A Bootstrap Study Based on Linear Model

Zihuan Qiao

2.(a) Fitting the linear model

```
# load data
data("faithful")
attach(faithful)

# fit the linear model: Y ~ X (waiting ~ eruptions)
model <- lm (waiting ~ eruptions,faithful)

# regression model results
summary(model)
```

```
##
## Call:
## lm(formula = waiting ~ eruptions, data = faithful)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.0796  -4.4831   0.2122   3.9246  15.9719
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  33.4744     1.1549   28.98   <2e-16 ***
## eruptions    10.7296     0.3148   34.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.914 on 270 degrees of freedom
## Multiple R-squared:  0.8115, Adjusted R-squared:  0.8108
## F-statistic:  1162 on 1 and 270 DF,  p-value: < 2.2e-16
```

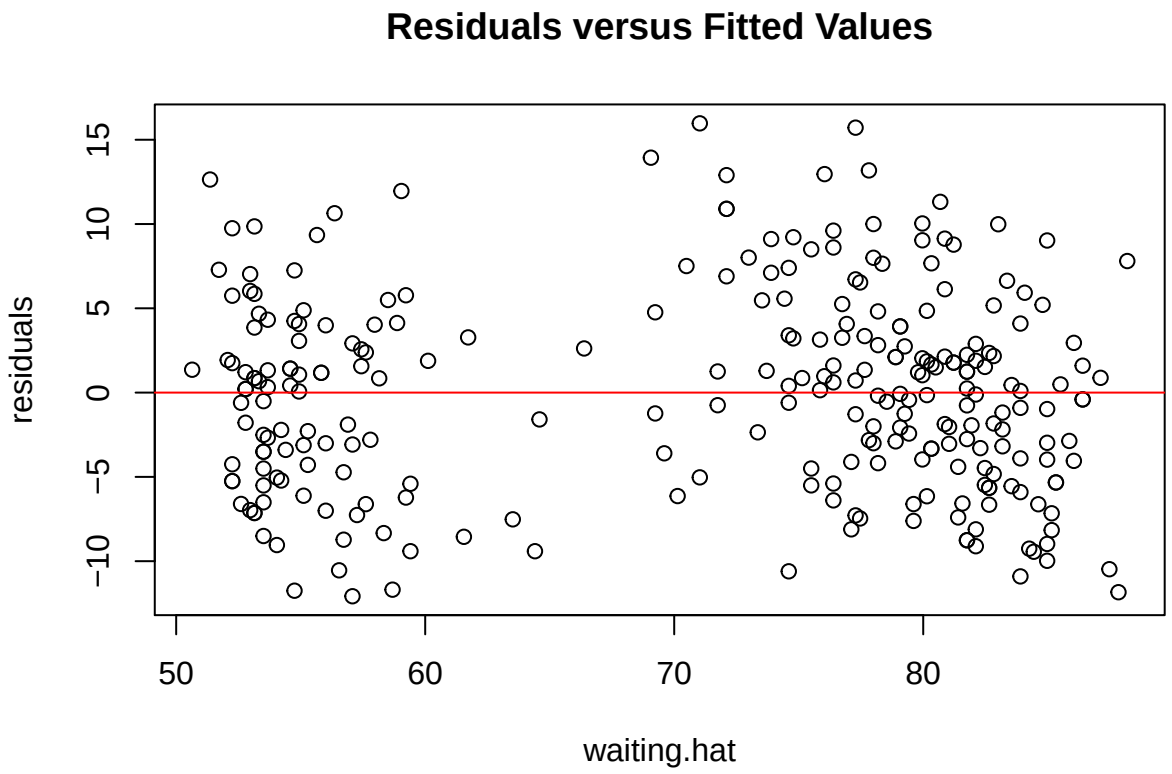
```
(beta.hat <- model$coefficients)
```

```
## (Intercept)   eruptions
##    33.47440    10.72964
```

```
(sigma.hat <- summary(model)$sigma)
```

```
## [1] 5.914009
```

```
# draw plots
residuals <- model$residuals
waiting.hat <- model$fitted.values
# residuals versus Y.hat
plot(residuals ~ waiting.hat, main = "Residuals versus Fitted Values")
abline(h=0, col="red")
```




```
# Y  versus Y.hat
plot(waiting ~ waiting.hat, main = "True Values versus Fitted Values")
lines(x=c(0,100),y=c(0,100), col="red")
```

By fitting the linear regression model using 'faithful' data, the formula for the linear model is: $waiting_i =$

$33.4744 + 10.72964 * eruptions_i + e_i, i = 1, ..., n$. The residual standard error is 5.914009.

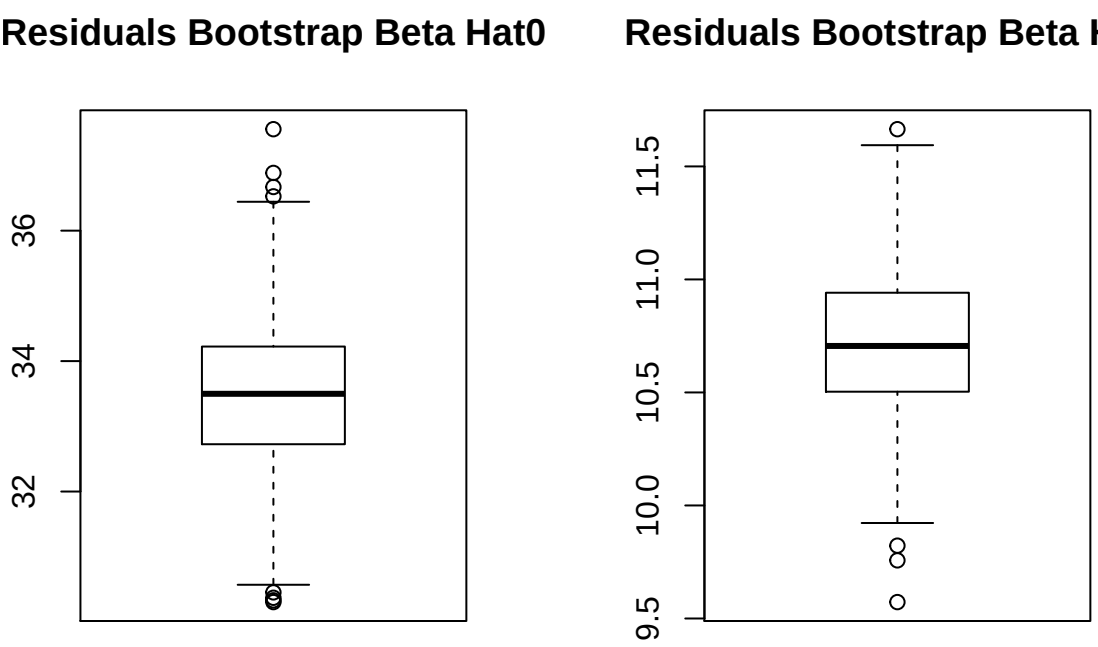
From the plot 'Residuals versus Fitted Values', we can see that at each waiting.hat, residuals $\hat{e_i}$ is at random. So residual are independent at each observation. Also, residuals are distributed almost evenly above and below the horizontal line residuals=0. This conforms assumptions for error that E[e_i]=0 and constant variance.

