Note in above, assumed

$\hat{\theta} = T(F_n)$ 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 $X_i = (Y_i, \ldots, Y_{i+p-1})$.

p-tuple.

3.2.3 Choosing Block Size

Solution: resample blocks of X_i's not Y_i's

$\Leftarrow$ p-tuples of Y_i

If the block length is too short,

The resampling cannot capture the dependence ($b = 1$, is iid bootstrap).

If the block length is too long,

not many blocks to resample (does not mimic data generation & leads to high variance in estimators).

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

?

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

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

where $B_0 = \sum_{k=-\infty}^{\infty} k\, r(k)$ $\qquad$ $G_\infty^2 = \lim_{n\to\infty} \mathrm{Var}(T_n) = \sum_{k=-\infty}^{\infty} r(k)$.

$r(k) = \mathrm{Cov}(Y_1, Y_{1+k})$

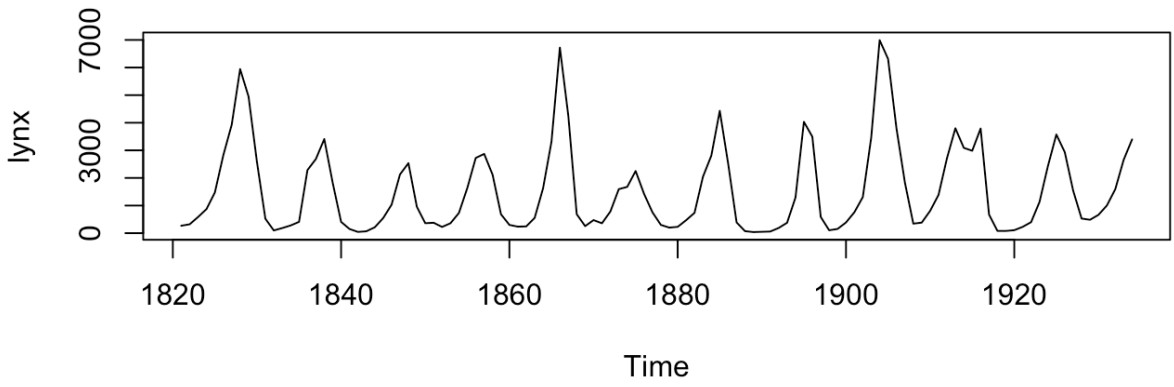
Outside of variance estimation, not much is known about "optimal" block sizes.

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

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
```

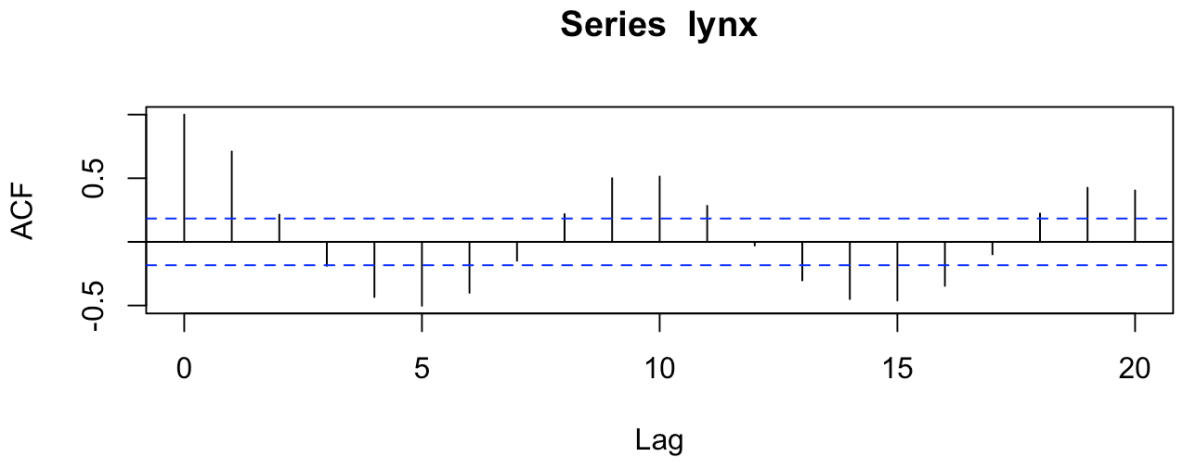
3.2.4 Independent Bootstrap

```
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

many problem types, especially when standard assumptions are doubted.

Remember: bootstrap principle says the bootstrap dsn should approximate the sampling dsn of the statistic as long as bootstrap sampling scheme mimics original data generating process.

Bootstrap methods can fail when

We have extremes or heavy-tailed dsns (rare events hard to bootstrap).

Need to be careful w/ non-iid data.

can be computationally expensive.