Bootstrap Methods

Typically we use (asymptotic) theory to derive the sampling distribution of a statistic. From the sampling distribution, we can obtain the variance, construct confidence intervals, perform hypothesis tests, and more.

Challenge:

Basic idea of bootstrapping:

“Bootstrap World”

1 Nonparametric Bootstrap

Let $Y_1, \ldots, Y_n \sim F$ with pdf $f(y)$. Recall, the empirical cdf is defined as

Theoretical:

Bootstrap:

The idea behind the nonparametric bootstrap is to sample many data sets from $F_n(y)$, which can be achieved by resampling from the data **with replacement**.

```r
# observed data
x <- c(2, 2, 1, 1, 5, 4, 4, 3, 1, 2)

# create 10 bootstrap samples
x_star <- matrix(NA, nrow = length(x), ncol = 10)
for(i in 1:10) {
  x_star[, i] <- sample(x, length(x), replace = TRUE)
}
x_star
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    2    4    1    2    1    2    3    3     4
##  [2,]    4    4    1    1    1    2    2    1    2     1
##  [3,]    2    2    2    4    5    4    4    5    1     4
##  [4,]    4    4    2    5    2    4    5    5    1     3
##  [5,]    2    1    5    1    3    2    4    2    4     4
##  [6,]    4    4    2    1    4    4    4    3    1     2
##  [7,]    1    1    2    1    2    1    2    2    3     1
##  [8,]    4    4    1    3    3    3    5    1    2     4
##  [9,]    4    1    2    3    2    1    2    1    4     2
## [10,]    3    4    5    1    5    4    5    2    4     1
```

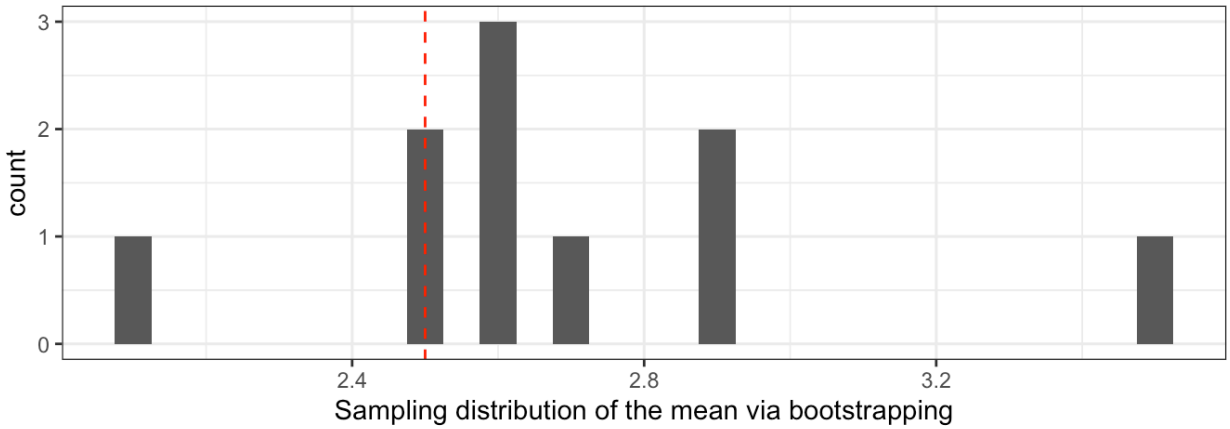
```r
# compare mean of the same to the means of the bootstrap samples
mean(x)
```

```
## [1] 2.5
```

```r
colMeans(x_star)
```

```
##  [1] 2.9 2.7 2.6 2.1 2.9 2.6 3.5 2.5 2.5 2.6
```

```r
ggplot() +
  geom_histogram(aes(colMeans(x_star)), binwidth = .05) +
  geom_vline(aes(xintercept = mean(x)), lty = 2, colour = "red") +
  xlab("Sampling distribution of the mean via bootstrapping")
```



1.1 Algorithm

Goal: estimate the sampling distribution of a statistic based on observed data $x_1, \ldots, x_n$.

Let θ be the parameter of interest and $\hat{\theta}$ be an estimator of θ. Then,

1.2 Justification for iid data

Suppose $Y_1, \ldots, Y_n$ are iid with $\mathrm{E}Y_1 = \mu \in \mathbb{R}$, $\mathrm{Var}(Y_1) = \sigma^2 \in (0, \infty)$. Let's approximate the distribution of $T_n = \sqrt{n}(\bar{Y}_n - \mu)$ via the bootstrap.