In the plot 'True Values versus Fitted Values', the red line is the diagonal line. If a point is exactly on the red line, it means the model got by least squares estimates provide a perfect match for this point. The closer to the red line, the better fit is the model with the point. So in general, most points spread evenly around the red line which means the estimated model can provide a good fit to the data. However, some points is a little bit far from the red line, residuals at these points are big.

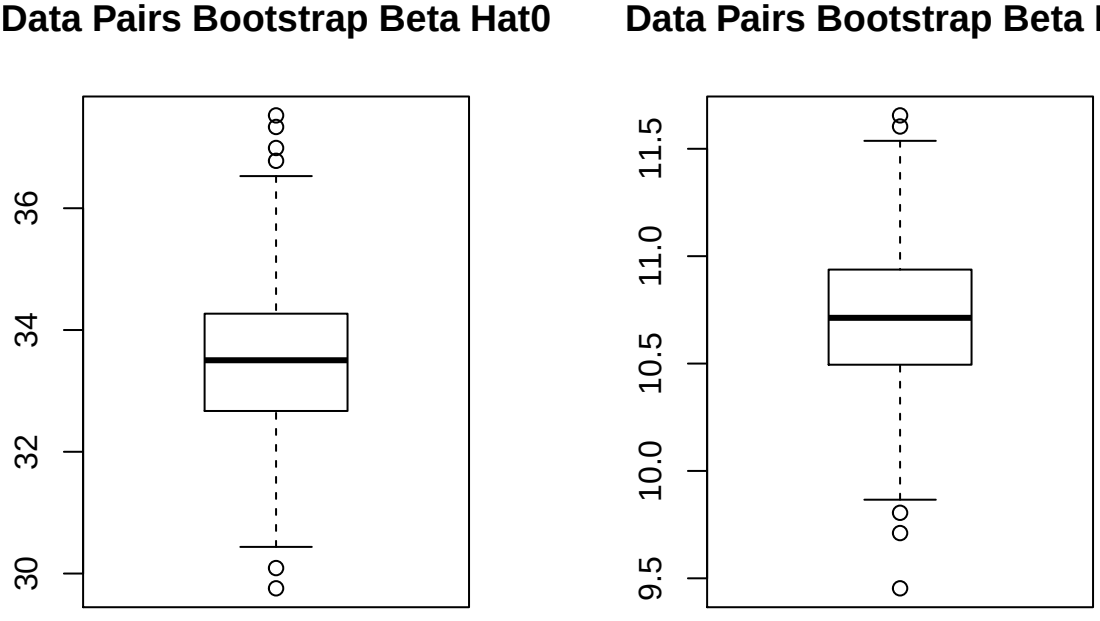
2.(b) Bootstrap simulations

```
# Bootstrap: linear regression
# `ll` is lm object, `type` is bootstrap type
# function returns coefficients of each bootstrap linear regression
lm.bootstrap <- function (ll, B = 100,
                          type = c('residual', 'pair', 'parametric')) {
  n <- length(fitted(ll))
  type <- match.arg(type)
  if (type == 'residual') {
    res <- residuals(ll)
    replicate(B, {
                yb <- fitted(ll) + res[sample.int(n, replace = TRUE)]
                boot <- model.matrix(ll)
                c(coef(lm(yb ~ boot - 1)))
              })
  }
  else if (type == 'parametric') {
    sigma.hat <- sqrt(deviance(ll) / df.residual(ll))
    replicate(B, {
                yb <- fitted(ll) + rnorm(n, 0, sigma.hat)
                boot <- model.matrix(ll)
                c(coef(lm(yb ~ boot - 1)))
              })
  }
  else # 'pair'
    replicate(B, c(coef(update(ll, subset = sample.int(n, replace = TRUE)))))
}


# conduct three bootstrap simulations
res.boot <- lm.bootstrap(model, B=1000, type = "residual")
pair.boot <- lm.bootstrap(model, B=1000, type = "pair")
para.boot <- lm.bootstrap(model, B=1000, type = "parametric")
res.boot <- t(res.boot)
pair.boot <- t(pair.boot)
para.boot <- t(para.boot)
colnames(res.boot) <- c("res.beta0hat","res.beta1hat")
colnames(pair.boot) <- c("pair.beta0hat","pair.beta1hat")
colnames(para.boot) <- c("para.beta0hat","para.beta1hat")
```

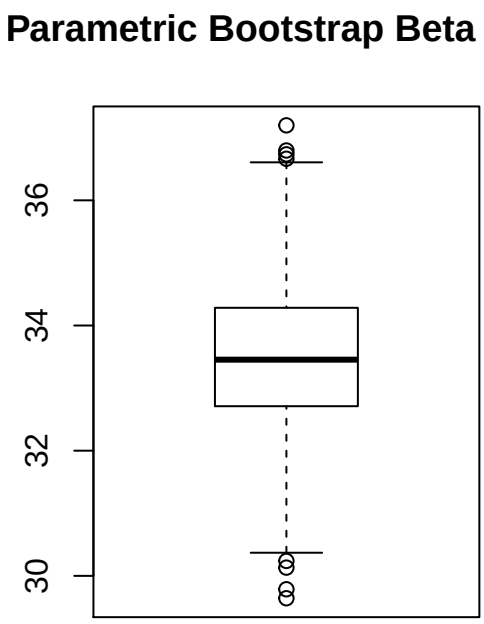
```r
# draw box plots
par(mfrow=c(1,2))
boxplot(res.boot[,1], main="Residuals Bootstrap Beta Hat0")
boxplot(res.boot[,2], main="Residuals Bootstrap Beta Hat1")
```

