Week 3

3.1 The bootstrap method

For estimating an expected value of the form

$$\psi_h(F) = \int_{-\infty}^{\infty} h(y) dF(y),$$

the plug-in method is easily applied, leading to an estimator that is a simple average. However, when the function of interest is more complicated, evaluating $\psi_h(\widehat{F})$ is often difficult and, hence, an alternative is to use Monte Carlo simulation, as described in Section 2.2. This method is convenient because drawing a random sample from the distribution with distribution function $\widehat{F}(\cdot)$ is particularly simple.

Let $Y_1, Y_2, \ldots, Y_n$ denote i.i.d. random variables, each distributed according to the distribution with distribution function $F(\cdot)$ and let $\widehat{F}(\cdot)$ denote the corresponding empirical distribution. If y_j denotes the observed value of Y_j, $j = 1, 2, \ldots, n$, then $\widehat{F}(\cdot)$ may be viewed as the distribution function of the discrete distribution taking values $y_1, y_2, \ldots, y_n$. Hence, drawing a random sample of size n from the distribution with distribution function $\widehat{F}(\cdot)$ is equivalent to drawing a random sample with replacement from the set $\{y_1, y_2, \ldots, y_n\}$.

Let $Y_1^*, Y_2^*, \ldots, Y_n^*$ denote such a random sample. Then, for each $j = 1, 2, \ldots, n$, Y_j^* is obtained by randomly choosing a value from $\{y_1, y_2, \ldots, y_n\}$ such that each element in this set has the same chance of being selected. If $y_1, y_2, \ldots, y_n$ are unique, each value has probability $1/n$ of being selected; if, e.g., $y_1 = y_2 = c$ for some constant c and no other y_j are equal to c, then the probability of c being selected is $2/n$.

This procedure, known as the *bootstrap method*, is illustrated on the following example.

Example 3.1 Consider the returns on Apple stock, stored in the variable `apple`; see Example **??**. To illustrate the bootstrap method we will first use only the first five return values in `apple`, which are stored in the variable `apple5`:

```
> apple5<-apple[1:5]
> apple5
[1]  0.127  0.188  0.105 -0.026 -0.011
```

The sample mean of these five values is 0.0766 and the standard error of the sample mean 0.0412:

```
> mean(apple5)
[1] 0.0766
> sd(apple5)/(5^.5)
[1] 0.0412
```

Now consider the simulation-based approach to calculating this standard error. The basic approach is to draw five random variates from some specified distribution; let $F_0(\cdot)$ denote the distribution function of this distribution and let $Y_1^*, Y_2^*, Y_3^*, Y_4^*, Y_5^*$ denote five simulated values.

We then calculate the corresponding sample mean

$$\bar{Y}^* = \frac{1}{5}\sum_{j=1}^{5} Y_j^*.$$

This gives a sample of size 1 from the distribution of $\bar{Y}$.

Repeating this procedure $m-1$ times, yields a sample of size m from the distribution of $\bar{Y}$: $\bar{Y}_1^*, \bar{Y}_2^*, \ldots, \bar{Y}_m^*$. This sample can be used to determine properties of the distribution of $\bar{Y}$, such as its standard error, which is just the standard deviation of the distribution of $\bar{Y}$. That is, we calculate the standard error of $\bar{Y}$ by calculating the sample standard deviation of $\bar{Y}_1^*, \bar{Y}_2^*, \ldots, \bar{Y}_m^*$. If m is large this will approximate the true standard error of $\bar{Y}$.

The result will depend on distribution used for the simulation; that is, it will depend on the distribution function $F_0(\cdot)$. In the bootstrap method, we take $F_0(\cdot)$ to be the empirical distribution function, $\widehat{F}(\cdot)$.

We have seen that $\widehat{F}(\cdot)$ corresponds to the distribution function of a random variable randomly selected from the set $\{y_1, y_2, y_3, y_4, y_5\}$ which, in this example, are the five observed returns on Apple stock. Hence, we can follow the procedure described in Section 2.2 for sampling from a finite population.

The function `sample` may be used to draw a random sample from a set of integers. Specifically, `sample(5, replace=T)` draws a random sample with replacement from the set $\{1, 2, 3, 4, 5\}$:

```
> samp<-sample(5, replace=T)
> samp
[1] 2 1 1 5 3
```

Using the sampled integers as the indices of the vector `apple5` yields a random sample with replacement from the set of returns values in `apple5`:

```
> apple5[samp]
[1]  0.188  0.127  0.127 -0.011  0.105
```

The sample mean of `apple5[samp]` yields a simulated value of the sample mean return $\bar{Y}$ for Apple stock:

```
> mean(apple5[samp])
[1] 0.107
```

This procedure may be repeated multiple times; for example,

```
>  mean(apple5[sample(5, replace=T)])
[1] 0.0326
>  mean(apple5[sample(5, replace=T)])
[1] 0.0474
> mean(apple5[sample(5, replace=T)])
[1] 0.121
```

and so on.