Theorem: If $Y_1, Y_2, \ldots$ are iid with $\mathrm{Var}(Y_1) = \sigma^2 \in (0, \infty)$, then
$\sup_{y \in \mathbb{R}} |P(T_n \leq y) - P_*(T_n^* \leq y)| \equiv \Delta_n \to 0$ as $n \to \infty$ almost surely (a.s).

The proof of this theorem requires two facts:

i. (Berry-Esseen Lemma) Let $Y_1, \ldots, Y_n$ be independent with $\mathrm{E}Y_i = 0$ and $\mathrm{E}|Y_i|^3 < \infty$ for $i = 1, \ldots, n$. Let $\sigma_n^2 = n\mathrm{Var}(\bar{Y}_n) = n^{-1} \sum_{i=1}^n \mathrm{E}Y_i^2 > 0$. Then,

$$\sup_{y \in \mathbb{R}} \left| P\left(\frac{\sqrt{n}\bar{Y}_n}{\sigma_n} \leq y \right) - \Phi(y) \right| = \sup_{x \in \mathbb{R}} \left| P\left(\sqrt{n}\bar{Y}_n \leq x \right) - \Phi\left(\frac{x}{\sigma_n} \right) \right| \leq \frac{2.75}{n^{3/2}\sigma_n^3} \sum_{i=1}^n \mathrm{E}|Y_i|^3.$$

ii. (Marcinkiewicz-Zygmund SLLN) Let X_i be a sequence of iid random variables with $\mathrm{E}|X_i|^p < \infty$ for $p \in (0, 2)$. Then, for $S_n = \sum_{i=1}^n X_i$,

$$\frac{1}{n^{1/p}}(S_n - nc) \to 0 \text{ as } n \to \infty \text{ almost surely } (*)$$

for any $c \in \mathbb{R}$ if $p \in (0, 1)$ and for $c = \mathrm{E}X_1$ if $p \in [1, 2)$. If $(*)$ holds for some $c \in \mathbb{R}$, then $\mathrm{E}|X_1|^p < \infty$.

1.3 Properties of Estimators

We can use the bootstrap to estimate different properties of estimators.

1.3.1 Standard Error

Recall $se(\hat{\theta}) = \sqrt{Var(\hat{\theta})}$. We can get a **bootstrap** estimate of the standard error:

1.3.2 Bias

Recall $\text{bias}(\hat{\theta}) = E[\hat{\theta} - \theta] = E[\hat{\theta}] - \theta$. We can get a **bootstrap** estimate of the bias:

Overall, we seek statistics with small se and small bias.

1.4 Sample Size and # Bootstrap Samples

$$n = \text{sample size} \quad \& \quad B = \#\text{ bootstap samples}$$

If n is too small, or sample isn't representative of the population,

Guidelines for B –

Best approach –

Your Turn

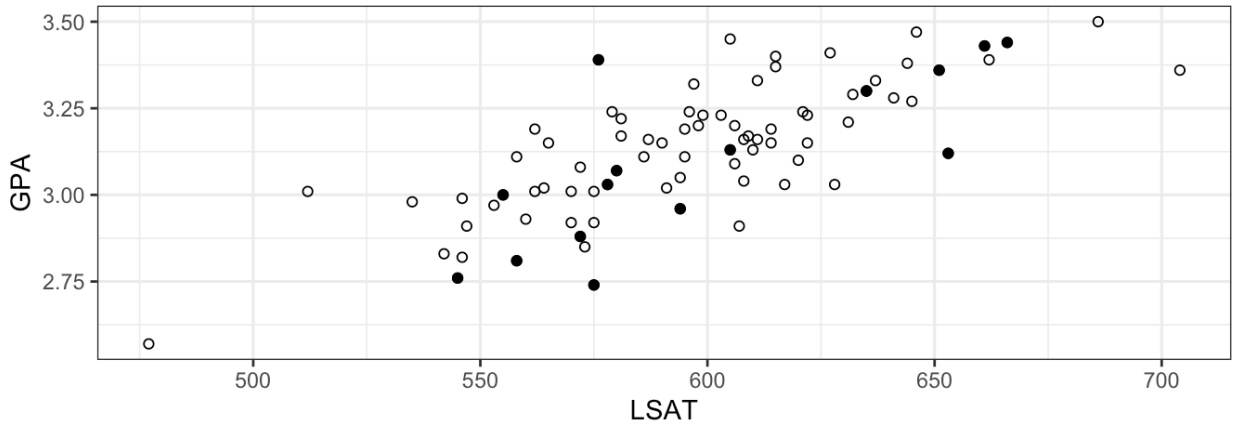
In this example, we explore bootstrapping in the rare case where we know the values for the entire population. If you have all the data from the population, you don't need to bootstrap (or really, inference). It is useful to learn about bootstrapping by comparing to the truth in this example.

