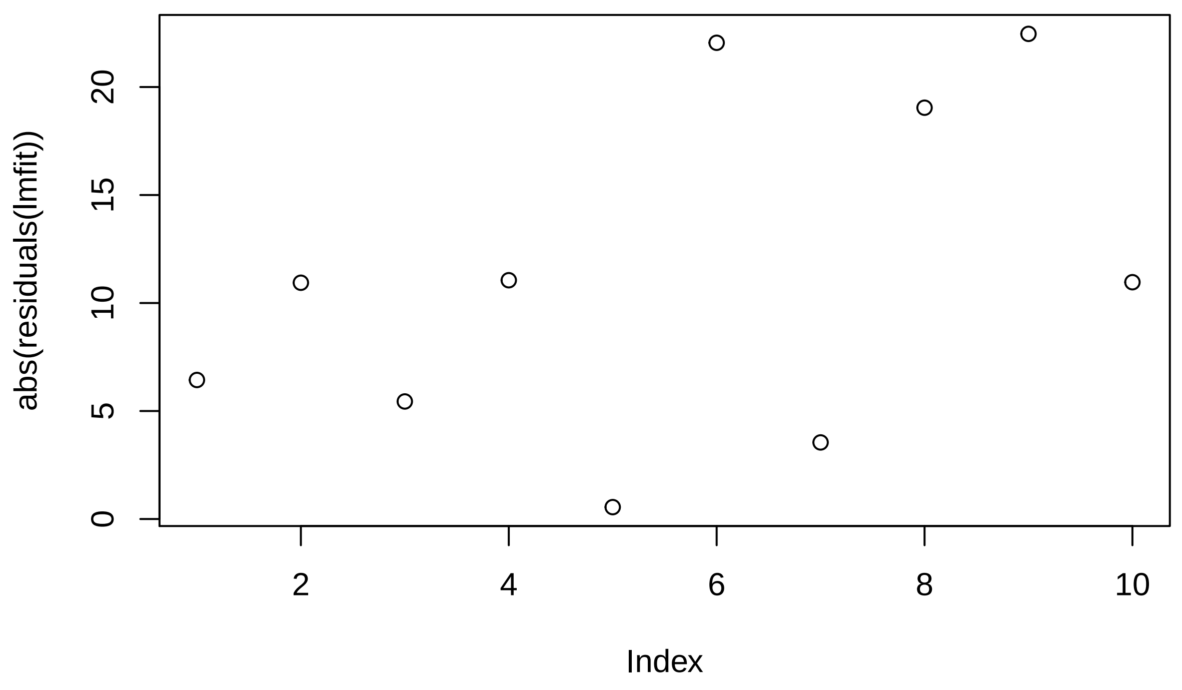


BP Test

```r
ggplot(sales_df, aes(x= year, y=abs(residuals(lmfit)))) +
  geom_point() +
  xlab("Year") +
  ylab("abs(Residuals)")
```



```r
plot(abs(residuals(lmfit)))
```



```r
# conduct bp test
bptest(sales~year, data = sales_df)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  sales ~ year
## BP = 2.3972, df = 1, p-value = 0.1216
```

```r
shapiro.test(residuals(lmfit))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lmfit)
## W = 0.96123, p-value = 0.7998
```

```r
library(nortest)
```

```
## Warning: package 'nortest' was built under R version 4.0.3
```