Note that each time `mean(apple5[sample(5, replace=T)])` is calculated a new set of random numbers is drawn. Suppose we perform this procedure 100000 times, storing the sample means in the variable `apple5.boot`; these values represent a type of random sample drawn from the distribution of the sample mean of 5 returns on Apple stock. Here are the first eight values.

```
> apple5.boot[1:8]
[1] 0.0986 0.0767 0.1152 0.1349 0.0474 0.0737 0.0615 0.0781
```

The sample standard deviation of `apple5.boot` yields an estimate of the standard error of the sample mean $\bar{Y}$.

```
> sd(apple5.boot)
[1] 0.0378
```

Note that the bootstrap standard error is close to, but not exactly the same as, the value given by the usual formula for the standard error of the sample mean, 0.0412. There are two reasons for the difference. One is that the sample standard deviation uses a divisor of $n-1$; it may be shown that estimate of the standard deviation implicity used by the bootstrap

method is equivalent to the one with a divisor of n. The effect of these different divisors is highlighted in the example since in that case $n = 5$. For instance, when analyzing the full set of Apple return data, $n = 36$ and $\sqrt{36/35} = 1.014$ so that the difference is less important.

The other reason for the difference between the bootstrap standard error and the usual value is that the bootstrap method is based on a random sample. If the method is repeated, a different standard error will be obtained. For instance, the bootstrap method was repeated three times, with results

```
> sd(apple5.boot1)
[1] 0.0372
> sd(apple5.boot2)
[1] 0.0380
> sd(apple5.boot3)
[1] 0.0371
```

If a very large bootstrap sample size is used we expect that the result will be closer to that obtained by the usual method (taking into account the difference in the divisors). E.g., `apple5.boot100k` contains a random sample of size 100000 from the distribution of $\bar{Y}$.

```
> sd(apple5.boot100k)
[1] 0.0369
```

Note that $\sqrt{5/4} \doteq 1.118$; hence, after accounting for the difference in divisors, the result is nearly identical to the 0.0412 obtained from the usual formula.

The bootstrap method may also be used to estimate the bias of an estimator. Recall that the bias is the expected value of the sampling distribution of the estimator minus the true value of the parameter being estimated. Of course, we don't know the true value of the parameter; however, we have an estimate of it, the original estimate we calculated, in this case, the sample mean of `apple5`, 0.0767.

The mean of the sampling distribution may be estimated by the sample mean of the bootstrap values of $\bar{Y}$, here stored in the variable `apple5.boot100k`:

```
> mean(apple5.boot100k)
[1] 0.0767
```

Hence, the estimated bias is the mean of the sampling distribution (0.0767) minus the true value of the parameter. In the context of the simulation, the true parameter value is the sample mean of the data in `apple5`, 0.0767. It follows that, based on the results of the bootstrap method, the estimated bias of the sample mean 0 to four decimal places.

If more digits are used,

```
> mean(apple5.boot100k)
[1] 0.076673
```

and

```
> mean(apple5)
[1] 0.076745
```

so that the estimated bias is

$$0.076745 - 0.076673 = 0.000072$$

Of course, we know that the sample mean is an unbiased estimator of the mean of the distribution so it is not surprising that the estimated bias is very small. It is not exactly 0 because it is based on the random numbers used in constructing the bootstrap sample and, hence, the bias estimate itself has some sampling error. □

Obviously, the bootstrap method is not needed to calculate the standard error of a sample mean. However, it is extremely useful for calculating the standard error of more complicated statistics for which a simple formula is not available. Although it is possible to carry out the calculations described above for the sample mean for any statistic, fortunately, there are convenient R functions available for that purpose.

Here we will use the function `boot` in the package "boot".

Example 3.2 Let $Y_1, Y_2, \ldots, Y_n$ denote a sequence of i.i.d. random variables each with mean μ and standard deviation σ. The ratio μ/σ gives the mean of the distribution as a proportion of its standard deviation. When the Y_j are returns on a stock, as in the previous example, μ/σ is a measure of the expected return on a stock relative to its "risk", as measured by the return standard deviation; in that context, μ/σ is known as the *Sharpe ratio* of the stock.

Consider estimation of the Sharpe ratio for Apple stock; using data in the variable `apple`, the estimated Sharpe ratio is given by

```
> mean(apple)/sd(apple)
[1] 0.288
```

We may use the bootstrap method to compute a standard error of this estimate, along with an estimate of the bias of the estimator.

The function `boot` takes three arguments (there are other, optional, arguments for which we will use the default values). The most important of these is a function calculating the statistic of interest for a given set of data. Although this could be a standard R function, such as `mean` or `median`, in many cases it is a user-defined function.

For such a function to be used in `boot`, it must take two arguments: the data, in the form of a vector or matrix, and the indices of the data values to be used in the computation, similar to the way in which we have used the indices in Monte Carlo simulation when sampling from a finite population in Section 2.2 and in the sample mean example above.

Consider the function

```
> Sharpe<-function(x, ind){mean(x[ind])/sd(x[ind])}
```

This function takes the values in `x` corresponding to the indices in the vector `ind` and uses those values to compute the Sharpe ratio.

For example, consider the return data in the vector `apple`. To use all the data, we set `ind` to `1:36`, the vector of integers from 1 to 36.

```
> Sharpe(apple, 1:36)
[1] 0.288
```

which agrees with the result obtained previously by computing the Sharpe ratio directly.

