2 Parametric Bootstrap

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

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

$\uparrow$

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

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

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

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

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

For both methods,

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

② repeat procedure B times to get

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

and make inferences using the result.

2.1 Bootstrapping for linear regression

Consider the regression model $\underline{Y_i} = \boldsymbol{x}_i^T\boldsymbol{\beta} + \epsilon_i, i = 1, \ldots, n$ 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
fitted values based on model/data
$$y_i^* = \boxed{\boldsymbol{x}_i^T\hat{\boldsymbol{\beta}}} + \epsilon_i^*, \quad i = 1, \ldots, n \quad \rightarrow \text{get}$$
bootstrapped data.
$$\boxed{\begin{matrix}\{y_1^*, \underline{x}_1\}\\ \vdots \\ \{y_n^*, \underline{x}_n\}\end{matrix}}$$
5. Estimate $\hat{\boldsymbol{\beta}}^*$ ← fit regression model on bootstrapped data to get $\hat{\beta}^*$
6. Repeat steps ~~2-4~~ 3-5 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 $\hat{\beta} \leftarrow$ 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
```

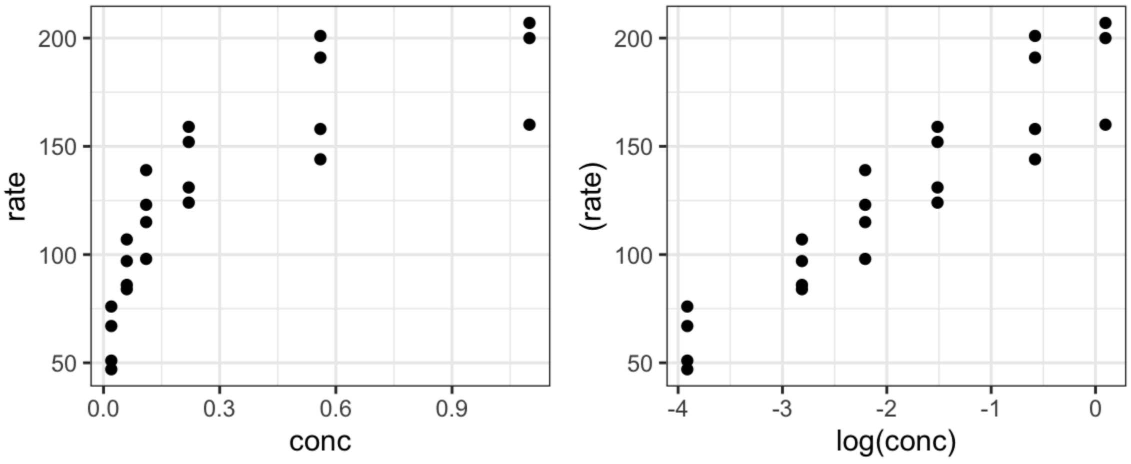
```
dim(Puromycin)
```

n=23

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

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

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



treated
untreated
ignore for now

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)
```

