2 Parametric Bootstrap

In a **nonparametric bootstrap**, we resample the observed data

Create a bootstrapped sample $y_1^*, \ldots, y_n^*$ iid from empirical dsn $\hat{F}_n$.

iid case, equivalent to resampling original data w/ replacement.

In a **parametric bootstrap**, Assume a parametric model.

Key idea: use a fitted parametric model $\hat{F}(y) = F(y|\hat{\psi})$ to estimate F where $\hat{\psi}$ estimate using MLE (or another method) from data.

Create a bootstrapped sample $y_1^*, \ldots, y_n^*$ iid from $F(y|\hat{\psi})$.

i.e. resample from a model w/ parameters estimated using original data.

For both methods,

(1) Compute $\hat{\theta}^{*(b)}$ for each bootstrapped sample $y_1^{*(b)}, \ldots, y_n^{*(b)}$

(2) repeat procedure B times to get

$\hat{\theta}^{*(1)}, \ldots, \hat{\theta}^{*(B)}$

and make inferences using the result.

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 $\epsilon_i \overset{iid}{\sim} N(0, \sigma^2)$.

$Y_1, \ldots, Y_n$ not iid! They have different conditional means.

Resampling in the bootstrap must be completed on iid quantities.

Two approaches for bootstrapping linear regression models –

1. Bootstrapping the residuals (model based, parametric).

2. Paired bootstrapping (case resampling, nonparametric)

2.1.1 Bootstrapping the residuals (model-based).

1. Fit the regression model using the original data to get $\hat{\beta}$

2. Compute the residuals from the regression model, errors ε_i are assumed iid.

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

3. Sample $\hat{\epsilon}_1^*, \ldots, \hat{\epsilon}_n^*$ with replacement from $\hat{\epsilon}_1, \ldots, \hat{\epsilon}_n$.

4. Create the bootstrap sample

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

($\boldsymbol{x}_i^T \hat{\boldsymbol{\beta}}$: fitted values based on model/data) → get

Bootstrapped data: $\{y_1^*, x_1\}, \ldots, \{y_n^*, x_n\}$

5. Estimate $\hat{\boldsymbol{\beta}}^*$ ← fit regression model on bootstrapped data to get $\hat{\beta}^*$

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

Assumptions:

- ε_i are iid
  - ↳ i.e. we have fit a "good" model.
- design matrix $\underline{X}$ is fixed.

2.1.2 Paired bootstrapping (case resampling).

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

Fit regression model w/ n bootstrapped pairs $(y_i, \underline{x}_i)^*$.

$$y_i^* = (\underline{x}_i^*)^T \underline{\beta} + \varepsilon_i \quad i = 1, \ldots, n$$

Assumptions:

Assume $(y_i, \underline{x}_i)$ are iid from a population.

Can have varying design matrix $\underline{X}$.

2.1.3 Which to use?

1. Standard inferences - i.e. earlier part of this class, likelihood approaches.

   Most of the time.

2. Bootstrapping the residuals -

   - most appropriate for designed experiments where $\underline{X}$ is fixed in advance.
   - model based, model must be reasonable fit for the data.
   - useful if complex sampling dsn for $\hat{\beta}$ — maybe some weird nonlinear function maybe.

3. Paired bootstrapping -

   - robust to model misspecification.
   - useful for observational studies where values of predictors aren't fixed in advance
     $\Rightarrow$ bootstrap mirrors data generating process.