If `1:36` is replaced by `1:5`, the result is the Sharpe ratio based on the first five values; recall that these are stored in the variable `apple5`.

```
> Sharpe(apple, 1:5)
[1] 0.832
> mean(apple5)/sd(apple5)
[1] 0.832
```

The other arguments to the `boot` are `data`, the data used in calculating the statistic of interest and `R`, the number of bootstrap replications to be used.

To calculate the standard error of the estimated Sharpe ratio for the data in `apple` based on a bootstrap sample size of 100000, we use the command

```
> library(boot)
> boot(apple, Sharpe, 100000)
ORDINARY NONPARAMETRIC BOOTSTRAP

Bootstrap Statistics :
    original  bias    std. error
t1*    0.288  0.0118       0.181
```

The output gives the value of the estimate, under the heading "original"; hence, the estimated Sharpe ratio for these data is 0.288, as calculated previously. The standard error, given under the "std. error" heading, is 0.181.

The output of the `boot` function also includes an estimate of the bias of the estimator. A *bias-corrected* estimate may be formed by subtracting the bias from the estimate; e.g., a bias-corrected estimate of the Sharpe ratio for Apple stock is $0.288 - 0.012 = 0.376$. Whenever the bias is small relative to the standard error, the impact of the bias correction is small and, hence, it may be ignored. A simple rule-of-thumb is that the estimated bias may be ignored when it is less than one-fourth of the standard error; of course, such a guideline will not be appropriate in all cases. □

Note that the bootstrap method, as described above, is a nonparametric procedure, in the sense that it does not rely on the assumption of a specific parametric model for the data.

However, it can be used to calculate the standard error of a parameter estimate in a parametric model; when used in this way, the bootstrap provides a standard error that is valid even if the parametric model assumption is not satisfied.

Example 3.3 Consider the following data on the failure times (in operating hours) of aircraft air-conditioning equipment, stored in the R variable `ac`

```
> ac
[1] 359   9  12  270  603  3  104  2  438
```

Suppose we model these data as i.i.d. random variables $Y_1, Y_2, \ldots, Y_9$, each with an exponential distribution with rate parameter λ, which has density

$$\lambda \exp(-\lambda y), \quad y > 0.$$

It is straightforward to show that the maximum likelihood estimator of λ is given by

$$\widehat{\lambda} = \frac{1}{\bar{Y}}$$

where $\bar{Y}$ is the sample mean of $Y_1, Y_2, \ldots, Y_9$. For the present data, the estimate of λ is

```
> 1/mean(ac)
[1] 0.005
```

The bootstrap method can be used to determine a standard error for $1/\bar{Y}$ that is valid without assuming that the data are sampled from an exponential distribution.

To apply the bootstrap to the present example, we need to define a function that computes $\widehat{\lambda}$ for a given set of data and a given set of indices; thus, define a function `lambda` as follows:

```
> lambda<-function(x, ind){1/mean(x[ind])}
```

Note that

```
> lambda(ac, 1:9)
[1] 0.005
```

To compute the bootstrap standard error of $\widehat{\lambda}$ for the air-conditioning data using a bootstrap sample size of 100000 we use the command

```
> boot(ac, lambda, R=100000)
ORDINARY NONPARAMETRIC BOOTSTRAP


Bootstrap Statistics :
    original   bias    std. error
t1*    0.005 0.001093    0.005889
```

Thus, the standard error is approximately 0.00589.

The results from the `boot` function also indicate that the estimator is biased and a bias-corrected estimate of the failure rate is given by

$$0.00500 - 0.00109 = 0.00391.$$

□

3.2 Histograms

Although the distribution function completely determines the probability distribution of a random variable, many interesting and useful properties of a probability distribution, such as symmetry and whether the distribution is unimodal, are more easily determined from its density or frequency funciton.

Hence, for many purposes, a distribution's density function or frequency function provides more information about the distribution than does its distribution function.

The next two section present an introduction to estimation of density and frequency functions, focusing on "histogram estimators", the estimator implicitly used when constructing a histogram.

Although a histogram is a fairly simple way of estimating a density function, this simplicity has the advantage that the method is easy to describe and easy to analyze. The basic approaches used to derive the properties of histogram estimators form the basis for the methods used to analyze more sophisticated methods of estimation, such as kernel estimators, that we will consider in detail in the second half of the course.

3.3 Estimation of a frequency function

We begin by considering estimation a frequency function based on i.i.d. random variables $Y_1, Y_2, \ldots, Y_n$, each distributed according to the distribution with distribution function $F(\cdot)$. We assume that the distribution of Y_1 is discrete with frequency function $f(\cdot)$; for simplicity, take the range of Y_1 to be of the form $\{0, 1, \ldots, K\}$ for a positive integer K. It is straightforward to adapt the description there to cases in which the range of Y_1 is of a different form or in which the set of possible values of Y_1 is infinite, such as $(0, 1, 2, \ldots)$.

