

1.4 Bootstrap CIs

We will look at five different ways to create confidence intervals using the bootstrap and discuss which to use when.

1. Percentile Bootstrap CI
2. Basic Bootstrap CI
3. Standard Normal Bootstrap CI
4. Bootstrap t
5. Accelerated Bias-Corrected (BCa)

Key ideas:

1.4.1 Percentile Bootstrap CI

Let $\hat{\theta}^{(1)}, \dots, \hat{\theta}^{(B)}$ be bootstrap replicates and let $\hat{\theta}_{\alpha/2}$ be the $\alpha/2$ quantile of $\hat{\theta}^{(1)}, \dots, \hat{\theta}^{(B)}$.

Then, the $100(1 - \alpha)\%$ Percentile Bootstrap CI for θ is

In R, if `bootstrap.reps = c($\hat{\theta}^{(1)}, \dots, \hat{\theta}^{(B)}$)`, the percentile CI is

```
quantile(bootstrap.reps, c(alpha/2, 1 - alpha/2))
```

Assumptions/usage

1.4.2 Basic Bootstrap CI

The $100(1 - \alpha)\%$ Basic Bootstrap CI for θ is

Assumptions/usage

1.4.3 Standard Normal Bootstrap CI

From the CLT,

So, the $100(1 - \alpha)\%$ Standard Normal Bootstrap CI for θ is

Assumptions/usage

1.4.4 Bootstrap t CI (Studentized Bootstrap)

Even if the distribution of $\hat{\theta}$ is Normal and $\hat{\theta}$ is unbiased for θ , the Normal distribution is not exactly correct for z .

Additionally, the distribution of $\hat{se}(\hat{\theta})$ is unknown.

⇒ The bootstrap t interval does not use a Student t distribution as the reference distribution, instead we estimate the distribution of a “ t type” statistic by resampling.

The $100(1 - \alpha)\%$ Bootstrap t CI is

Overview

To estimate the “ t style distribution” for $\hat{\theta}$,

Assumptions/usage

1.4.5 BCa CIs

Modified version of percentile intervals that adjusts for bias of estimator and skewness of the sampling distribution.

This method automatically selects a transformation so that the normality assumption holds.

Idea:

The BCa method uses bootstrapping to estimate the bias and skewness then modifies which percentiles are chosen to get the appropriate confidence limits for a given data set.

In summary,

Your Turn

We will consider a telephone repair example from Hesterberg (2014). Verizon has repair times, with two groups, CLEC and ILEC, customers of the “Competitive” and “Incumbent” local exchange carrier.

```
library(resample) # package containing the data
```

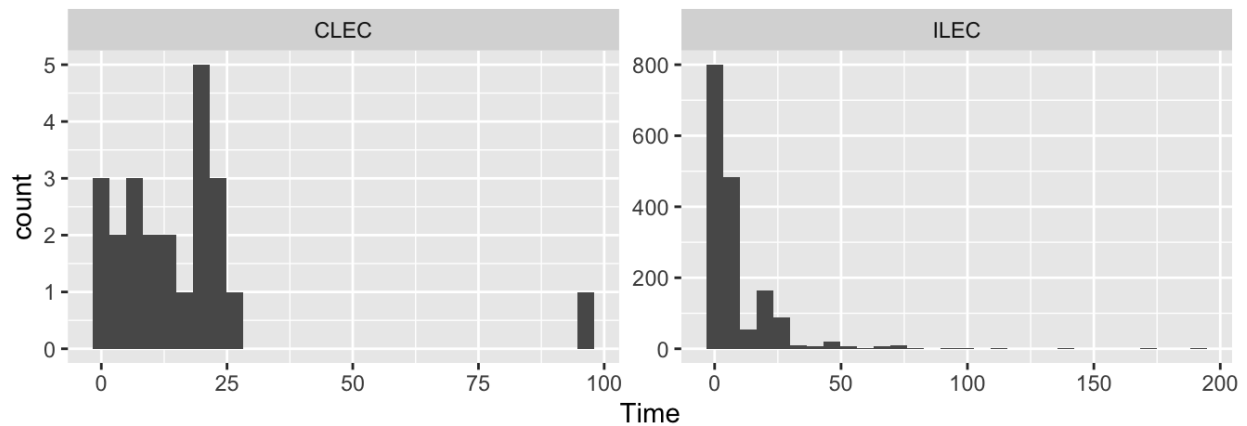
```
data(Verizon)
head(Verizon)
```

```
##      Time Group
## 1 17.50  ILEC
## 2  2.40  ILEC
## 3  0.00  ILEC
## 4  0.65  ILEC
## 5 22.23  ILEC
## 6  1.20  ILEC
```