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
```

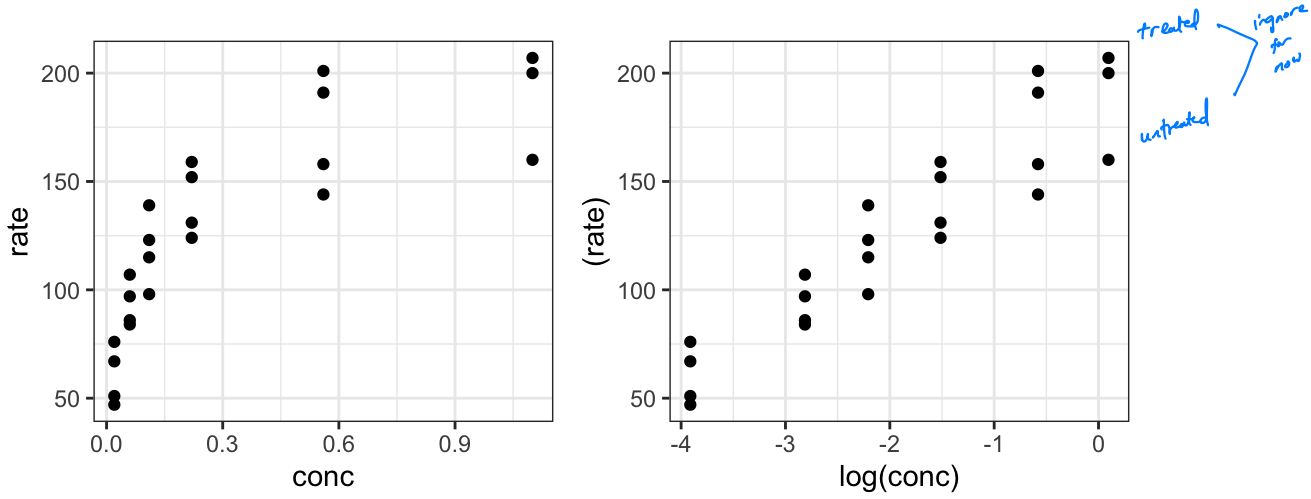
```
dim(Puromycin)
```

n=23