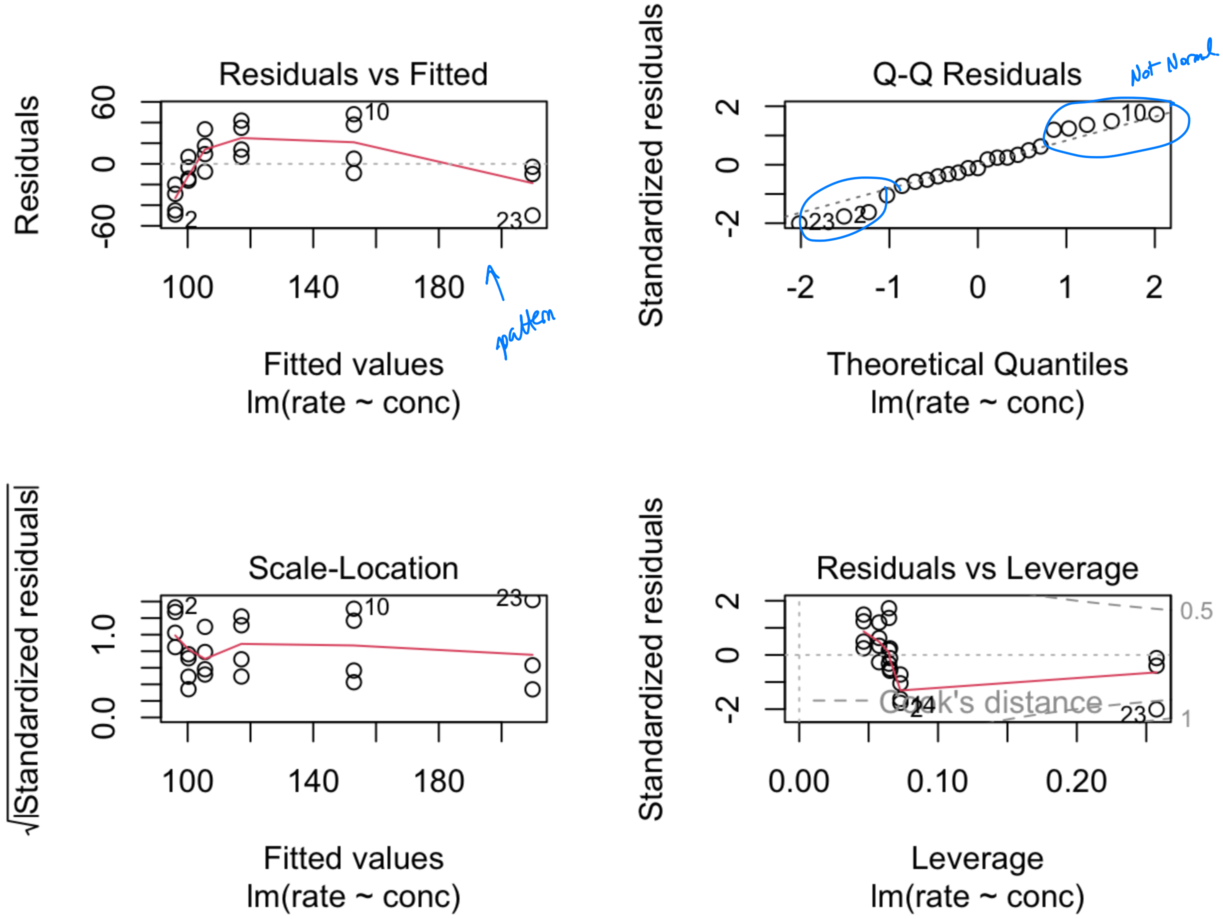
```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.250 -12.753   0.327  12.969  30.166
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  190.085      6.332   30.02  < 2e-16 ***
## log(conc)     33.203      2.739   12.12 6.04e-11 ***
## ---
##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

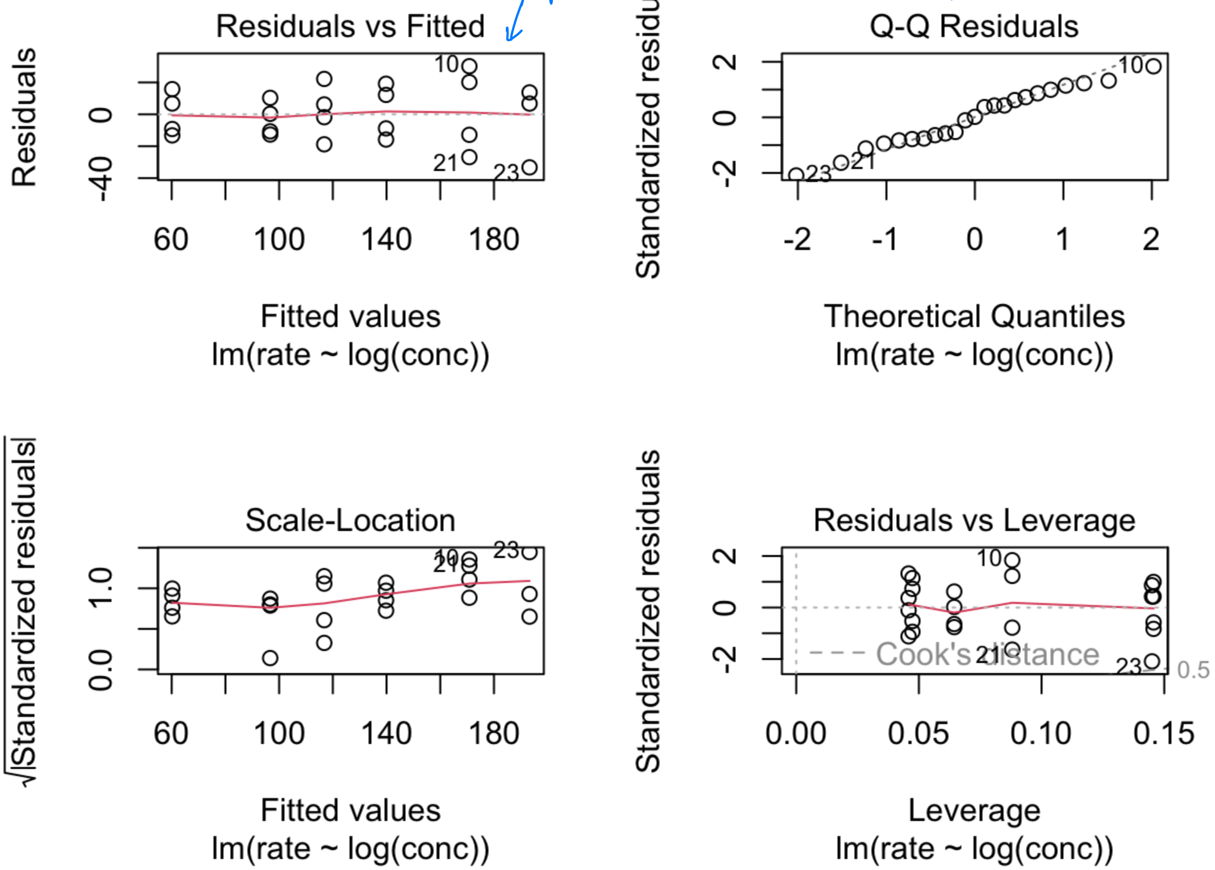
MO



Not Normal

pattern

M1



slight heteroskedasticity.

looks better

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.

↳ 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 $\boldsymbol{Y} = (Y_1, \dots, Y_n)$ which we assume is generated by an AR(1) process, i.e.,

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

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

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.

$$(X_{t_1}, \dots, X_{t_k}) \overset{d}{=} (X_{t_1+h}, \dots, X_{t_k+h})$$

for any $t_1, \dots, t_k, h$

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

"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 $\{X_1, \dots, X_n\}$ 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 $P_*(T_n^* \le x)$, $x \in \mathbb{R}$.

Th'm: If $X_1, X_2, \dots$ stationary, m-dependent process w/ $Var(X_1) = \sigma^2 < \infty$, then

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

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

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

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.

↳ turn our problem into iid bootstrap.

Recall AR(1): $Y_t = \alpha Y_{t-1} + \varepsilon_t \quad t = 1, \dots, n \quad |\alpha| < 1$ and $\varepsilon_1, \dots, \varepsilon_n \overset{iid}{\sim} (0, \sigma^2)$. !!

① Estimate $\hat{\alpha}$ from data (fit the model).

② Define estimated "innovations" $\hat{e}_t = Y_t - \hat{\alpha} Y_{t-1}, \quad t = 2, \dots, n$

and $\bar{\hat{e}} = \frac{1}{n-1} \sum_{t=2}^{n} \hat{e}_t$

③ Define the residuals as centered innovations.