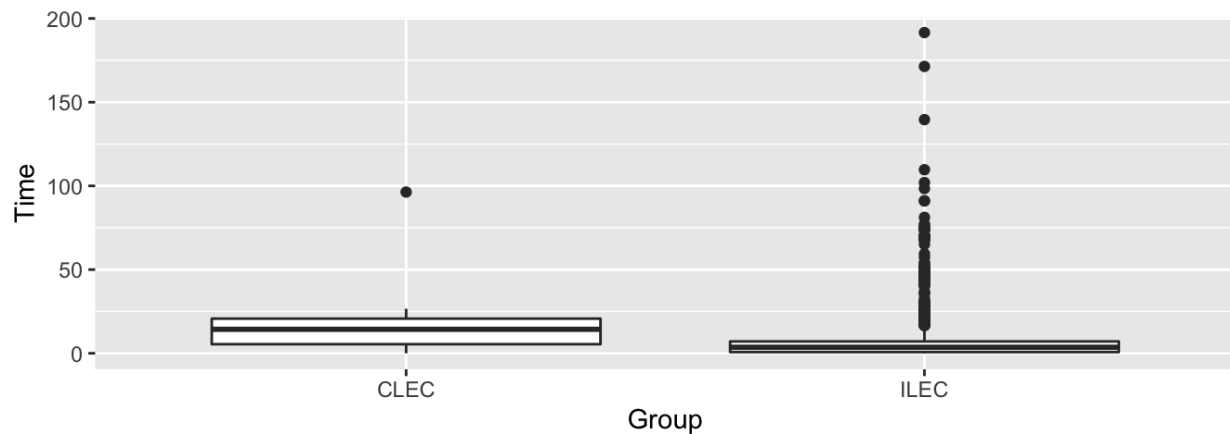
```
Verizon %>%
  group_by(Group) %>%
  summarize(mean = mean(Time), sd = sd(Time), min = min(Time), max =
max(Time)) %>%
  kable()
```

Group	mean	sd	min	max
CLEC	16.509130	19.50358	0	96.32
ILEC	8.411611	14.69004	0	191.60

```
ggplot(Verizon) +
  geom_histogram(aes(Time)) +
  facet_wrap(~Group, scales = "free")
```

```
ggplot(Verizon) +
  geom_boxplot(aes(Group, Time))
```



1.5 Bootstrapping CIs

There are many bootstrapping packages in R, we will use the `boot` package. The function `boot` generates R resamples of the data and computes the desired statistic(s) for each sample. This function requires 3 arguments:

1. `data` = the data from the original sample (data.frame or matrix).
2. `statistic` = a function to compute the statistic from the data where the first argument is the data and the second argument is the indices of the observations in the bootstrap sample.
3. R = the number of bootstrap replicates.

```

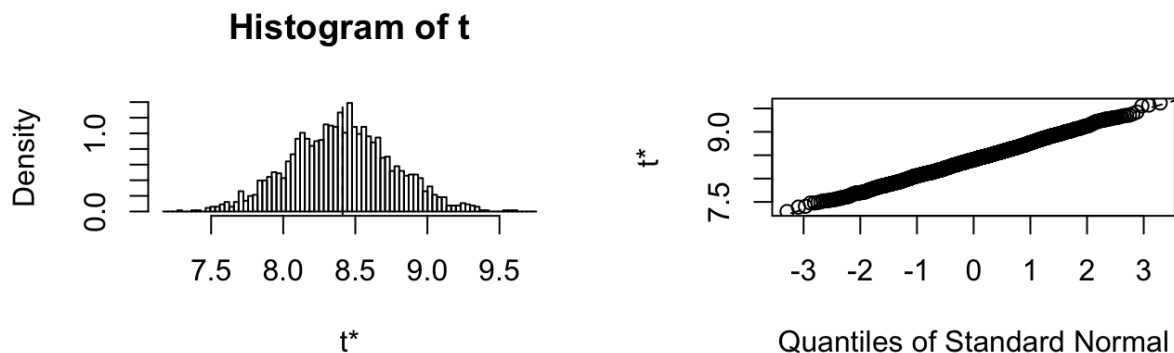
library(boot) # package containing the bootstrap function

mean_func <- function(x, idx) {
  mean(x[idx])
}

ilec_times <- Verizon[Verizon$Group == "ILEC",]$Time
boot.ilec <- boot(ilec_times, mean_func, 2000)

plot(boot.ilec)