In the package `bootstrap` is contained the average LSAT and GPA for admission to the population of 82 USA Law schools (an old data set – there are now over 200 law schools). This package also contains a random sample of size $n = 15$ from this dataset.

```
library(bootstrap)

head(law)
```

```
##   LSAT  GPA
## 1  576 3.39
## 2  635 3.30
## 3  558 2.81
## 4  578 3.03
## 5  666 3.44
## 6  580 3.07
```

```
ggplot() +
  geom_point(aes(LSAT, GPA), data = law) +
  geom_point(aes(LSAT, GPA), data = law82, pch = 1)
```



We will estimate the correlation $\theta = \rho(\text{LSAT}, \text{GPA})$ between these two variables and use a bootstrap to estimate the sample distribution of $\hat{\theta}$.

```
# sample correlation
cor(law$LSAT, law$GPA)
```

```
## [1] 0.7763745
```

```
# population correlation
cor(law82$LSAT, law82$GPA)
```

```
## [1] 0.7599979
```

```
# set up the bootstrap
B <- 200
n <- nrow(law)
r <- numeric(B) # storage

for(b in B) {
  ## Your Turn: Do the bootstrap!
}
```

1. Plot the sample distribution of $\hat{\theta}$. Add vertical lines for the true value θ and the sample estimate $\hat{\theta}$.
2. Estimate $sd(\hat{\theta})$.
3. Estimate the bias of $\hat{\theta}$

1.5 Bootstrap CIs

We will look at five different ways to create confidence intervals using the boostrap 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:

1.5.1 Percentile Bootstrap CI

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

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

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

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

Assumptions/usage

1.5.2 Basic Bootstrap CI

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

Assumptions/usage

1.5.3 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.

$\Rightarrow$ The bootstrap t interval does not use a Student t distribution as the reference distribuion, instead we estimate the distribution of a "t type" statistic by resampling.

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

Overview

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

Assumptions/usage

1.5.4 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.

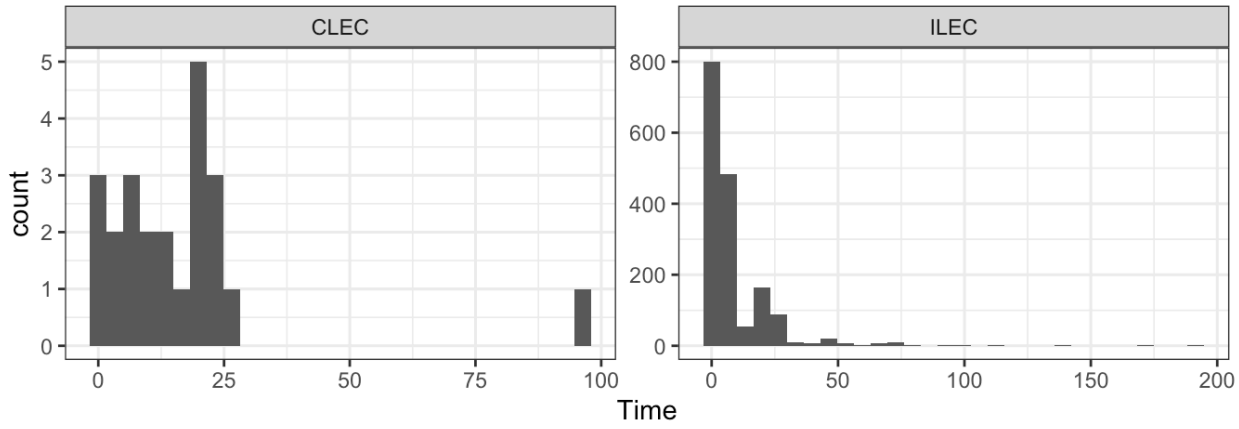
```
library(resample) # package containing the data

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
```