The frequency function corresponding to $F(\cdot)$ is a function $f(\cdot)$ on $(0, 1, \ldots, K)$ defined by

$$f(k) = \Pr(Y_1 = k), \quad k = 0, 1, \ldots, K$$

Then

$$F(y) = \sum_{k \leq y} f(k)$$

and

$$f(k) = \begin{cases} F(1) & \text{if } k = 1 \\ F(k) - F(k-1) & \text{if } k = 2, 3, \ldots, K \end{cases}. \tag{3.1}$$

To estimate $f(k)$, we may use the empirical frequency that $Y_j = k$:

$$\widehat{f}(k) = \frac{\#\left(Y_j = k\right)}{n}$$

so that, e.g., $\hat{f}(2)$ is simply the proportion of the random variables $Y_1, Y_2, \ldots, Y_n$ exactly equal to 2.

Let $\widehat{F}(\cdot)$ denote the empirical distribution function based on $Y_1, Y_2, \ldots, Y_n$. Then, as we have seen, for any y, $\widehat{F}(y)$ is an estimator of $F(y)$; furthermore, the frequency function corresponding to $\widehat{F}(\cdot)$ is exactly the estimator $\widehat{f}(\cdot)$ described previously. That is, using the relationship given in (3.1),

$$\widehat{f}(k) = \begin{cases} \widehat{F}(1) & \text{if } k = 1 \\ \widehat{F}(k) - \widehat{F}(k-1) & \text{if } j = 2, 3, \ldots, K \end{cases}. \tag{3.2}$$

A plot of $\widehat{f}(k)$ versus k, drawn as a bar chart, is known as a *histogram*.

Example 3.4 The R variable `betts` (available in the dataset "betts") contains the number of hits by Boston Red Sox outfielder Mookie Betts in each game he played during the 2016 season; thus, `betts` is a vector of length 158 containing integers $0, 1, 2, 3, 4, 5$. For example, the first ten values are

```
> betts[1:10]
 [1] 2 0 0 1 0 3 2 2 1 0
```

To estimate the frequency function corresponding to these data, we can use the function `table`, which counts the number of instances of each observed value in a given vector. Thus,

```
> table(betts)
betts
 0  1  2  3  4  5
30 61 52 12  2  1
```

shows that in 30 games Betts had 0 hits, in 61 games Betts had 1 hit, and so on. The estimated frequencies can be obtained by dividing the results of the `table` function by the number of values in the argument:

```
> table(betts)/length(betts)
betts
     0      1      2      3      4      5
0.1899 0.3861 0.3291 0.0759 0.0127 0.0063
```

Hence,

$$\widehat{f}(1) = 0.1899, \quad \widehat{f}(2) = 0.3861, \quad \widehat{f}(3) = 0.3291, \quad \widehat{f}(4) = 0.0759, \quad \text{and} \quad \widehat{f}(5) = 0.0063.$$

To obtain a histogram based on `betts`, we may use the command

```
> hist(betts, breaks=(-1:5)+0.5, freq=F)
```

which produces the histogram in Figure 3.1. Note that, in the function `hist`, the argument `breaks` specifies the "bins" used in construction of the histogram; using `breaks=(-1:5)+0.5` leads to one bin for each on the integers $0, 1, 2, 3, 4, 5$. Including `freq=F` instructs the function to display the relative frequencies, rather than the frequencies, on the y-axis.

The counts in each bin are available in the component `$counts` of the results of the command.

```
> hist(betts, breaks=(-1:5)+0.5, freq=F)$counts
[1] 30 61 52 12  2  1
```

□

If K is large relative to n that it may be useful to combine values when forming the histogram.

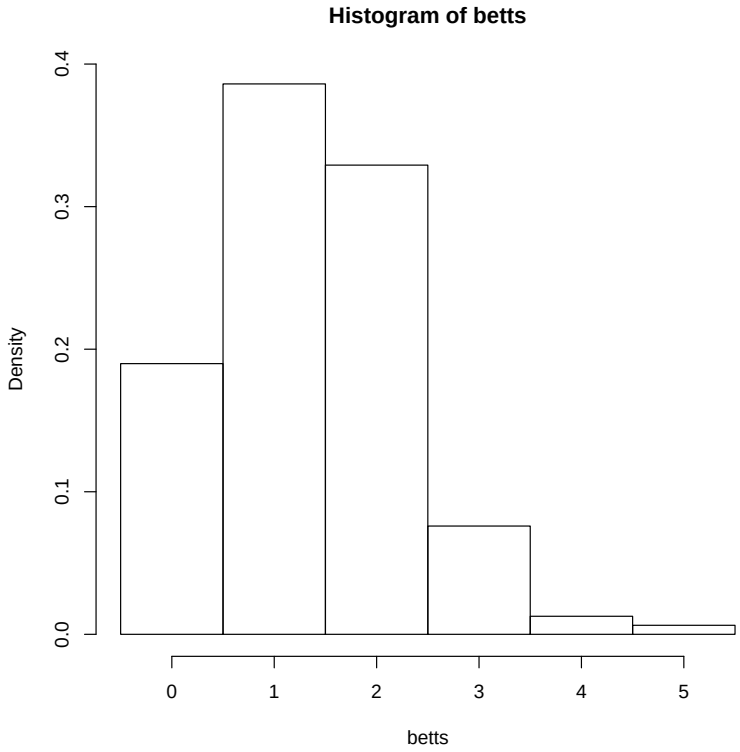

Figure 3.1: Histogram of the Betts Data

Example 3.5 The variable `ticks` (available in the dataset "ticks") contains the number of ticks counted on 82 sheep; the values range from 0 to 25. Figure 3.2 contains the histogram of the data based 26 bins, corresponding to the range of the variable, $0, 1, 2, \ldots, 25$. It is computed using the command

```
> hist(ticks, breaks=(-1:25)+0.5, freq=F)$counts
 [1]  4  5 11 10  9 11  3  5  3  2  2  5  0  2  2  1  1  0  0
[20]  0  0  1  1  1  0  2
```