$$\hat{\varepsilon}_t = \hat{e}_t - \bar{\hat{e}} \qquad [E\varepsilon_i = 0]$$

④ For $r = 1, \dots, R$

a) Create a bootstrap sample $\hat{\varepsilon}^*_0, \dots, \hat{\varepsilon}^*_n$ by randomly sampling $n+1$ values from the $n-1$ values $\hat{\varepsilon}_t$, $t = 2, \dots, n$.

b) Construct pseudo data $Y^* = (Y^*_1, \dots, Y^*_n)$ from

$$Y^*_0 = \hat{\varepsilon}^*_0, \quad Y^*_t = \hat{\alpha} Y^*_{t-1} + \hat{\varepsilon}^*_t, \quad t = 1, \dots, n.$$

c) define $\hat{\alpha}^*_r$ as the estimate of α from $Y^*_1, \dots, Y^*_n$.

⑤ dsn of $\hat{\alpha}^*_1, \dots, \hat{\alpha}^*_r$ is bootstrap estimate of dsn of $\hat{\alpha}$.

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

As we know, this may not always be a good assumption.

3.2 Nonparametric approach

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

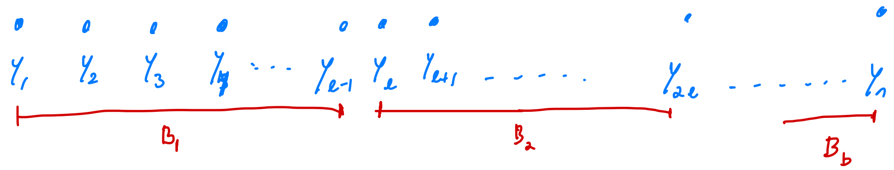
Idea:

resample data in blocks to preserve the dependence structure within the blocks.

3.2.1 Nonoverlapping Blocks (NBB)

Carlstein (1986).

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 $\boldsymbol{B}_k = (Y_{(k-1)\ell+1}, \ldots, Y_{k\ell})$, $k = 1, \ldots, b. = \lfloor \frac{n}{\ell} \rfloor$

① Sample nonoverlapping blocks $B_1^*, \ldots, B_b^*$ independently from $B_1, \ldots, B_b$ with replacement to form pseudo data set $Y^* = (B_1^*, \ldots, B_b^*)$.

② estimate statistic of interest from Y^* to get $\hat{\theta}^*$.

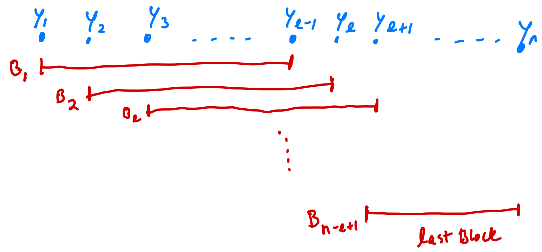
③ Repeat ①-② R times to obtain $\hat{\theta}^{*(1)}, \ldots, \hat{\theta}^{*(R)}$ to estimate dsn of $\hat{\theta}$.

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)

Künsch (1989)
Liu & Singh (1992).

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



Now we have more blocks to choose from! $\left(N = n-\ell+1 \text{ vs. } b = \lfloor \frac{n}{\ell} \rfloor\right)$.

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