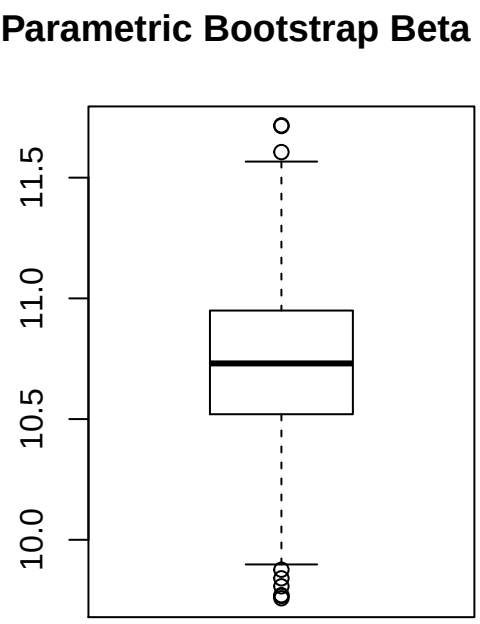



```r
boxplot(pair.boot[,1], main="Data Pairs Bootstrap Beta Hat0")
boxplot(pair.boot[,2], main="Data Pairs Bootstrap Beta Hat1")
```




```r
boxplot(para.boot[,1], main="Parametric Bootstrap Beta Hat0")
boxplot(para.boot[,2], main="Parametric Bootstrap Beta Hat1")
```






```
par(mfrow=c(1,1))
```

From three boxplots, we can see that the estimates of residual bootstrap, data pairs bootstrap and parametric bootstrap are similar to each other. The median of $\hat{beta}_0$ is approximately 33.6 and the median of $\hat{beta}_1$ is around 10.7. Other quantiles are also close although the maximum and minimum value may vary.

```
attach(data.frame(res.boot))
attach(data.frame(pair.boot))
attach(data.frame(para.boot))

# list estimates for the standard errors of beta hats
# i) residuals bootstrap
res.se <- c(res.se0=sd(res.beta0hat),res.se1=sd(res.beta1hat))
res.se
```

```
##   res.se0   res.se1
## 1.1312363 0.3122234
```

```
# ii) data pairs bootstrap
pair.se <- c(pair.se0=sd(pair.beta0hat), pair.se1=sd(pair.beta1hat))
pair.se
```

```
##  pair.se0  pair.se1
## 1.1536800 0.3161013
```

```
# iii) parametric bootstrap
para.se <- c(para.se0=sd(para.beta0hat), para.se1=sd(para.beta1hat))
para.se
```

```
##  para.se0  para.se1
## 1.1891724 0.3252603
```

```r
# least square estimate
ols.se <- c(summary(model)$coefficients[1,2],summary(model)$coefficients[2,2])
ols.se
```

```
## [1] 1.1548735 0.3147534
```

```r
# absolute relative difference bootstrap estimates to least square estimates
abs((res.se-ols.se)/ols.se)
```

```
##     res.se0     res.se1
## 0.020467385 0.008037925
```

```r
abs((pair.se-ols.se)/ols.se)
```

```
##    pair.se0    pair.se1
## 0.001033442 0.004282454
```

```r
abs((para.se-ols.se)/ols.se)
```

```
##   para.se0   para.se1
## 0.02969924 0.03338123
```

Estimates for the standard errors are very close to the least-squares estimates. To compare them, we can refer to the absolute relative difference between them, which are all very small.

2.(c) Obtain confidence intervals

First look at the marginal confidence interval for estimates, $\hat{\beta}_0$ and $\hat{\beta}_1$.

```r
# 95% percentile confidence interval for linear regression bootstrap

# Marginal confidence interval
# 'bootout' is beta0 and beta1 data for each bootstrap sampling
CI <- function(bootout,alpha)
{
  me0 <- qnorm(1-alpha/2)*sd(bootout[,1])
  me1 <- qnorm(1-alpha/2)*sd(bootout[,2])
  #return lower bound and upper bound of CI for beta 0 and beta 1
  c(mean(bootout[,1])-me0,mean(bootout[,1])+me0,
    mean(bootout[,2])-me1,mean(bootout[,2])+me1)
}

alpha <- 0.05
# residual 95% percentile CI
CI(res.boot,alpha)
```

```
## [1] 31.29214 35.72650 10.10599 11.32988
```

```r
# pair 95% percentile CI
CI(pair.boot,alpha)
```

```
## [1] 31.25312 35.77546 10.09456 11.33365
```

```r
# para 95% percentile CI
CI(para.boot,alpha)
```

```
## [1] 31.15512 35.81659 10.09228 11.36727
```