Figure 3.3 contains a second histogram of these data, based on 13 bins, so that each bin includes two values: $(0, 1), (2, 3), \ldots, (24, 25)$. It was computed using the command

```
> hist(ticks, breaks=1+2*(-1:13), freq=F)$counts
 [1]  9 21 20  8  5  7  2  3  1  0  1  2  2  0
```

Note that the counts correspond to combining the counts from adjacent bins in the previous `hist` command. □

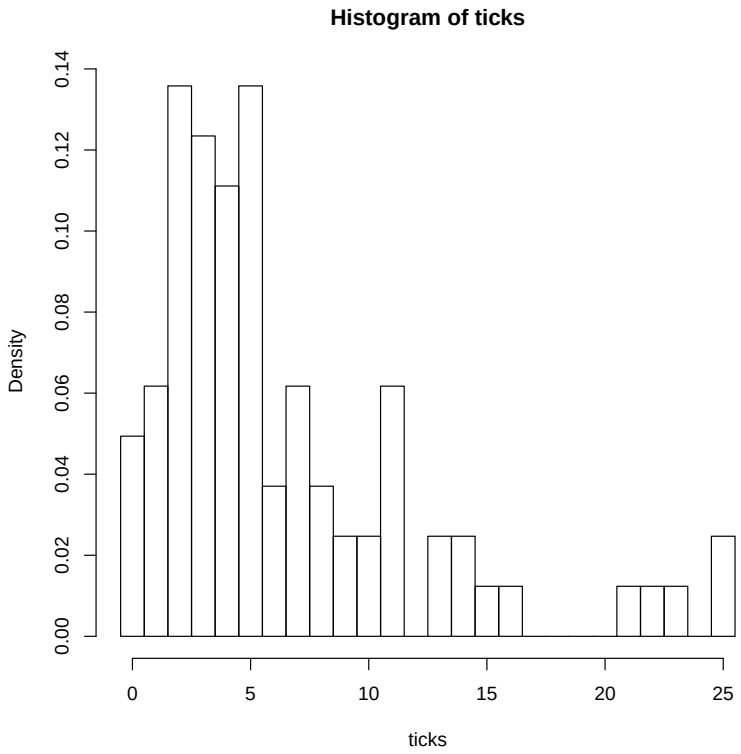

Figure 3.2: Histogram of the Ticks Data

3.4 Estimation of a density function

Now consider the more interesting, and more challenging, case in which the Y_j are i.i.d. random variables each with a continuous distribution and let $p(\cdot)$ denote the density function of the distribution. Hence,

$$F(y) = \int_{-\infty}^{y} p(t)dt, \quad -\infty < y < \infty$$

and, using standard results from calculus,

$$p(y) = F'(y)$$

for all y for which $F'(y)$ exists.

Throughout the course we will assume that all density functions are "smooth" functions. That is, for each density function $p(\cdot)$ under consideration there is an interval (a, b), $-\infty \leq a < b \leq \infty$, such that $p(y) > 0$ if and only if $y \in (a, b)$ and that $p(\cdot)$ a twice-differentiable function on (a, b). These conditions are satisfied for (nearly) all density functions encountered in applications.

Then $F'(y)$ exists for all $y \in (a, b)$ and, hence, the density function corresponding to $F(\cdot)$ satisfies $p(y) = F'(y)$ for all $y \in (a, b)$.

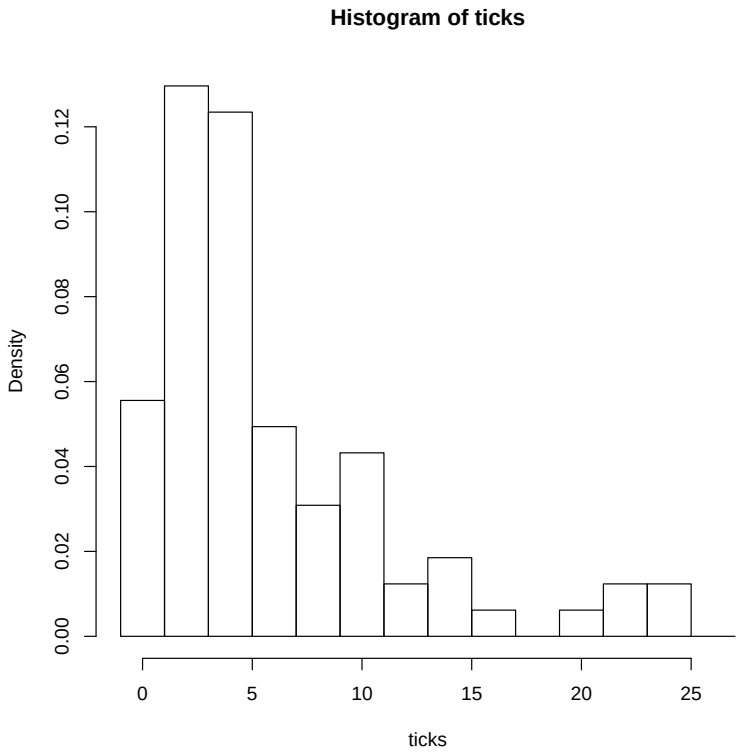

Figure 3.3: Histogram of the Ticks Data Based on Bins Consisting of 2 Values

An important property of the empirical distribution function is that it is the distribution function of a discrete distribution. That is, even if the true distribution of the data is continuous, the empirical distribution function for that data is a step function.