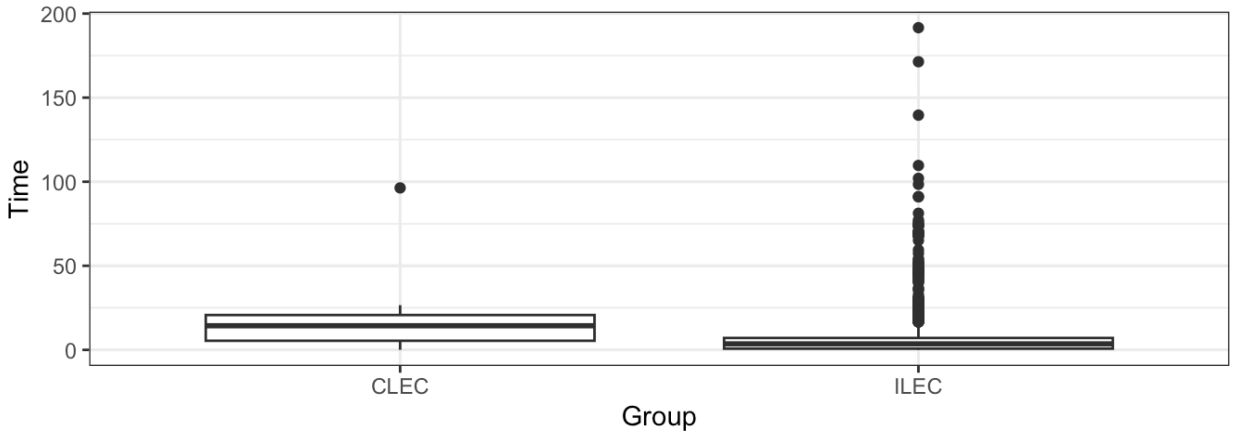
```
Verizon |>
  group_by(Group) |>
  summarize(mean = mean(Time), sd = sd(Time), min = min(Time), max =
        max(Time)) |>
  kable()
```

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))
```

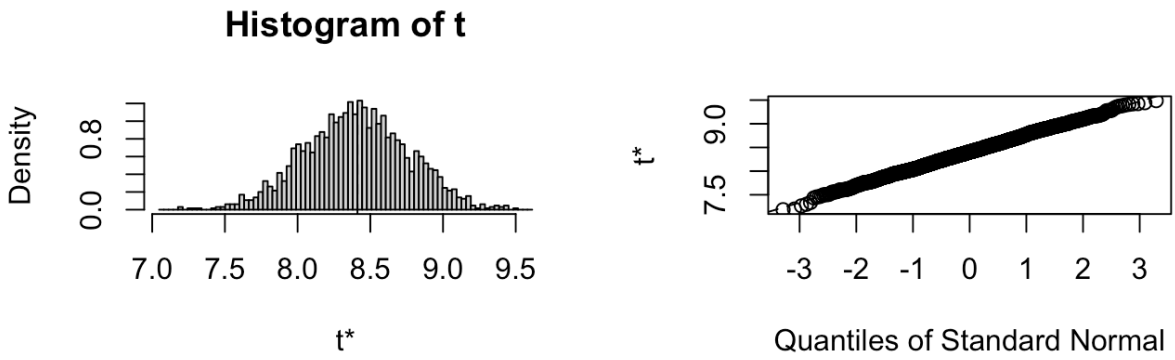


1.6 Bootstrapping CIs

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 obervations in the boostrap sample.
3. R = the number of bootstrap replicates.

```r
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 different nonparametric bootstrap confidence intervals.

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

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.ilec, conf = 0.95, type = c("perc",
"basic",
##     "bca"))
##
## Intervals :
## Level      Basic              Percentile            BCa
## 95%   ( 7.733,  9.110 )   ( 7.714,  9.091 )   ( 7.755,  9.125 )
## Calculations and Intervals on Original Scale
```

```r
## we can do some of these on our own
## percentile
quantile(boot.ilec$t, c(.025, .975))
```

```
##     2.5%    97.5%
## 7.714075 9.084725
```

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

```
##    97.5%     2.5%
## 7.738496 9.109147
```

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