```



If we want to get Bootstrap CIs, we can use the `boot.ci` function to generate the 5 different nonparametric bootstrap confidence intervals.

```

boot.ci(boot.ilec, conf = .95, type = c("perc", "basic", "norm",
    "bca"))

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.ilec, conf = 0.95, type = c("perc", "basic",

```

```

##      "norm", "bca"))
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 7.719,  9.114 )   ( 7.709,  9.119 )
##
## Level      Percentile          BCa
## 95%   ( 7.704,  9.114 )   ( 7.752,  9.164 )
## Calculations and Intervals on Original Scale

## we can do some of these on our own
## normal
mean(boot.ilec$t) + c(-1, 1)*qnorm(.975)*sd(boot.ilec$t)

## [1] 7.709670 9.104182

## normal is bias corrected
2*mean(ilec_times) - (mean(boot.ilec$t) - c(-1,
  1)*qnorm(.975)*sd(boot.ilec$t))

## [1] 7.719039 9.113551

## percentile
quantile(boot.ilec$t, c(.025, .975))

##      2.5%      97.5%
## 7.707656 9.111150

## basic
2*mean(ilec_times) - quantile(boot.ilec$t, c(.975, .025))

##      97.5%      2.5%
## 7.712071 9.115565

```

To get the studentized bootstrap CI, we need our statistic function to also return the variance of $\hat{\theta}$.

```
mean_var_func <- function(x, idx) {  
  c(mean(x[idx]), var(x[idx])/length(idx))  
}  
  
boot.ilec_2 <- boot(ilec_times, mean_var_func, 2000)  
boot.ci(boot.ilec_2, conf = .95, type = "stud")  
  
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 2000 bootstrap replicates  
##  
## CALL :  
## boot.ci(boot.out = boot.ilec_2, conf = 0.95, type = "stud")  
##  
## Intervals :  
## Level      Studentized  
## 95%      ( 7.733,  9.231 )  
## Calculations and Intervals on Original Scale
```

Which CI should we use?

1.6 Bootstrapping for the difference of two means

Given iid draws of size n and m from two populations, to compare the means of the two groups using the bootstrap,

The function `two.boot` in the `simpleboot` package is used to bootstrap the difference between univariate statistics. Use the bootstrap to compute the shape, bias, and bootstrap sample error for the samples from the Verizon data set of CLEC and ILEC customers.

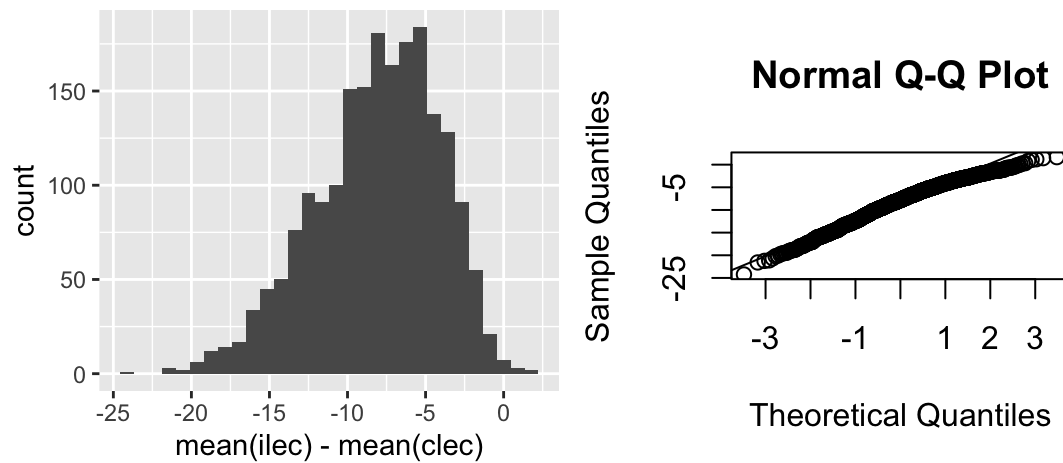
```
library(simpleboot)

clec_times <- Verizon[Verizon$Group == "CLEC",]$Time

diff_means.boot <- two.boot(ilec_times, clec_times, "mean", R = 2000)

ggplot() +
  geom_histogram(aes(diff_means.boot$t)) +
  xlab("mean(ilec) - mean(clec)")

qqnorm(diff_means.boot$t)
qqline(diff_means.boot$t)
```



Your turn: estimate the bias and se of the sampling distribution

Which confidence intervals should we use?