Thus, there is not a density function corresponding to $\widehat{F}(\cdot)$; it follows that it is not possible to estimate $p(\cdot)$ directly from the empirical distribution function.

However, it is possible to use $\widehat{F}(\cdot)$ to estimate $p(\cdot)$ by using *data binning* to construct the histogram:

(1) Divide the range of $Y_1, Y_2, \ldots, Y_n$ into m classes, or "bins", of equal length

$$(b_0, b_1],\ (b_1, b_2], \ldots, (b_{m-1}, b_m],$$

where $b_0 < b_1 < \cdots < b_m$ with $b_0 \leq \min(Y_1, Y_2, \ldots, Y_n)$ and $b_m \geq \max(Y_1, Y_2, \ldots, Y_n)$. Let

$$h = b_k - b_{k-1}, \quad k = 1, 2, \ldots, m$$

denote the *bin length*.

(2) Let

$$f_k = \#\left(\text{ observations in } (b_{k-1}, b_k]\right), \quad k = 1, 2, \ldots, m.$$

Note that f_k is the frequency of observations in the interval $(b_{k-1}, b_k]$ and that

$$f_k = n\left(\widehat{F}(b_k) - \widehat{F}(b_{k-1})\right).$$

(3) Define

$$\widehat{p}_H(y) = \frac{f_k}{nh} \quad \text{for} \quad b_{k-1} < y \leq b_k$$

and set $\widehat{p}_H(y) = 0$ for y outside the range $(b_0, b_m]$.

The estimator $\widehat{p}_H(\cdot)$ is known as the *histogram estimator* of $p(\cdot)$.

Example 3.6 The R variable `speed` (available in the dataset "Michelson") contains Michelson's determinations of the velocity of light in air; the values given +299000 are the determinations of velocity in km per sec. The command

```
> hist(speed, breaks=20, freq=F)
```

computes a histogram of the data using 20 bins. Note that when `breaks` is specified as a single integer, it is interpreted as the number of bins. However, the algorithm chooses "nice" break points, so that the number of bins used does not always match the number requested. The argument `freq =F` specifies that the scale on the y-axis of the histogram is taken to be the density estimates. A plot of the histogram is given in Figure 3.4. □

Some properties of the histogram density estimator

We now consider some properties of $\widehat{p}_H(\cdot)$ as an estimator of $p(\cdot)$. First note that $\widehat{p}_H(\cdot)$ is a genuine density function: it is non-negative, that is,

$$\widehat{p}_H(y) \geq 0 \quad \text{for all} \quad -\infty < y < \infty$$

and it integrates to 1,

$$\begin{aligned}
\int_{-\infty}^{\infty} \widehat{p}_H(y)dy &= \int_{b_0}^{b_m} \widehat{p}_H(y)dy \\
&= \sum_{k=1}^{m} \int_{b_{k-1}}^{b_k} \widehat{p}_H(y)dy \\
&= \sum_{k=1}^{m} \int_{b_{k-1}}^{b_k} \frac{f_k}{nh}dy \\
&= \sum_{k=1}^{m} \frac{f_k}{nh}(b_k - b_{k-1}) = \frac{1}{n}\sum_{k=1}^{m} f_k \\
&= 1.
\end{aligned}$$

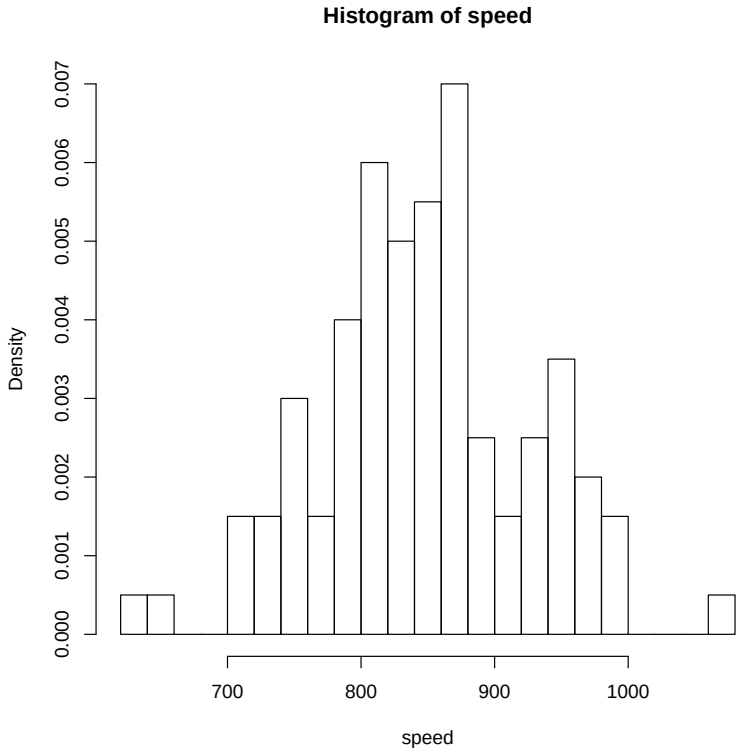

Figure 3.4: Histogram of the Michelson Speed of Light Data

Here we use the fact that $b_k - b_{k-1} = h$ for all $k = 1, 2, \ldots, m$.