```r
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%   ( 7.728,  9.183 )
## Calculations and Intervals on Original Scale
```

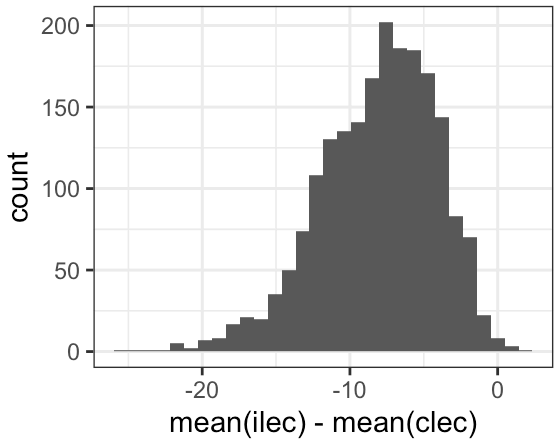
Which CI should we use?

1.7 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)")
```



```
# 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 $Y_i = \boldsymbol{x}_i^T \boldsymbol{\beta} + \epsilon_i, i = 1, \ldots, n$ with $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$.

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,

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

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

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

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

Assumptions:

2.1.2 Paired bootstrapping

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

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.

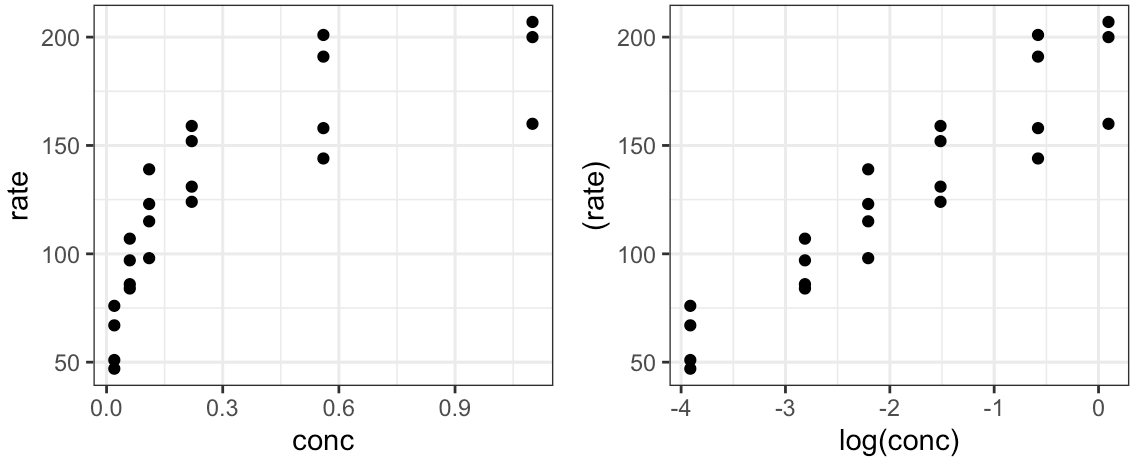
```
head(Puromycin)
```