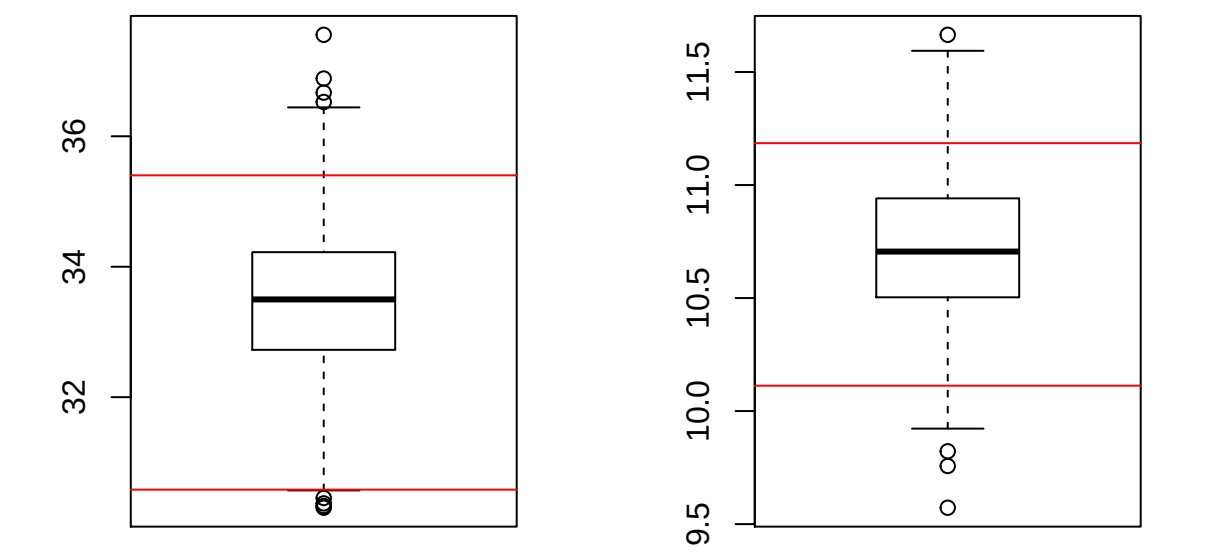
```r
# "bias-corrected and accelerated" bootstrap CI
bcabci <- function (x, g, B = 100, alpha = .05) {
  n <- nrow(x)
  theta.hat <- g(x)
  # residual bootstrap
  theta.boot <- replicate(B,g(cbind(model.matrix(lm(x[,2]~x[,1]))[,-1], fitted(lm(x[,2]~x[,1]))+
                                    residuals(lm(x[,2]~x[,1]))[sample.int(n, replace = TRUE)])))
  # z0 is roughly the median bias of hat(theta)^*, in normal units:
  z0 <- qnorm(mean(theta.boot < theta.hat))
  theta.jack <- sapply(1:n, function (i) g(x[-i, ])) # jacknife
  theta.jack <- mean(theta.jack) - theta.jack
  a <- sum(theta.jack ^ 3) / sum(theta.jack ^ 2) ^ 1.5 / 6 # "acceleration"
  # compute interval
  za <- qnorm(alpha / 2); z1a <- qnorm(1 - alpha / 2)
  alpha1 <- pnorm(z0 + (z0 + za) / (1 - a * (z0 + za)))
  alpha2 <- pnorm(z0 + (z0 + z1a) / (1 - a * (z0 + z1a)))
  list(CI = quantile(theta.boot, c(alpha1, alpha2)), z0 = z0, a = a, coverage = alpha2 - alpha1)
}

BCA0 <- function(x){
  coef(lm(x[,2]~x[,1]))[1]
}

BCA1 <- function(x){
  coef(lm(x[,2]~x[,1]))[2]
}

(BCA.beta0 <- bcabci(faithful,BCA0))
```

```
## $CI
## 1.745076% 96.48743%
##  30.58196  35.40108
##
## $z0
## [1] -0.07526986
##
## $a
## [1] 0.0002426678
##
## $coverage
## [1] 0.9474236
```

```
(BCA.beta1 <- bcabci(faithful,BCA1))
```

```
## $CI
## 1.314457% 95.54815%
##  10.11238  11.18519
##
## $z0
## [1] -0.1256613
##
## $a
## [1] -0.002430717
##
## $coverage
## [1] 0.9423369
```

```
# draw endpoints of the BCA interval in boxplot
par(mfrow=c(1,2))
boxplot(res.boot[,1], main="Residuals Bootstrap Beta Hat0 BCA")
abline(h=BCA.beta0$CI,col="red")
boxplot(res.boot[,2], main="Residuals Bootstrap Beta Hat1 BCA")
abline(h=BCA.beta1$CI,col="red")
```