The statistical properties of $\widehat{p}_H(y)$, such as its bias and variance, can be determined by noting that f_k is a binomial random variable with index n and "success probability" $F(b_k) - F(b_{k-1})$. It follows that, for $y \in (b_{k-1}, b_k]$,

$$\mathrm{E}\left(\widehat{p}_H(y)\right) = \frac{\mathrm{E}\left(f_k\right)}{nh} = \frac{F(b_k) - F(b_{k-1})}{h} = \frac{F(b_{k-1} + h) - F(b_{k-1})}{h}.$$

Recall that, using the definition of the derivative of a function,

$$\lim_{h \to 0} \frac{F(b_{k-1} + h) - F(b_{k-1})}{h} = F'(b_{k-1})$$

and, because $|y - b_{k-1}| \leq h$,

$$F'(b_{k-1}) \doteq F'(y)$$

when $|h|$ is small. It follows that, when $h \doteq 0$,

$$\mathrm{E}\left(\widehat{p}_H(y)\right) \doteq F'(y) = p(y).$$

That is, when the bin width is small, $\widehat{p}_H(y)$ is an approximately unbiased estimator of $p(y)$. Thus, if we want $\widehat{p}_H(y)$ to have small bias when the sample size is large (so that

the estimator is very accurate in large samples, clearly a desirable property), we must have $h \to 0$ as $n \to \infty$.

Now consider the variance of $\widehat{p}_H(y)$ for $y \in (b_{k-1}, b_k]$: using the properties of the binomial distribution,

$$\begin{aligned}\text{Var}\left(\widehat{p}_H(y)\right) &= \frac{\text{Var}(f_k)}{(nh)^2} = \frac{n[F(b_k) - F(b_{k-1})][1 - (F(b_k) - F(b_{k-1}))]}{(nh)^2} \\ &= \frac{F(b_k) - F(b_{k-1})}{h}\frac{1}{nh} - \left(\frac{F(b_k) - F(b_{k-1})}{h}\right)^2 \frac{1}{n}. \end{aligned} \tag{3.3}$$

If $h \doteq 0$, then

$$\frac{F(b_k) - F(b_{k-1})}{h} \doteq p(y)$$

and

$$\text{Var}\left(\widehat{p}_H(y)\right) \doteq p(y)\frac{1}{nh} - p(y)^2\frac{1}{n}.$$

Note that, for small h, $1/(nh)$ is larger than $1/n$ and if h is near 0 it is much larger. Hence, for $h \doteq 0$,

$$\text{Var}\left(\widehat{p}_H(y)\right) \doteq p(y)\frac{1}{nh}$$

which is relatively large.

That is, the bias of $\widehat{p}_H(y)$ can be made to be small by choosing the bin width to be small; however, doing so makes the variance of $\widehat{p}_H(y)$ large. Conversely, we can make $\text{Var}\left(\widehat{p}_H(y)\right)$ small by using a relatively large value of h (i.e., a value not near 0); however, doing so leads to an estimator $\widehat{p}_H(y)$ with large bias.

To have a small bias and small variance when the sample size is large, h must be close to 0 when n is large and nh must be large when n is large. These requirements can be expressed mathematically as

$$h \to 0 \quad \text{as} \quad n \to \infty,$$

that is, h is small when n is large, and

$$nh \to \infty \quad \text{as} \quad n \to \infty,$$

that is, nh is large when n is large.

Thus, in practical terms, for the histogram estimator to be an accurate estimator of the density $p(\cdot)$ when n is large, we need the bin width h to be small when n is large, but not so small that nh is also small.

These properties are not unexpected. When h is large, the histogram is based on a few, long bins and, hence, the estimate of $p(y)$ is based on values of Y_j that are not particularly

close to y; this leads to a large bias for $\widehat{p}(y)$. However, these long bins include many observations, so that the variance of the estimator $\widehat{p}(y)$ is relatively small. On the other hand, when using a small value of h, the estimate of $p(y)$ only includes those Y_j that are close to y, reducing the bias; however, there are relatively few such Y_j so that the variance of $\widehat{p}(y)$ is large.

This is an example of the *bias-variance tradeoff* that is common in nonparametric function estimation (as well as in other areas of statistics): estimators with small bias often have large variance and vice versa. This fact will arise repeatedly throughout the course and it has a large impact on the methodology.

3.5 Exercises

3.1. The dataset "Peabody" in Canvas contains scores on the Peabody Picture Vocabulary Test given to a sample of preschool children as part of a study on early childhood education.