get/collect blocks $\mathcal{C} \equiv \{B_1, \dots, B_N\}$ sampling $B_1^*, \dots, B_b^*$ from $\mathcal{C}$, $b = \lfloor \frac{n}{\ell} \rfloor$, put together to get $Y^* = (B_1^*, \dots, B_b^*)$.

Alternative but equivalent formulation let $I_1, \dots, I_b$ be iid w/ $P(I_1 = j) = \frac{1}{N}$, $j = 1, \dots, N$ set $B_i^* = B_{I_i}^*$ $i = 1, \dots, b$.

Ex: let $\hat{\theta}_n = \bar{Y}_n$. Get MBB sample mean version $\bar{Y}_m^* = \sum_{i=1}^{m} Y_i^*/\underset{\sim}{m}$, Find $E_*(\bar{Y}_m^*)$ and $Var_*(\sqrt{m}\,\bar{Y}_m^*)$ which estimate $E(\bar{Y}_n)$ and $Var(\sqrt{n}\,\bar{Y}_n)$.
$m = b \cdot \ell$

Note: $\bar{Y}_m^* = \frac{1}{b}\sum_{i=1}^{b} \bar{Y}_{B_i^*}^*$ ← sample mean of ith block B_i^*, $b = \lfloor \frac{n}{\ell} \rfloor$

① $$E_*(\bar{Y}_m^*) = \frac{1}{b}\sum_{i=1}^{b} E_*(\bar{Y}_{B_i^*}^*) \overset{\text{sample blocks iid}}{=} E_*(\bar{Y}_{B_i^*}^*) = \frac{1}{\underbrace{n-\ell+1}_{\text{uniform blocks}}} \overset{\# \text{ blocks}}{\sum_{i=1}^{n-\ell+1}} \left(\sum_{t=i}^{i+\ell-1} Y_t / \ell \right)$$

$$= \frac{1}{N}\sum_{i=1}^{N} \bar{Y}_i$$ where $\bar{Y}_i$ = sample mean of block B_i and $N = n-\ell+1$.

$$\neq \bar{Y}_n$$

② $$Var_*(\sqrt{m}\,\bar{Y}_m^*) = Var_*\left(\sqrt{m}\,\frac{1}{b}\sum_{i=1}^{b} \bar{Y}_{B_i^*}^*\right) = \frac{m}{b^2}\sum_{i=1}^{b} Var_*(\bar{Y}_{B_i^*}^*) = \frac{\overset{b\cdot\ell}{m}}{b^2}\cdot b\, Var_*(\bar{Y}_{B_i^*}^*)$$
Bootstrap blocks sampled iid

$$= \ell\, E_*\left(\bar{Y}_{B_i^*}^* - E_*\bar{Y}_{B_i^*}^*\right)^2 = \ell\, \frac{1}{N}\sum_{i=1}^{N} (\bar{Y}_i - \hat{\mu})^2$$ where $\hat{\mu} \equiv \frac{1}{N}\sum_{i=1}^{N} \bar{Y}_i$ as above.

↑ this looks like a sample variance of $\sqrt{\ell}\,\bar{Y}_1, \dots, \sqrt{\ell}\,\bar{Y}_N$ of sample means from blocks.

This directly estimates the variance of sample mean of length ℓ block $\sqrt{\ell}\,\bar{Y}_i$.

$\Rightarrow Var_*(\sqrt{m}\,\bar{Y}_m^*)$ estimate $Var(\sqrt{\ell}\,\bar{Y}_1) = \ell\, Var\bar{Y}_1 \approx n\, Var\bar{Y}_n$ (target for MBB).