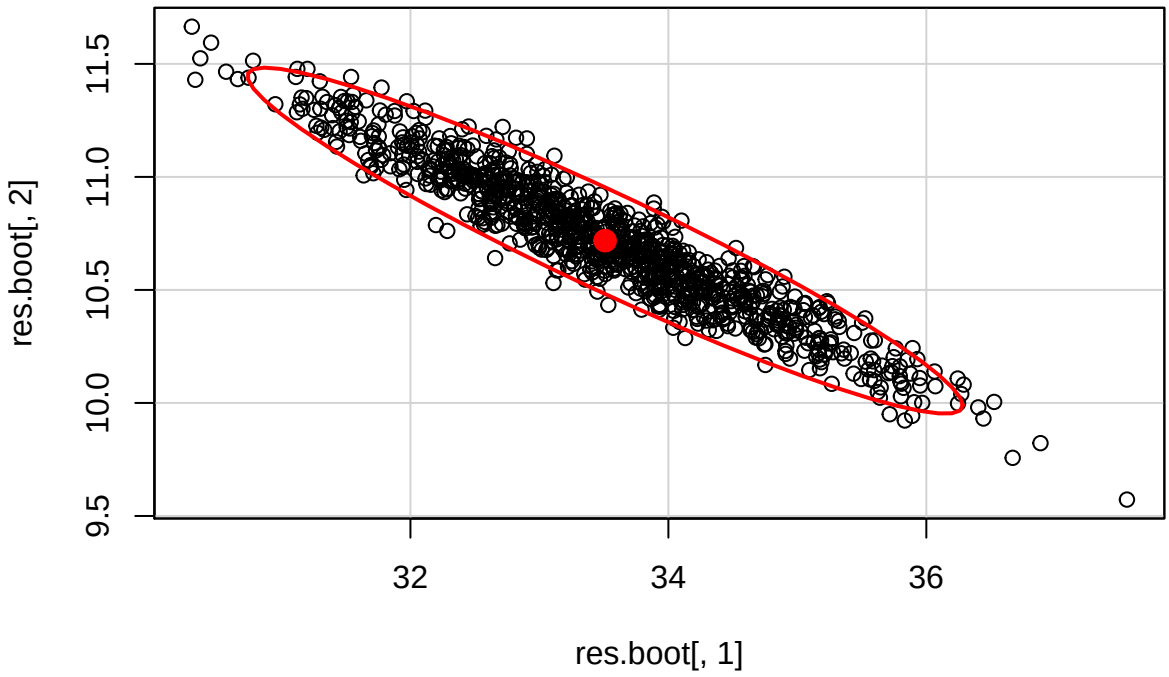



```
par(mfrow=c(1,1))
```

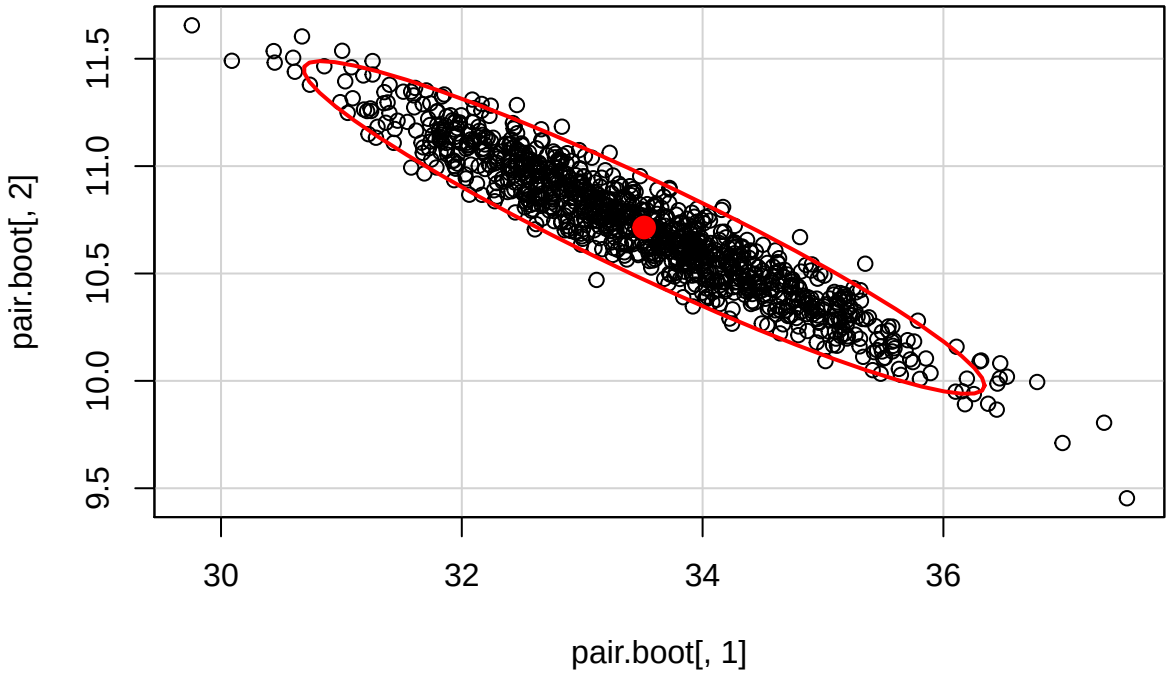
The coverage range of the BCA confidence interval is $30.5819554, 35.4010838$ for $\hat{\beta}_0$ and $10.1123766, 11.1851859$ for $\hat{\beta}_1$. The coverage probability for them are 0.9423369 and 0.9423369

Since estimates $\hat{\beta}$ are corelated with each other, to be more precise, now consider their joint confidence region.

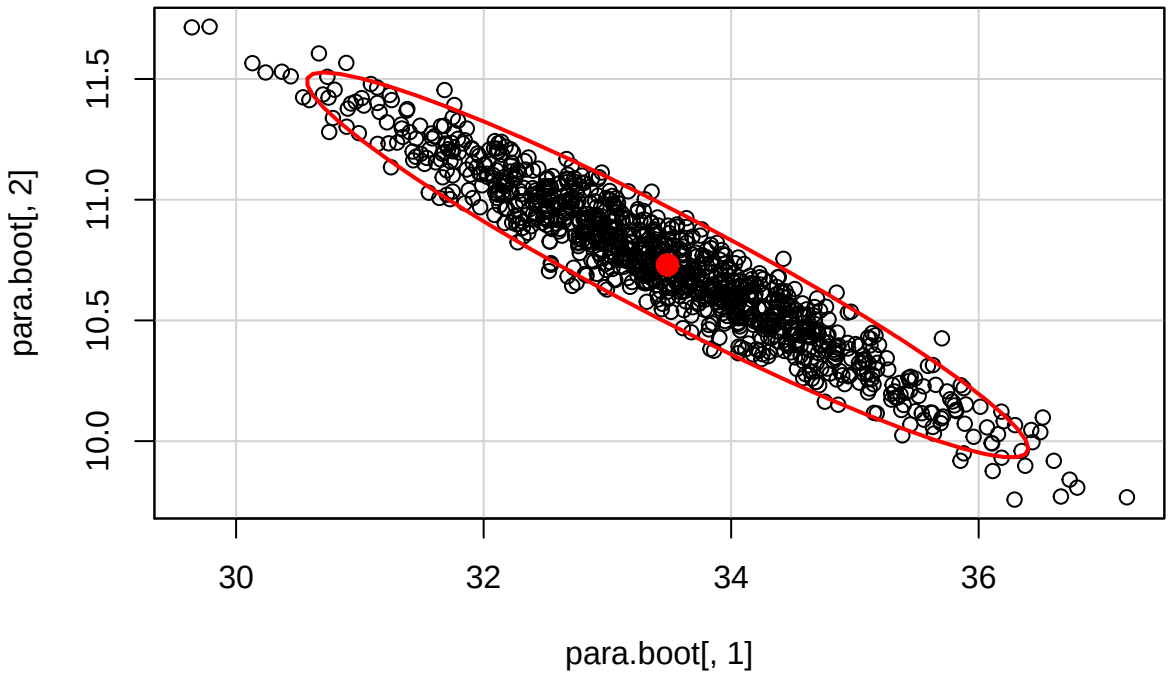
```
# 95% Joint confidence regions
library(car)
dataEllipse(res.boot[,1],res.boot[,2],level = 0.95)
```




```
dataEllipse(pair.boot[,1],pair.boot[,2],levels = 0.95)
```




```
dataEllipse(para.boot[,1],para.boot[,2],levels = 0.95)
```