Your turn: get the chosen CI using boot.ci

Is there evidence that

$$H_0 : \mu_1 - \mu_2 = 0$$

$$H_a : \mu_1 - \mu_2 < 0$$

is rejected?

2 Parametric Bootstrap

In a **nonparametric bootstrap**, we

In a **parametric bootstrap**,

For both methods,

2.1 Bootstrapping for linear regression

Consider the regression model $Y_i = \mathbf{x}_i^T \boldsymbol{\beta} + \epsilon_i, i = 1, \dots, n$ with $\epsilon_i \stackrel{iid}{\sim} N(0, \sigma^2)$.

Two approaches for bootstrapping linear regression models –

- 1.
- 2.

2.1.1 Bootstrapping the residuals

1. Fit the regression model using the original data
2. Compute the residuals from the regression model,

$$\hat{\epsilon}_i = y_i - \hat{y}_i = y_i - \mathbf{x}_i^T \hat{\boldsymbol{\beta}}, \quad i = 1, \dots, n$$

3. Sample $\hat{\epsilon}_1^*, \dots, \hat{\epsilon}_n^*$ with replacement from $\hat{\epsilon}_1, \dots, \hat{\epsilon}_n$.
4. Create the bootstrap sample

$$y_i^* = \mathbf{x}_i^T \hat{\boldsymbol{\beta}} + \epsilon_i^*, \quad i = 1, \dots, n$$

5. Estimate $\hat{\boldsymbol{\beta}}^*$
6. Repeat steps 2-4 B times to create B bootstrap estimates of $\hat{\boldsymbol{\beta}}$.

Assumptions:

2.1.2 Paired bootstrapping

Resample $z_i^* = (y_i, \mathbf{x}_i)^*$ from the empirical distribution of the pairs (y_i, \mathbf{x}_i) .

Assumptions:

2.1.3 Which to use?

1. Standard inferences -
2. Bootstrapping the residuals -
3. Paired bootstrapping -

Your Turn

This data set is the Puromycin data in R. The goal is to create a regression model about the rate of an enzymatic reaction as a function of the substrate concentration.

```
head(Puromycin)
```

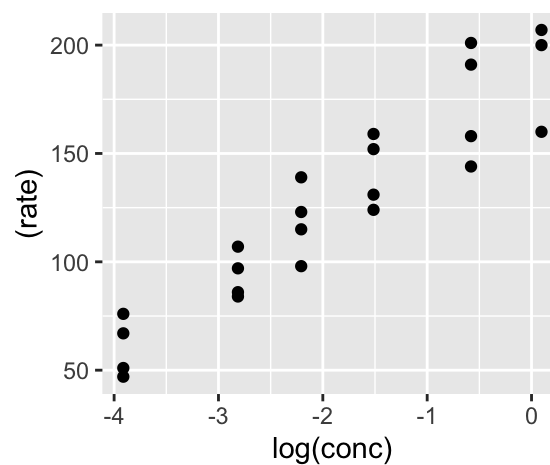
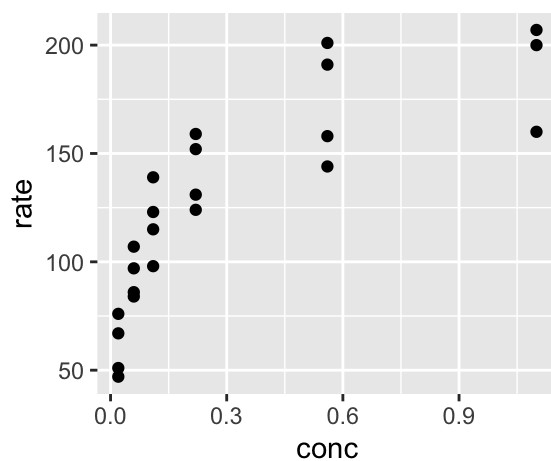
```
##   conc rate  state
## 1 0.02   76 treated
## 2 0.02   47 treated
## 3 0.06   97 treated
## 4 0.06  107 treated
## 5 0.11  123 treated
## 6 0.11  139 treated
```

```
dim(Puromycin)
```

```
## [1] 23  3
```

```
ggplot(Puromycin) +
  geom_point(aes(conc, rate))
```

```
ggplot(Puromycin) +
  geom_point(aes(log(conc), (rate)))
```