```
## [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)
```

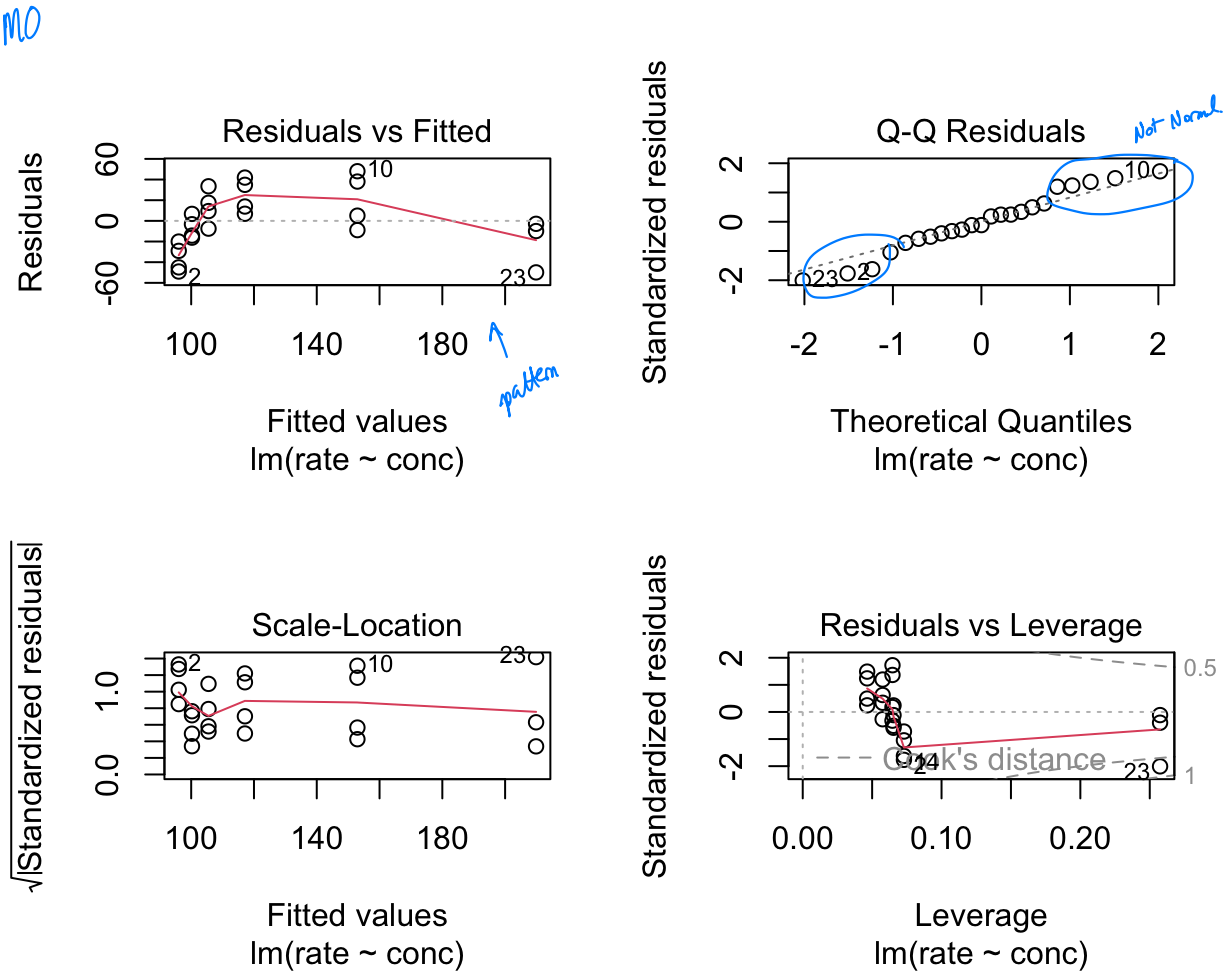
```
##
## Call:
## lm(formula = rate ~ log(conc), data = Puromycin)
```

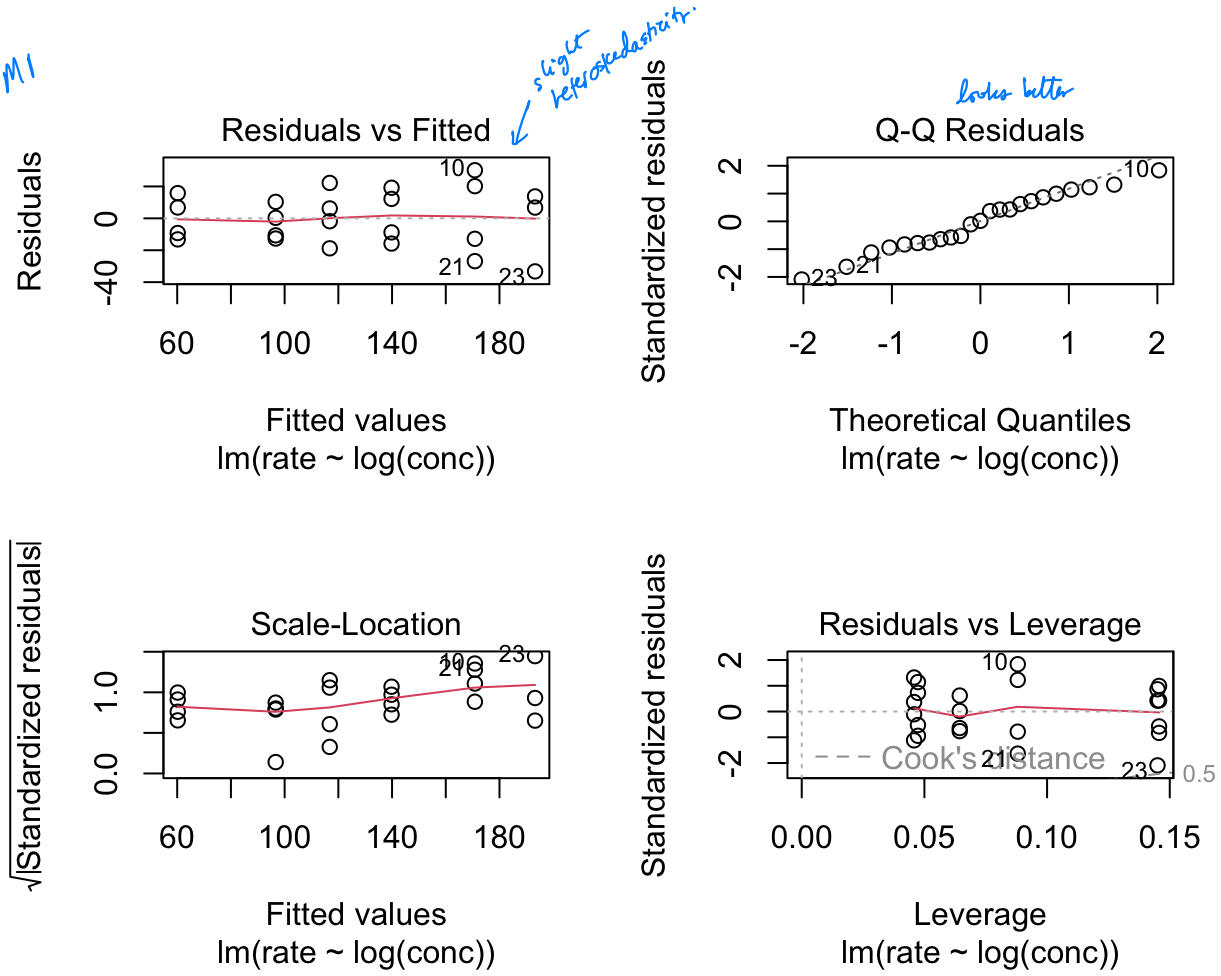
```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.250 -12.753   0.327  12.969  30.166
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  190.085      6.332   30.02  < 2e-16 ***
## log(conc)     33.203      2.739   12.12 6.04e-11 ***
## ---
##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