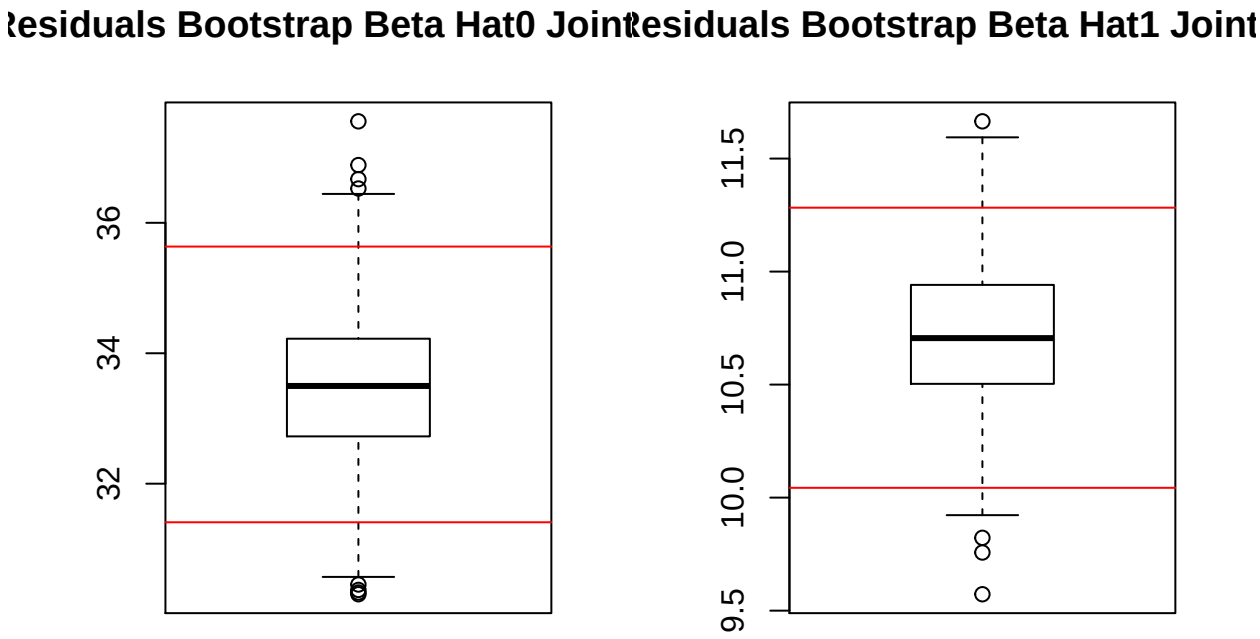



```
# joint "bias-corrected and accelerated" bootstrap CI
bcabci.joint <- function (x, g, B = 100, alpha = .05) {
  n <- nrow(x)
  z <- 0
  theta.hat <- g(x)
  # residual bootstrap
  theta.boot <- matrix(replicate(B,g(cbind(model.matrix(lm(x[,2]~x[,1]))[,-1],
  fitted(lm(x[,2]~x[,1]))+residuals(lm(x[,2]~x[,1]))[sample.int(n, replace = TRUE)]))),nrow=2)
  # z0 is roughly the median bias of hat(theta)^*, in normal units:
  for (i in 1:ncol(matrix(theta.boot,nrow=2)))
  {
    z[i] <- mean(matrix(theta.boot,nrow=2)[,i]<theta.hat)
  }
  z0 <- qnorm(mean(z))
  theta.jack <- sapply(1:n, function (i) g(x[-i, ])) # jacknife
  theta.jack1 <- mean(theta.jack[1,]) - theta.jack[1,]
  theta.jack2 <- mean(theta.jack[2,]) - theta.jack[2,]
  a1 <- sum(theta.jack1 ^ 3) / sum(theta.jack1 ^ 2) ^ 1.5 / 6 # "acceleration"
  a2 <- sum(theta.jack2 ^ 3) / sum(theta.jack2 ^ 2) ^ 1.5 / 6 # "acceleration"
  # compute interval
  za <- qnorm(alpha / 2)
  z1a <- qnorm(1 - alpha / 2)
  alpha11 <- pnorm(z0 + (z0 + za) / (1 - a1 * (z0 + za)))
  alpha21 <- pnorm(z0 + (z0 + z1a) / (1 - a1 * (z0 + z1a)))
  alpha12 <- pnorm(z0 + (z0 + za) / (1 - a2 * (z0 + za)))
  alpha22 <- pnorm(z0 + (z0 + z1a) / (1 - a2 * (z0 + z1a)))
  list(CI1 = quantile(theta.boot[1,], c(alpha11, alpha21)), z0 = z0, a1 = a1,
       coverage1 = alpha21 - alpha11, CI2 = quantile(theta.boot[2,], c(alpha12, alpha22)),
       z0 = z0, a1 = a2, coverage2 = alpha22 - alpha12)
}
BCA <- function(x){
  coef(lm(x[,2]~x[,1]))
```

```
}
(BCA.joint <- bcabci.joint(faithful,BCA))
```

```
## $CI1
## 2.505451% 97.50545%
##  31.40865  35.63560
##
## $z0
## [1] 0
##
## $a1
## [1] 0.0002426678
##
## $coverage1
## [1] 0.95
##
## $CI2
## 2.445668% 97.44519%
##  10.04335  11.28262
##
## $z0
## [1] 0
##
## $a1
## [1] -0.002430717
##
## $coverage2
## [1] 0.9499952
```

```
# draw endpoints of the BCA interval in boxplot
par(mfrow=c(1,2))
boxplot(res.boot[,1], main="Residuals Bootstrap Beta Hat0 Joint BCA")
abline(h=BCA.joint$CI1,col="red")
boxplot(res.boot[,2], main="Residuals Bootstrap Beta Hat1 Joint BCA")
abline(h=BCA.joint$CI2,col="red")
```




```
par(mfrow=c(1,1))
```

The coverage range of the joint BCA confidence interval is $31.4086535, 35.6355972$ for $\hat{\beta}_0$ and $10.0433491, 11.2826234$ for $\hat{\beta}_1$. The coverage probability for them are 0.95 and 0.9499952