2.1.4 Standard regression

```

m0 <- lm(rate ~ conc, data = Puromycin)
plot(m0)
summary(m0)

##
## Call:
## lm(formula = rate ~ conc, data = Puromycin)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -49.861 -15.247  -2.861  15.686  48.054
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    93.92      8.00    11.74 1.09e-10 ***
## conc          105.40     16.92     6.23 3.53e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 28.82 on 21 degrees of freedom
## Multiple R-squared:  0.6489, Adjusted R-squared:  0.6322
## F-statistic: 38.81 on 1 and 21 DF, p-value: 3.526e-06

confint(m0)

##              2.5 %    97.5 %
## (Intercept) 77.28643 110.5607
## conc        70.21281 140.5832

m1 <- lm(rate ~ log(conc), data = Puromycin)
plot(m1)
summary(m1)

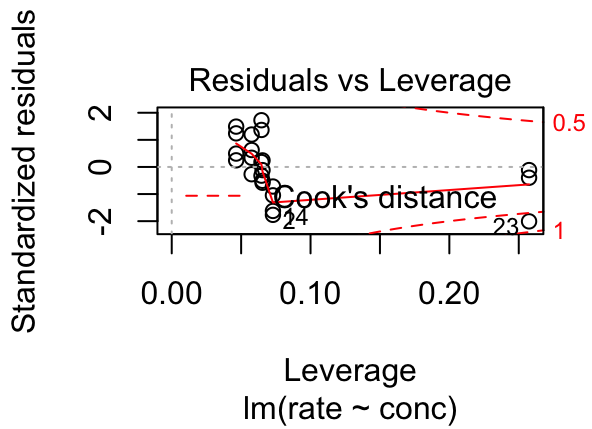
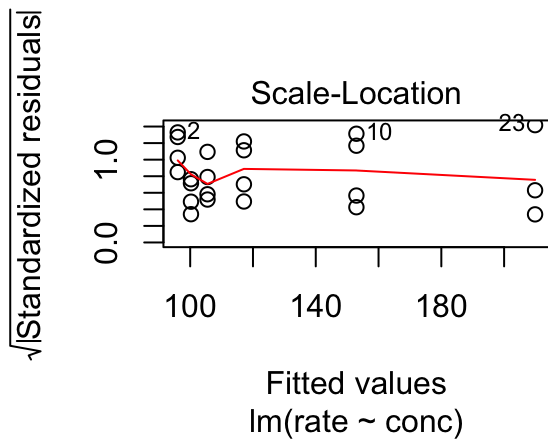
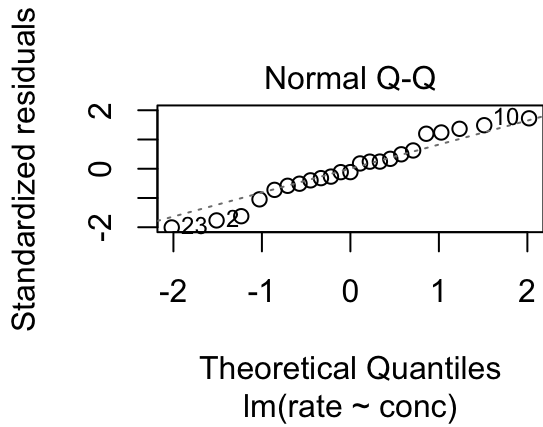
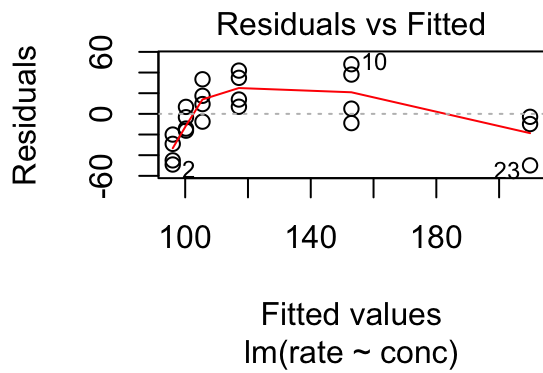
##
## Call:
## lm(formula = rate ~ log(conc), data = Puromycin)
##

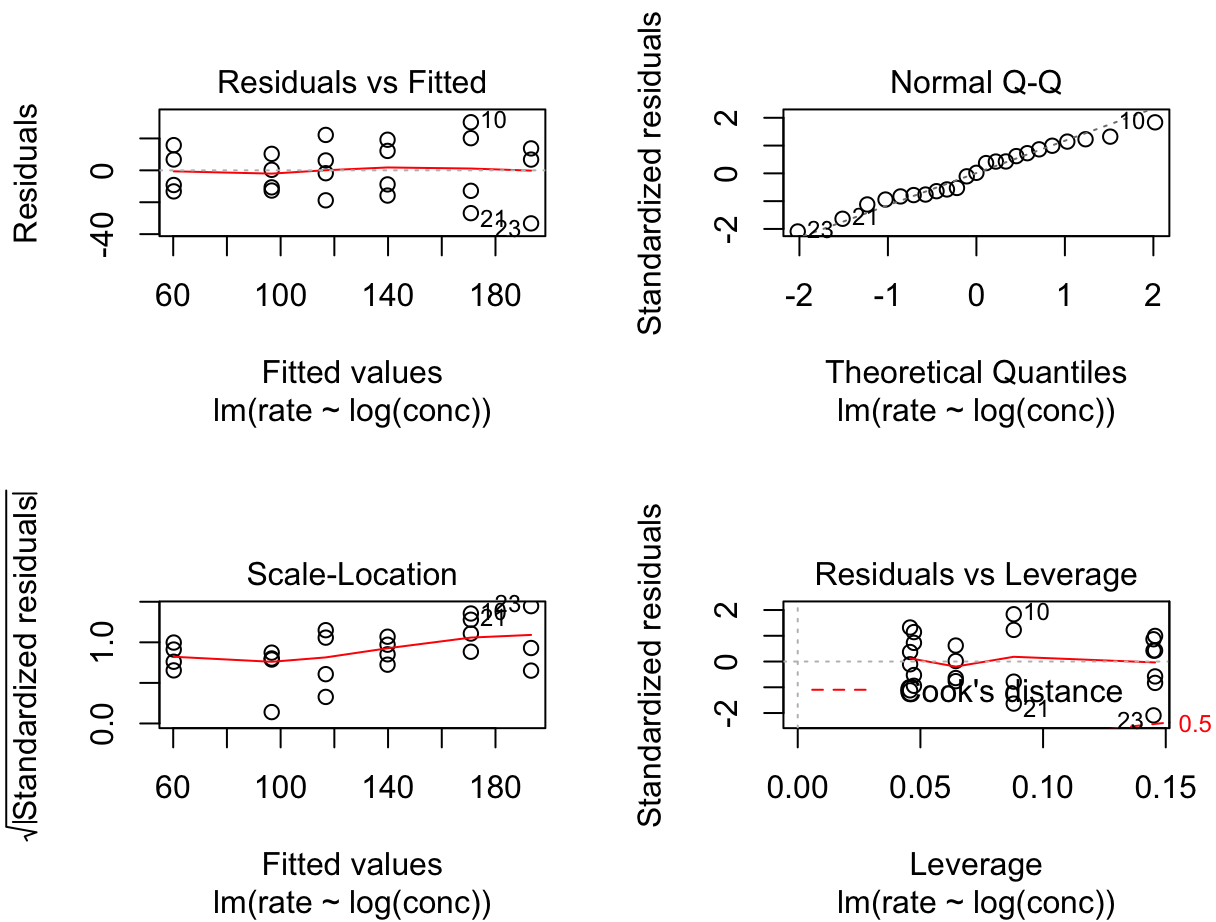
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -33.250 -12.753   0.327  12.969  30.166
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  190.085      6.332   30.02 < 2e-16 ***
## log(conc)     33.203      2.739   12.12 6.04e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.2 on 21 degrees of freedom
## Multiple R-squared:  0.875, Adjusted R-squared:  0.869
## F-statistic: 146.9 on 1 and 21 DF, p-value: 6.039e-11
```

```
confint(m1)
```

```
##              2.5 %    97.5 %
## (Intercept) 176.91810 203.2527
## log(conc)   27.50665  38.8987
```





2.1.5 Paired bootstrap

```
# Your turn
library(boot)

reg_func <- function(dat, idx) {
  # write a regression function that returns fitted beta
}

# use the boot function to get the bootstrap samples

# examining the bootstrap sampling distribution, make histograms

# get confidence intervals for beta_0 and beta_1 using boot.ci
```

2.1.6 Bootstrapping the residuals

```
# Your turn
library(boot)

reg_func_2 <- function(dat, idx) {
  # write a regression function that returns fitted beta
  # from fitting a y that is created from the residuals

}

# use the boot function to get the bootstrap samples

# examining the bootstrap sampling distribution, make histograms

# get confidence intervals for beta_0 and beta_1 using boot.ci
```