(a) Compute the sample mean of the peabody data, along with the standard error of the sample mean using the usual $S/\sqrt{n}$ formula.

(b) Use the R function `median` to calculate the sample median of the peabody data.

(c) Unlike the sample mean, there is not a simple formula for the standard error of the sample median. Hence, the bootstrap provides a useful approach. Use the bootstrap to find the standard error of the sample median for the peabody data. Note that, although there is a base R function for the sample median (`median`), you will need to write a function that can be used in the bootstrap function `boot`. Use 100000 bootstrap replications and use the random seed `35202`.

(d) Suppose that we are interested in comparing the sample mean and sample median. Calculate $\bar{Y} - m$ for the peabody data, where $\bar{Y}$ is the sample mean and m is the sample median. Use the bootstrap method to find the standard error of $\bar{Y} - m$; use 100000 bootstrap replications and use the random seed `35202`. Is there evidence that the true mean and median differ for the peabody data? Why or why not?

3.2. Consider the failure data for a software system used in a previous exercise, and available in the dataset "software". Suppose that we are interested in whether or not these data follow an exponential distribution.

Note that if Y follows an exponential distribution, then the mean and standard deviation of Y are equal. Hence, if the data $Y_1, Y_2, \ldots, Y_n$ follow an exponential distribution, the value

of the statistic

$$T = \frac{\bar{Y}}{S}$$

should be about 1, where $\bar{Y}$ and S^2 are the sample mean and sample variance, respectively, of the data.

(a) Calculate the value of T for the software failure data.

(b) Using the bootstrap method, find the bias and standard error of T. Use $100,000$ bootstrap replications and use the random seed 35203.

(c) Based on these results, is there evidence that the distribution of the software failure data is not an exponential distribution? Why or why not?

3.3. Consider i.i.d. random variables $Y_1, Y_2, \ldots, Y_n$ each distributed according to the distribution with frequency function

$$\theta(1-\theta)^x, \quad x = 0, 1, 2, \ldots$$

where $0 < \theta < 1$; note that this is a geometric distribution. The maximum likelihood estimator of θ may be shown to be

$$\frac{1}{1+\bar{Y}}, \quad \text{where } \bar{Y} = \frac{1}{n}\sum_{j=1}^{n} Y_j.$$

(a) Consider a set of observations

```
3 1 0 0 0 2 1 2 3 0 0 1
```

corresponding to the random variables $Y_1, Y_2, \ldots, Y_n$ described above (for $n = 12$). Based on these observations, find the maximum likelihood estimate of θ.

(b) Using the bootstrap method, find the bias and standard error of the maximum likelihood estimator; use $100,000$ bootstrap replications and set the random seed to 35204.

3.4. The dataset "crustaceans" contains the number of a certain type of crustacean found in a sample of marine plankton, for 150 such samples. Construct a histogram of these data. Choose the number of cells subjectively, with the goal of constructing histogram that provides useful information regarding the distribution of crustacean counts in samples of marine plankton.

3.5. Consider the failure data for a software system used in Exercises 3.6 and 4.2, and available in the dataset "software".

(a) Construct a histogram for the failure data using the default number of bins. Note that to use the default number of bins, simply omit the argument `breaks` when using the function `hist`.

(b) Construct a histogram for the failure data using 5 bins.

(c) Construct a histogram for the failure data using 15 bins.

(d) Construct a histogram for the failure data using 20 bins.

(e) Which of the three histograms constructed in parts (a) – (d) appears to be most useful for describing the distribution of the failure data?

3.6. The purpose of this exercise is to study the relationship between the number of bins, the accuracy of a histogram estimator, and the sample size.

(a) Plot the density function of the chi-squared distribution with 3 degrees-of-freedom. Use the function `dchisq` to compute the density function; the argument for the degrees-of-freedom is `df`. Evaluate the density function at the values in `seq(0, 12, 0.001)`.

(b) Generate a sample of size 100 of random variates from an chi-squared distribution with 3 degrees-of-freedom (use `rchisq`); use the random seed 35206.

(c) Construct a histogram of these data with `breaks` taken to be 5 and save the results to a variable; call it `hout`.

(d) Let `mid` denote the component `$mid` of the histogram result from part (a) and let `den` denote the density estimate given in component `$density`. Then `den` contains a vector of estimates of the density function of the exponential distribution with rate parameter 1 evaluated at the values in `mid`.

Therefore, the values in `den` are estimates of the values in `dchisq(mid, df=3)` (the function `dchisq` computes the density of the chi-squared distribution.

Hence,

```
> mean((den - dchisq(mid, df=3))^2)^.5
```

is the square root of the average squared error of the histogram density estimate, where the average is over the values in `mid`. Compute this value for the histogram estimate computed in part (b).

(e) Repeat parts (c) and (d) twice, with the value of `breaks` taken to 10 and 20, respectively.

(f) Based on these results, which value of `breaks` produces the most accurate density estimate?

(g) Repeat parts (b) - (f) for a sample of size 500; use the same random seed (35206).

(h) Repeat parts (b) - (f) for a sample of size 1000; use the same random seed (35206).

(i) Based on these results, what do you conclude about the the relationship between the number of bins, the accuracy of a histogram estimator, and the sample size. Does this conclusion agree with the theoretical results given in the notes?