Both NBB and MBB fix the variance issue from page 35.

NOTE: The MBB version of $\sqrt{n}(\bar{Y}_n - \mu) = \sqrt{n}(\bar{Y}_n - E\bar{Y}_n)$ is NOT $\sqrt{m}(\bar{Y}_m^* - \bar{Y}_n)$!

is actually $\sqrt{m}(\bar{Y}_m^* - E_*\bar{Y}_m^*) = \sqrt{m}(\bar{Y}_m^* - \hat{\mu})$
$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} \bar{Y}_i$.

Note in above (and previously) $\hat{\theta}_n = T(F_n)$ is a function of the empirical dsn of Y. Sometimes we need to look at other statistics, which are functions of the empirical dsn of $X_i = \underbrace{(Y_i, \dots, Y_{i+p-1})}_{\text{p-tuple.}}$, $i = 1, \dots, n-p+1$. Solution: resample blocks of X_i's, not Y_i's

↑ p-tuple's of Y_i's.

3.2.3 Choosing Block Size

If the block length is too short,

The resample cannot to capture the dependence ($\ell = 1$ is iid bootstrap!).

If the block length is too long,

not many blocks to resample ($\ell = n$, only have 1 block). does not mimic data generation. & leads to high variance.

Asymptotic result: block length should increase w/ length of the time series. If MBB & NBB produce consistent estimators of moments, etc.

For variance estimation, it is known the optimal block length:

$$b_n^{opt} = \begin{cases} \left[3B_0^2 / 2\sigma_\infty^4\right]^{1/3} n^{1/3} + o(n^{1/3}) & \text{MBB} \\ \left[B_0^2 / \sigma_\infty^4\right]^{1/3} n^{1/3} + o(n^{1/3}) & \text{NBB.} \end{cases}$$

$$\text{where} \quad B_0 = \sum_{K=-\infty}^{\infty} K \underset{\underset{\text{Cov}(Y_1, Y_{1+K}).}{\uparrow}}{r(K)} \quad \text{and} \quad \sigma_\infty^2 = \lim_{n \to \infty} Var(T_n) = \sum_{K=-\infty}^{\infty} r(K).$$

Outside of variance estimation, not much known about "optimal" block sizes.

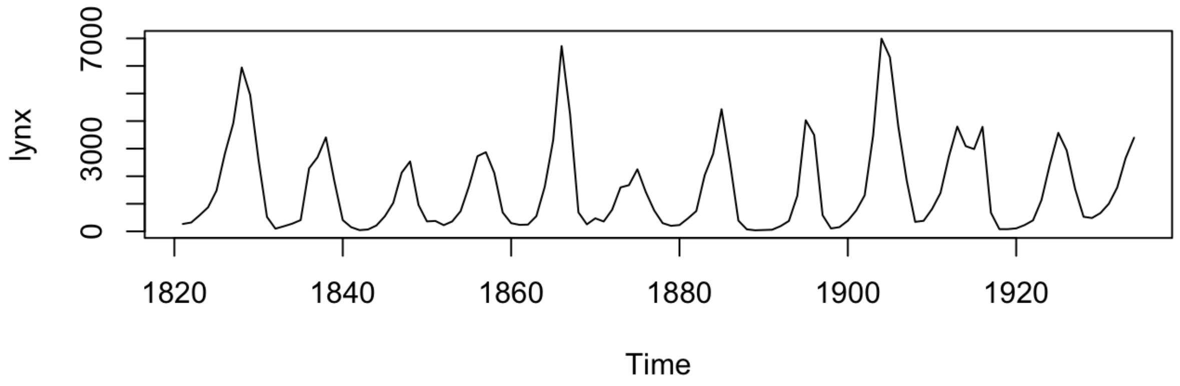
NPPI → Lahiri, et.al (2007) suggests a plug-in method based on a nonparametric approach for general block selection.

↑ See for details

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
```