```
##   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
```

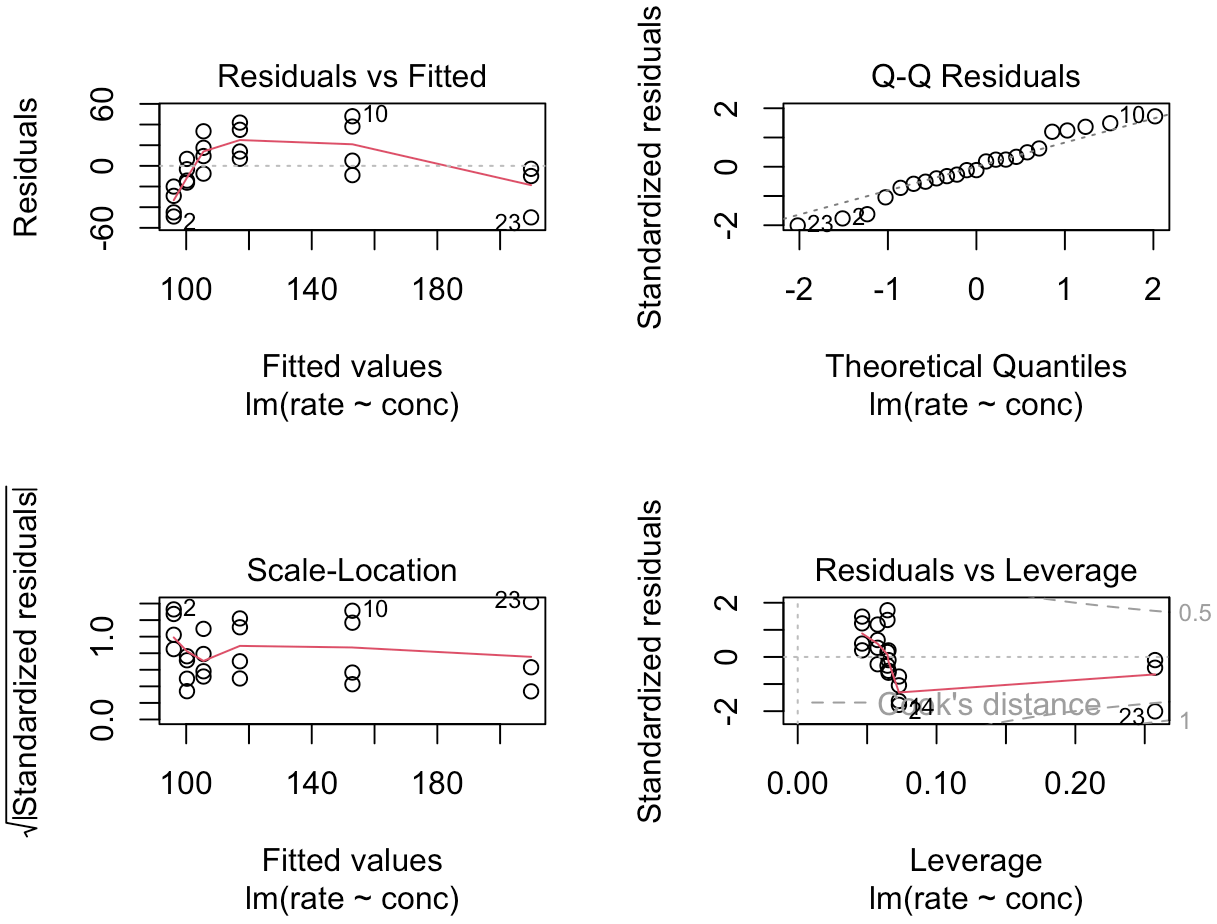
```
m1 <- lm(rate ~ log(conc), data = Puromycin)
plot(m1)
summary(m1)
```

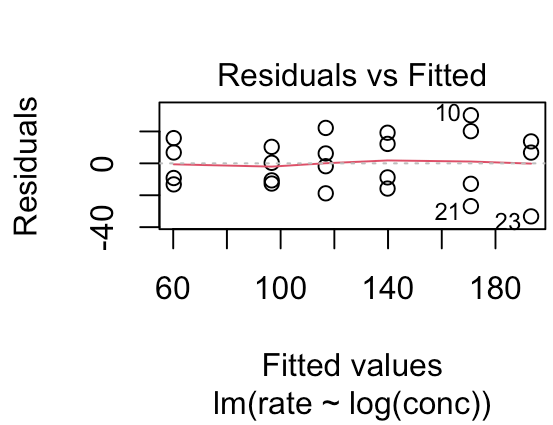
```
##
## Call:
## lm(formula = rate ~ log(conc), data = Puromycin)
```

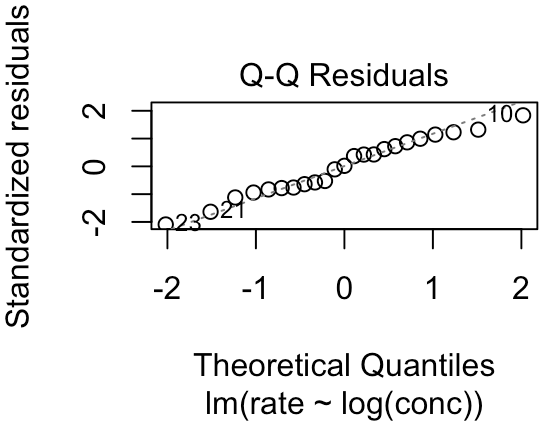
```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.250 -12.753   0.327  12.969  30.166
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  190.085      6.332   30.02  < 2e-16 ***
## log(conc)     33.203      2.739   12.12 6.04e-11 ***
## ---
## 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
```

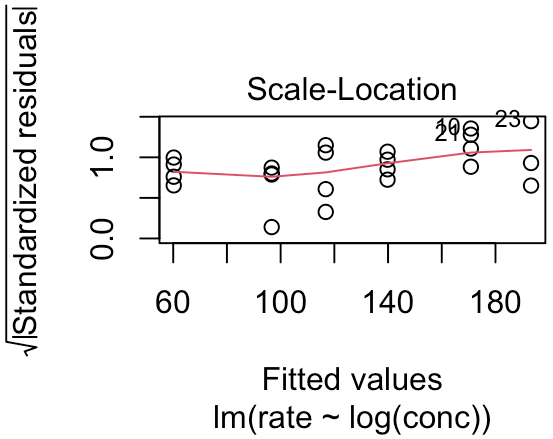
```
confint(m1)
```

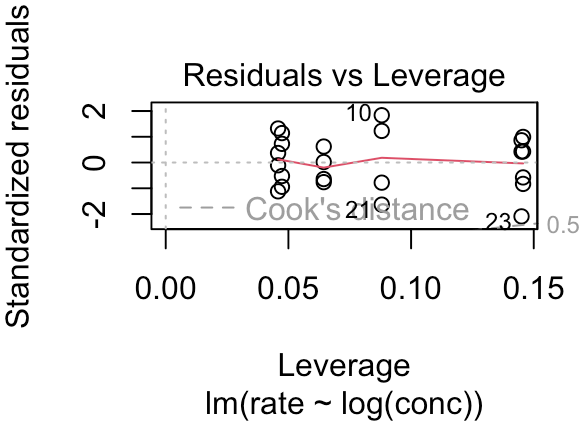
```
##                 2.5 %   97.5 %
## (Intercept) 176.91810 203.2527
## log(conc)    27.50665  38.8987
```


Residuals vs Fitted
Residuals
Fitted values
lm(rate ~ conc)
Q-Q Residuals
Standardized residuals
Theoretical Quantiles
lm(rate ~ conc)
Scale-Location
√|Standardized residuals|
Fitted values
lm(rate ~ conc)
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage
lm(rate ~ conc)


Residuals vs Fitted
Residuals
-40
0
60
100
140
180
10
21
23
Fitted values
lm(rate ~ log(conc))


Q-Q Residuals
Standardized residuals
-2
0
2
-2
-1
0
1
2
10
21
23
Theoretical Quantiles
lm(rate ~ log(conc))


Scale-Location
√|Standardized residuals|
0.0
1.0
60
100
140
180
10
21
23
Fitted values
lm(rate ~ log(conc))


Residuals vs Leverage
Standardized residuals
-2
0
2
10
21
23
Cook's distance
0.5
0.00
0.05
0.10
0.15
Leverage
lm(rate ~ log(conc))

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

# examing 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

# examing the bootstrap sampling distribution, make histograms

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