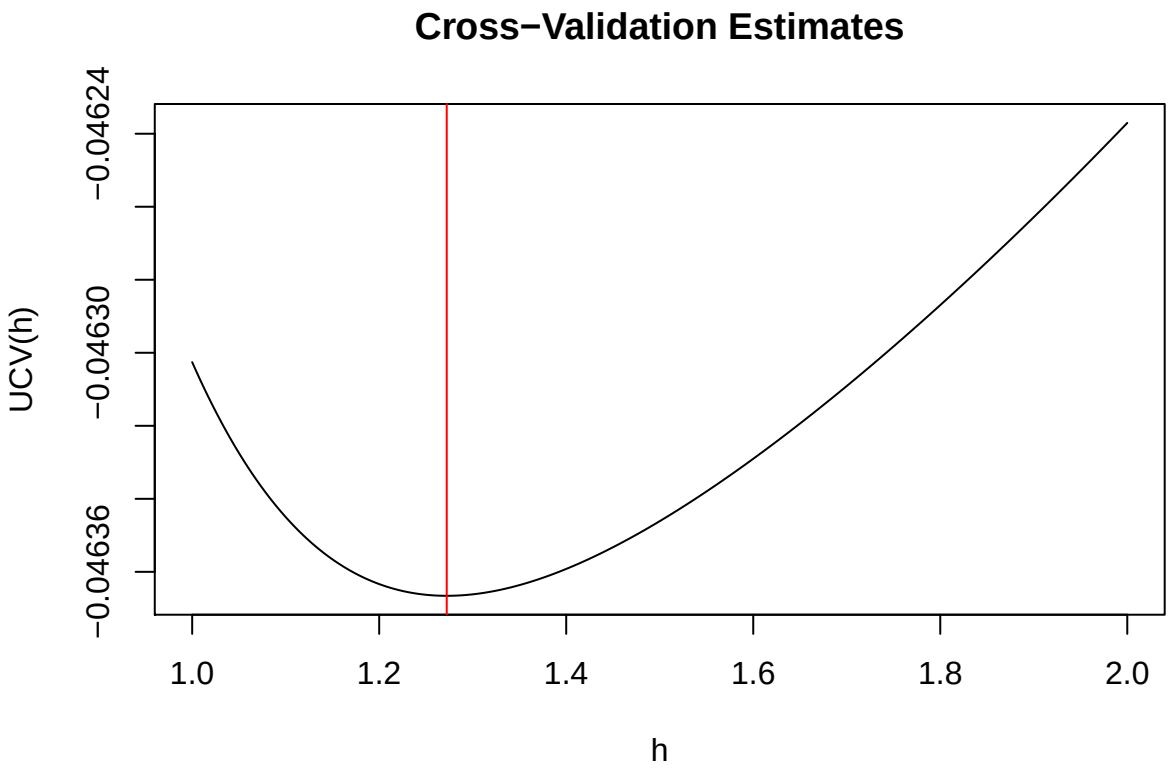
2.(d) Estimate distribution of residuals

Start by finding the best bandwidth using cross-validation:

```
# unbiased CV: `h` is bandwidth, `x` is data, `K` is kernel,
# `w` is kernel width in bandwidth units (e.g., w = 3 for Gaussian kernel),
# and `n` is integral grid resolution (#subdivisions)
ucv <- function (h, x, K = dnorm, w = 3, n = 100) {
  fhat2 <- function (t) (mean(K((t - x) / h)) / h) ^ 2
  fhat2vec <- function (xs) sapply(xs, fhat2)
  from <- min(x) - w * h; to <- max(x) + w * h
  J <- integrate(fhat2vec, from, to, subdivisions = n)$value # R(fhat)
  J - 2 * mean(sapply(seq_along(x),
                      function (i) mean(K((x[i] - x[-i]) / h)) / h))
}

# find bandwidth h.star
hs <- 2^seq(0, 1, length = 1000) # h: (1,2)
uh <- sapply(hs, ucv, residuals(model),n=100)
plot(hs, uh, type = 'l', xlab='h', ylab='UCV(h)',main = "Cross-Validation Estimates")
h.star <- hs[which.min(uh)]
h.star
```

```
## [1] 1.272219
```

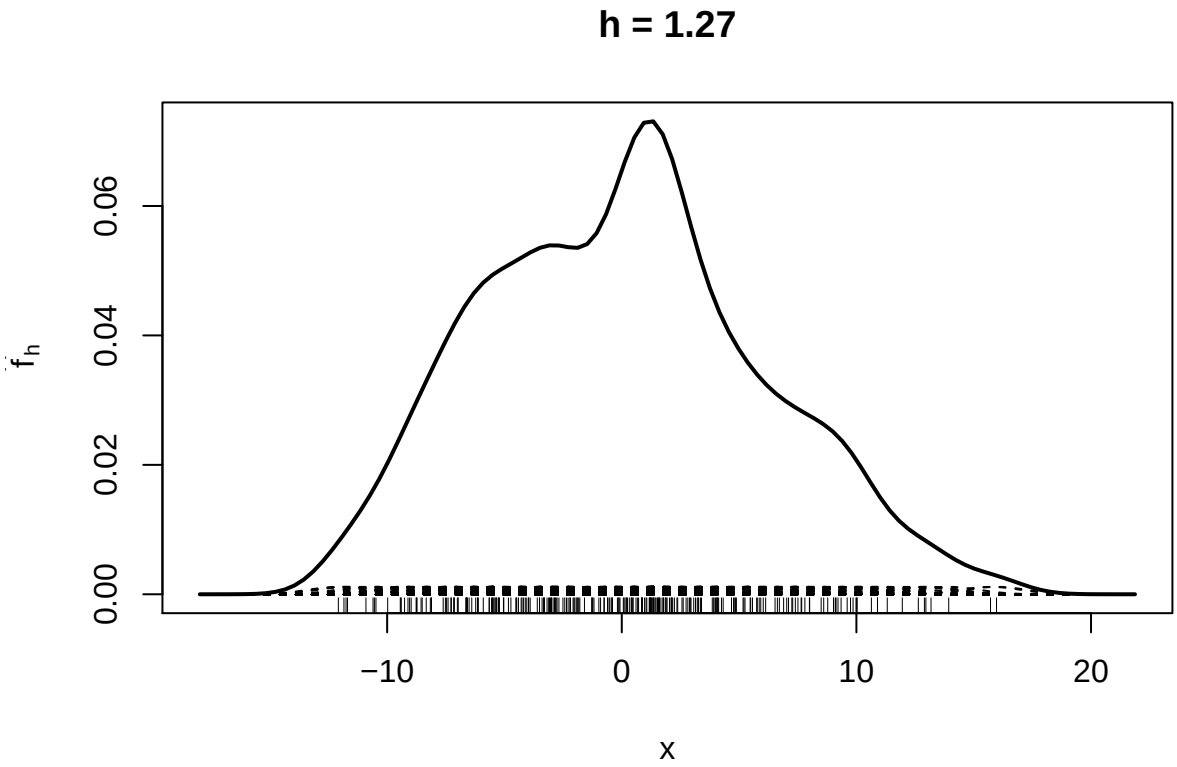
```
abline(v=h.star,col="red")
```



In the 'Cross-Validation Estimates' plot, the integrated square error J (modulo a constant) reach its minimum value when h.star=1.272219.