← based on asymptotic normality of MLE + Fisher Information.


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)


M1
slight heteroskedasticity
looks better
Residuals vs Fitted
Residuals
Fitted values
lm(rate ~ log(conc))
Q-Q Residuals
Standardized residuals
Theoretical Quantiles
lm(rate ~ log(conc))
Scale-Location
√|Standardized residuals|
Fitted values
lm(rate ~ log(conc))
Residuals vs Leverage
Standardized residuals
Cook's distance
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
}
                    or write your own is fine.
# 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 $F = F_{\boldsymbol{Y}} = F_{(Y_1,\ldots,Y_n)}$.

No longer assuming $Y_1, \ldots, Y_n$ independent.

Ex could be time series, spatial, network, etc.

Goal:

To approximate dsn of a statistic $\theta = T(Y)$.

Challenge:

Since Y_i's are dependent it is inappropriate to use the iid bootstrap.

Bootstrapped samples would no longer reproduce the data generating process.

(and sampling independently from $\hat{F}_n$ no longer mimics drawing original sample from F).

We will consider 2 approaches

(1) Model-based (parametric).

(2) Block bootstrap (nonparametric).

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.,

$$Y_t = \alpha Y_{t-1} + \varepsilon_t \quad t = 1, \ldots, n$$

$|\alpha| < 1$ and $\varepsilon_1, \ldots, \varepsilon_n \overset{iid}{\sim} (0, \sigma^2)$.

Why not just move forward with our nonparametric bootstrap procedure?

Failure of nonparametric iid bootstrap for TS data.

Let's suppose $\{X_t\}_{t \in \mathbb{Z}}$ is a stationary, m-dependent process with $EX_t = \mu$, $EX_t^2 < \infty$.

("almost" independent)

joint probability dsns don't change with time.

$(X_{t_1}, \ldots, X_{t_k}) \overset{d}{=} (X_{t_1+h}, \ldots, X_{t_k+h})$ for any $t_1, \ldots, t_k, h$

let $r(k) = \mathrm{Cov}(X_1, X_{1+k})$.

"m-dependent": $r(k) = 0$ for $k > m$.

This a stronger assumption than AR(1) on the dependence.

Say we want to approximate dsn of $T_n = \sqrt{n}(\bar{X}_n - \mu)$. Say we apply iid bootstrap:

Draw observations from $\{X_1, \ldots, X_n\}$ to get $T_n^* = \sqrt{n}(\bar{X}_n^* - E_* X_1^*) = \sqrt{n}(\bar{X}_n^* - \bar{X}_n)$ and approximate $P(T_n \le x)$ with $P_*(T_n^* \le x)$, $x \in \mathbb{R}$.

Thm: If $X_1, X_2, \ldots$ stationary, m-dependent process w/ $\mathrm{Var}(X_1) = \sigma^2 < \infty$, then

$$\sup_{x \in \mathbb{R}} \left| P_*(T_n^* \le x) - \Phi\left(\frac{x}{\sigma}\right) \right| \equiv \Delta_n \longrightarrow 0 \quad \text{as } n \to \infty \text{ a.s.}$$

Proof is very similar to iid version (pg. 6-7). Just relies on M-Z SLLN to hold for m-dependent process, which it does.