3 Bootstrapping Dependent Data

Suppose we have dependent data $\boldsymbol{y} = (y_1, \ldots, y_n)$ generated from some unknown distribution $F = F_{\boldsymbol{Y}} = F_{(Y_1,\ldots,Y_n)}$.

Goal:

Challenge:

We will consider 2 approaches

Example 3.1 Suppose we observe a time series $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ which we assume is generated by an AR(1) process, i.e.,

Why not just move forward with our nonparametric bootstrap procedure?

3.1 Model-based approach

If we assume an AR(1) model for the data, we can consider a method similar to bootstrapping residuals for linear regression.

Model-based – the performance of this approach depends on the model being appropriate for the data.

3.2 Nonparametric approach

To deal with dependence in the data, we will employ a nonparametric *block* bootstrap.

Idea:

3.2.1 Nonoverlapping Blocks (NBB)

Consider splitting $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ in b consecutive blocks of length ℓ.

We can then rewrite the data as $\boldsymbol{Y} = (\boldsymbol{B}_1, \ldots, \boldsymbol{B}_b)$ with $\boldsymbol{B}_k = (Y_{(k-1)\ell+1}, \ldots, Y_{k\ell})$, $k = 1, \ldots, b$.

Note, the order of data within the blocks must be maintained, but the order of the blocks that are resampled does not matter.

3.2.2 Moving Blocks (MBB)

Now consider splitting $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ into overlapping blocks of adjacent data points of length ℓ.

We can then write the blocks as $\boldsymbol{B}_k = (Y_k, \ldots, Y_{k+\ell-1})$, $k = 1, \ldots, n - \ell + 1$.

3.2.3 Choosing Block Size

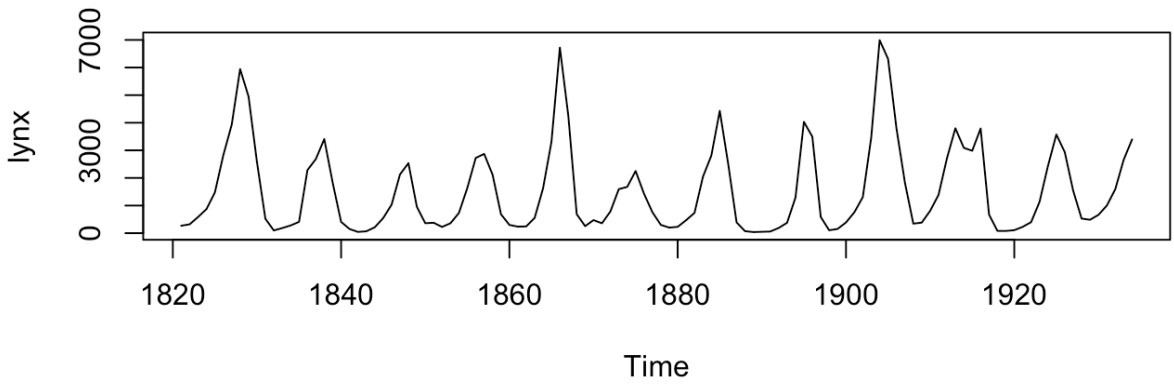
If the block length is too short,

If the block length is too long,

Your Turn

We will look at the annual numbers of lynx trappings for 1821–1934 in Canada. Taken from Brockwell & Davis (1991).