Then, estimate the distribution of residuals using a normal kernel with bandwidth h.star:

```
# estimate the distribution
kdensity <- function (x, K) {
  function (t, h) # f^_h(t)
    mean(K((t - x) / h)) / h
}

plot.density <- function (x, h, K = dnorm, ns = 100) {
  n <- length(x)
  f <- kdensity(x, K)
  t <- seq(min(x) - sd(x), max(x) + sd(x), length = ns)
  y <- sapply(t, f, h)
  plot(t, y, ylim = c(0, max(y)), type = 'l',
       main = paste0('h = ', sprintf('%.2f', h)),
       lwd = 2, xlab = 'x', ylab = expression(hat(f)[h]))
  for (xi in x)
    lines(t, K((t - xi) / h) / (n * h), lty = 2)
  rug(x)
}

plot.density(residuals(model),h.star)
```

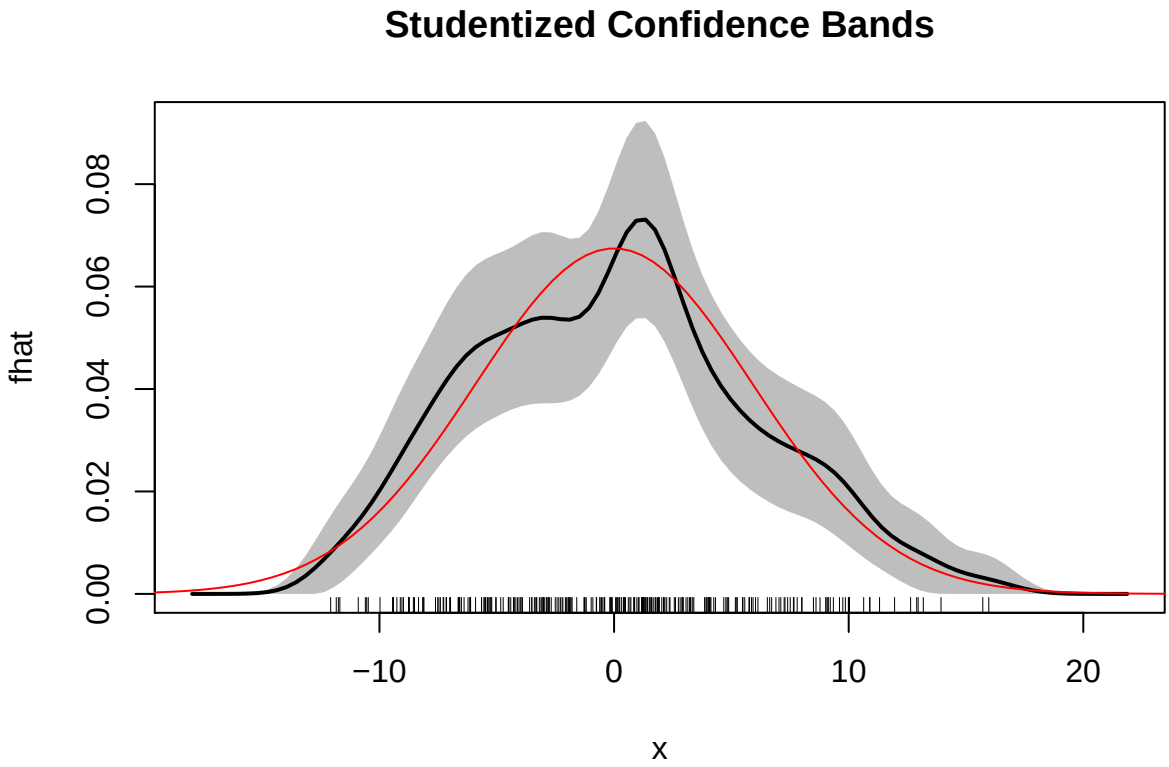


The plot is the plot of density estimated.

2.(e) density "studentized" confidence bands

```
density.ci.student <- function (x, h, alpha = .05, K = dnorm, w = 3, n = 100) {
  xs <- seq(min(x) - sd(x), max(x) + sd(x), length = n) # interval of interest
  m <- (20+20) / (w * h) # independent of n, used to define q; range (-20,20)
  q <- qnorm(.5 * (1 + (1 - alpha) ^ (1 / m)))
  fhat <- l <- u <- numeric(n)
  for (i in seq_along(xs)) {
    y <- K((xs[i] - x) / h) / h
    se <- sd(y) / sqrt(length(x))
    fhat[i] <- fx <- mean(y)
    l[i] <- max(fx - q * se, 0); u[i] <- fx + q * se
  }

  plot(xs, fhat, type = 'n', ylim = range(l, u), xlab = 'x',main = "Studentized Confidence Bands")
  polygon(c(xs, rev(xs)), c(l, rev(u)), col = 'gray', border = F)
  lines(xs, fhat, lwd=2)
  rug(x)
}

# plot 95% "studentized" confidence bands
density.ci.student(residuals(model),h.star)

# plot the normal density
x <- seq(-4, 4, length = 100) * sigma.hat + 0
```

```
y <- dnorm(x, 0, sigma.hat)
lines(x,y,col="red")
```



For most parts, the normal density fit inside the confidence bands except for the tail on the very left. For the tail on the left, the normal density is above the confidence bands.

When we do a parametric bootstrap, we sample from the error that follows a normal distribution, which corresponds to sample on the red line in the plot. And when we do a residual bootstrap, we sample from residuals which is from the confidence bands in the plot. Since most part of the normal density is inside the confidence bands, the sample should be similar between parametric bootstrap and nonparametric bootstrap. So the parametric and nonparametric confidence interval should also be similar as we can see from the results in part (c).