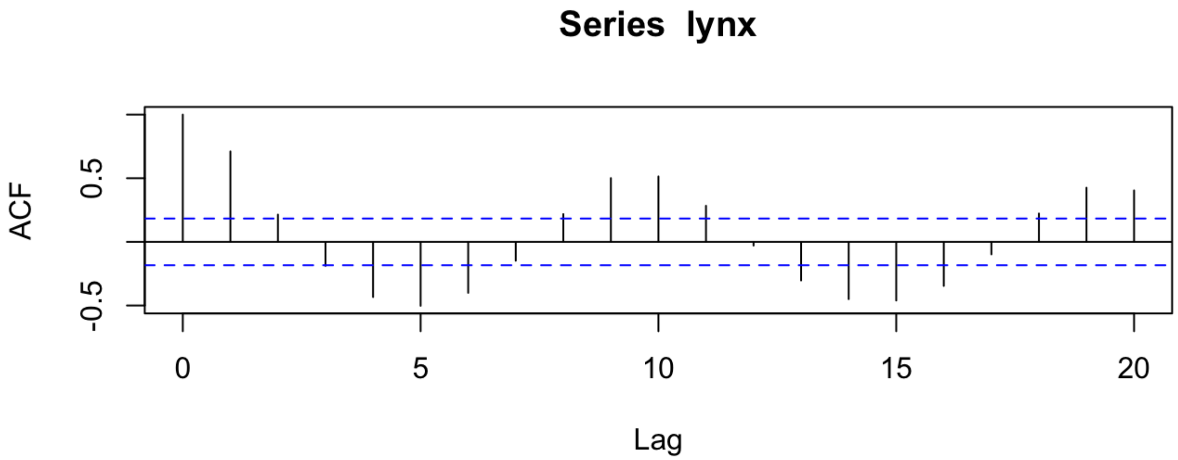
3.2.4 Independent Bootstrap

```r
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!

```r
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

```
# 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