```
data(lynx)
plot(lynx)
```



Goal: Estimate the sample distribution of the mean

```
theta_hat <- mean(lynx)
theta_hat
```

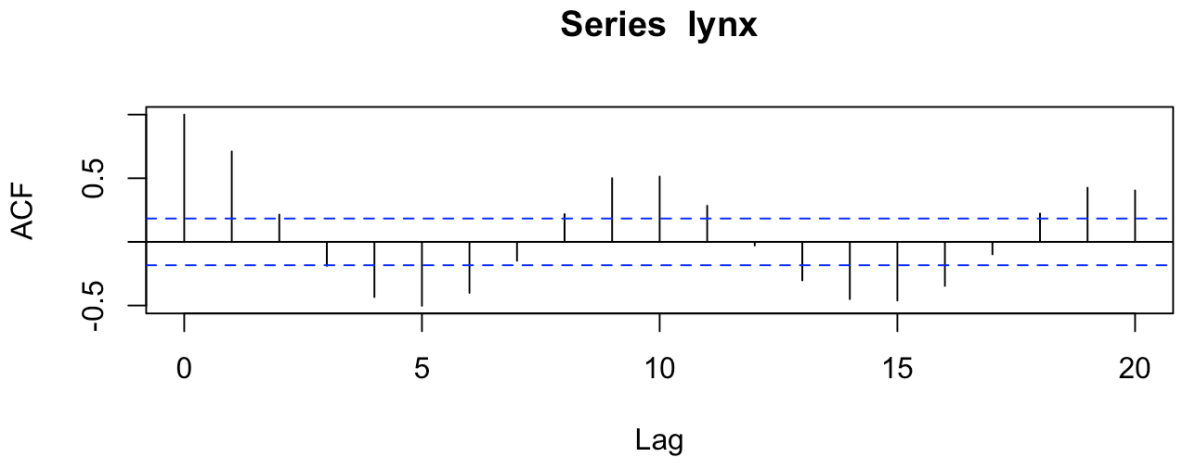
```
## [1] 1538.018
```

3.2.4 Independent Bootstrap

```
library(simpleboot)
B <- 10000

## Your turn: perform the independent bootstap
## what is the bootstrap estimate se?
```

We must account for the dependence to obtain a correct estimate of the variance!

```
acf(lynx)
```



The acf (autocorrelation) in the dominant terms is positive, so we are *underestimating* the standard error.

3.2.5 Non-overlapping Block Bootstrap

```r
# function to create non-overlapping blocks
nb <- function(x, b) {
  n <- length(x)
  l <- n %/% b

  blocks <- matrix(NA, nrow = b, ncol = l)
  for(i in 1:b) {
    blocks[i, ] <- x[((i - 1)*l + 1):(i*l)]
  }
  blocks
}

# Your turn: perform the NBB with b = 10 and l = 11
theta_hat_star_nbb <- rep(NA, B)
nb_blocks <- nb(lynx, 10)
for(i in 1:B) {
  # sample blocks
  # get theta_hat^*
}

# Plot your results to inspect the distribution
# What is the estimated standard error of theta hat? The Bias?
```

3.2.6 Moving Block Bootstrap

```r
# function to create overlapping blocks
mb <- function(x, l) {
  n <- length(x)
  blocks <- matrix(NA, nrow = n - l + 1, ncol = l)
  for(i in 1:(n - l + 1)) {
    blocks[i, ] <- x[i:(i + l - 1)]
  }
  blocks
}

# Your turn: perform the MBB with l = 11
mb_blocks <- mb(lynx, 11)
theta_hat_star_mbb <- rep(NA, B)
for(i in 1:B) {
  # sample blocks
  # get theta_hat^*
}

# Plot your results to inspect the distribution
# What is the estimated standard error of theta hat? The Bias?
```

3.2.7 Choosing the Block size

```
# Your turn: Perform the mbb for multiple block sizes l = 1:12
# Create a plot of the se vs the block size. What do you notice?
```

4 Summary

Bootstrap methods are simulation methods for frequentist inference.

Bootstrap methods are useful for

Bootstrap methods can fail when