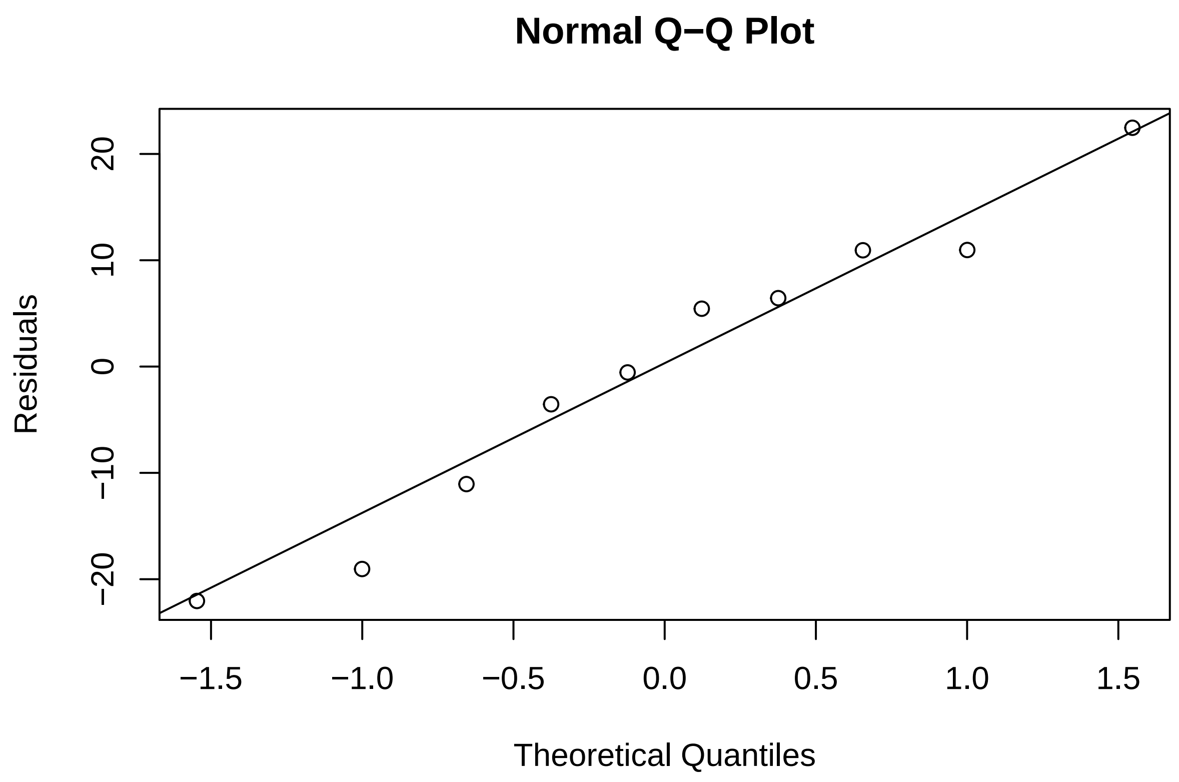
```r
ad.test(residuals(lmfit))
```

```
##
##  Anderson-Darling normality test
##
## data:  residuals(lmfit)
## A = 0.22053, p-value = 0.7703
```