Problem? If $\{X_t\}$ is stationary, m-dependent, then

$$\lim_{n \to \infty} \mathrm{Var}(\sqrt{n}\,\bar{X}_n) = \lim_{n \to \infty} \mathrm{Var}(T_n) \overset{\text{comes from dominated convergence theorem}}{=} \sum_{k=-\infty}^{\infty} r(k) \overset{\text{m-dependence}}{=} \sum_{-m}^{m} r(k) \equiv \sigma_\infty^2.$$

If $\sigma_\infty^2 > 0$, then $T_n \equiv \sqrt{n}(\bar{X}_n - \mu) \xrightarrow{d} N(0, \sigma_\infty^2)$ by a CLT.

So iid bootstrap will fail unless $\sigma_\infty^2 = \sigma^2 = r(0)$

In practice, $\sigma_\infty^2 > r(0)$ holds most often $\Rightarrow$ we are underestimating uncertainty w/ iid bootstrap!

This was for m-dependent process, which is a very strong assumption! Under more realistic process, may be even worse.

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

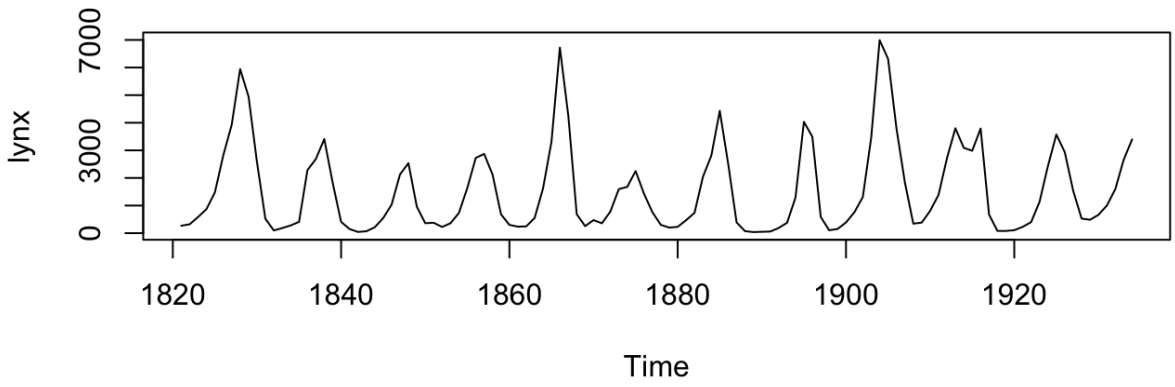
If the block length is too short,

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
```

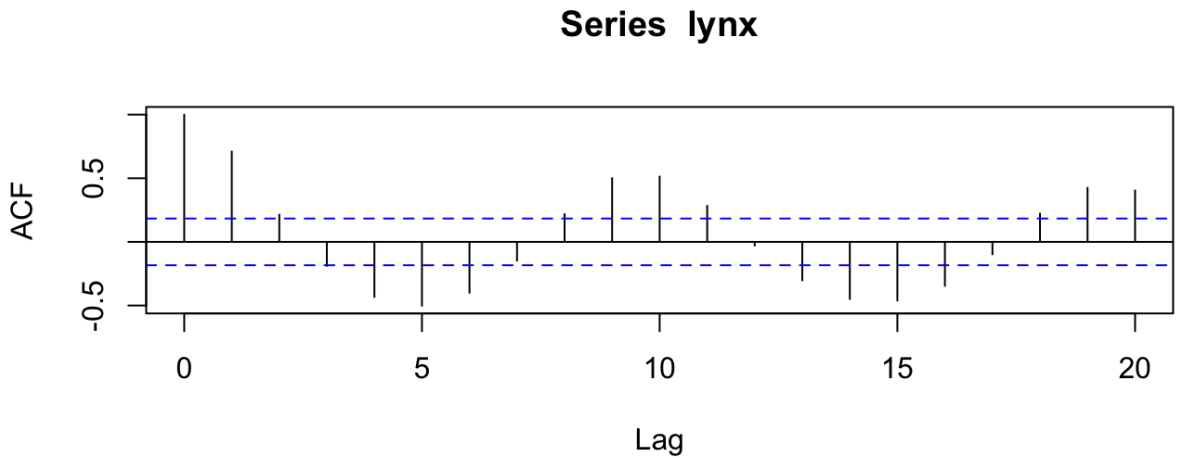
3.2.4 Independent Bootstrap

```
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