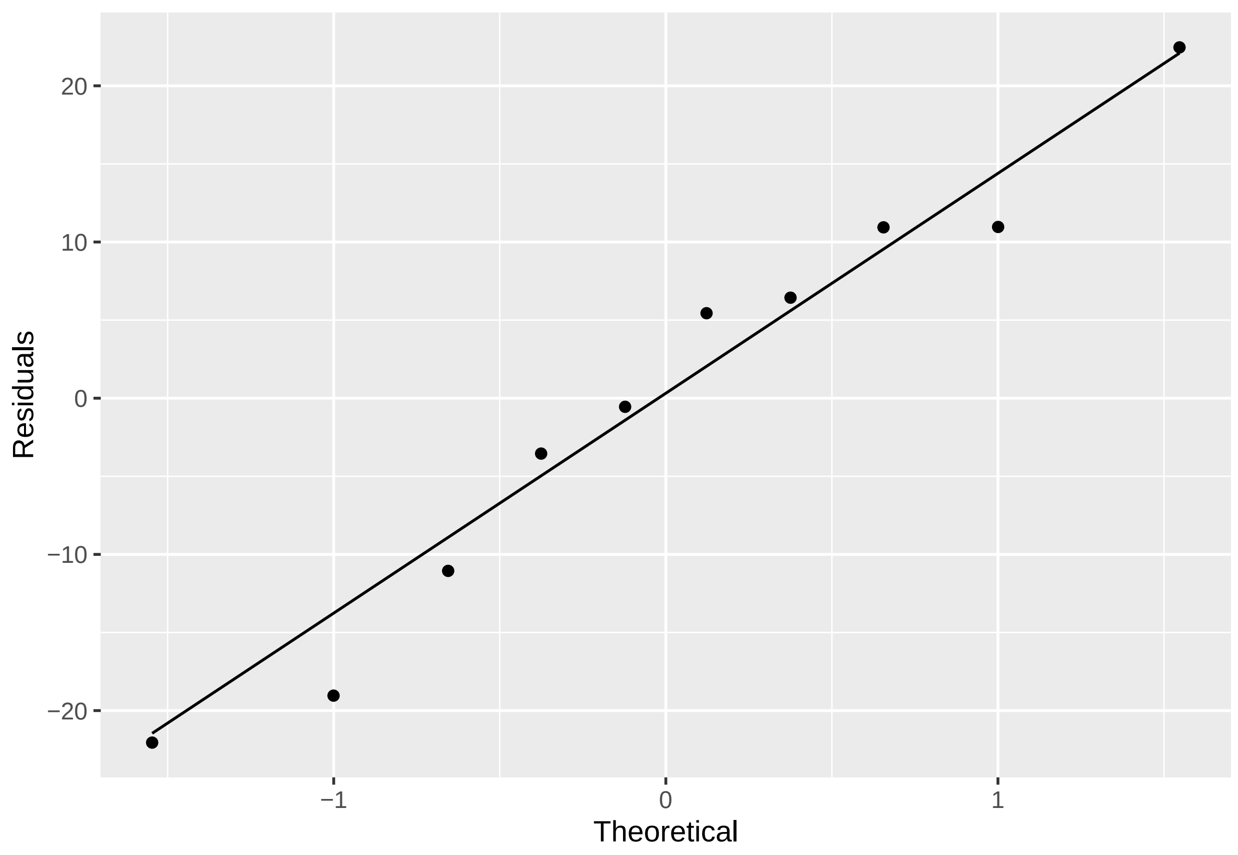
```
library(faraway)
```

```
## Warning: package 'faraway' was built under R version 4.0.3

## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
```

```
qqnorm(residuals(lmfit), ylab = "Residuals")
qqline(residuals(lmfit))
```



```
ggplot(lmfit, aes(sample=residuals(lmfit)))+
  stat_qq() +
  stat_qq_line() +
  xlab("Theoretical") +
  ylab("Residuals")
```



Appendix

```
g1<-c(6.4363636,  10.9393939 ,  5.4424242, -11.0545455 , -0.5515152)
g2<-c(-22.0484848 , -3.5454545,-19.0424242,  22.4606061,  10.9636364 )

n1 <- length(g1)
n2 <- length(g2)

d1<-abs(g1-median(g1))
d2<-abs(g2-median(g2))

  s1 <- sd(d1)
  s2 <- sd(d2)

    # # mean deviation
  d1_mean <- mean(d1)
  d2_mean <- mean(d2)

  t_stat=(d1_mean-d2_mean)/sqrt(((s1^2)/n1)+((s2^2)/n2))

  A=(s1^2)/n1
  B=(s2^2)/n2

  df=((A+B)^2)/(((A^2)/(n1-1))+((B^2)/(n2-1)))
  p = 2*pt(t_stat,df)
```

```
# BF test using function (not sure if this works)
group1 <- sample(lmfit$residuals, 5, replace = FALSE)
group2 <- lmfit$residuals[!lmfit$residuals %in% group1]

group1_mat <- data.frame(res = group1, group = c(rep(1,5)))
group2_mat <- data.frame(res = group1, group = c(rep(2,5)))

group_df <- rbind(group1_mat, group2_mat)

group_df$group <- as.factor(group_df$group)

bf.test(res ~ group, data = group_df)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : res and group
##
##   statistic  : 0
##   num df     : 1
##   denom df   : 8
##   p.value    : 1
##
##   Result     : Difference is not statistically significant.
## -------------------------------------------------------------
```

```
out <- bf.test(res ~ group, data = group_df)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : res and group
##
##   statistic  : 0
##   num df     : 1
##   denom df   : 8
##   p.value    : 1
##
##   Result     : Difference is not statistically significant.
## -------------------------------------------------------------
```

```
out$p.value
```

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