Laboration: Resampling Methods

Simon Sigurdhsson

September 17, 2010

Abstract

Resampling methods are an essential tool when analyzing sample data. Two of the more easily implemented methods are bootstrapping and jackknifing. Also, the Glivenko-Cantelli theorem is an essential foundation for the bootstrapping method.

1 The Glivenko-Cantelli theorem

The Glivenko-Cantelli theorem quite simply states that the empirical distribution, given enough data (i.e. infinitely many samples), will converge toward the true distribution of the sample data; and it will do this with probability 1. In short, the theorem looks like this:

$$P\left\{\lim_{n\to\infty}\max_{x\in\mathbb{R}}\left|\tilde{F}(x)-F(x)\right|=0\right\}=1$$

To illustrate this convergence, we make use of another theorem which is much easier to prove (although I won't):

$$\max_{x\in\mathbb{R}}\left|\tilde{F}(x)-F(x)\right|=\max_{1\leq i\leq n}\max\left\{\left|\frac{i-1}{n}-F(X_{(i)})\right|,\left|\frac{i}{n}-F(X_{(i)}\right|\right\}$$

Making use of this second theorem, and our sample data $F(X_{(i)})$ found in `boot1.dat`, we can illustrate the convergence by computing the value for some different n. In our case, these n are taken from the sequence $i^2,\ i=1,2,\ldots 40$.

To accomplish this task, the R programming language is used. First, we load the data from `boot1.dat`. After this, we simply compute the values needed, and extract the appropriate maxima from the calculated data:

```
data {\leftarrow} scan(‘‘boot1.dat’’)
limit = c();
for(i in 1:40){
  n {\leftarrow} i**2;
  max1 {\leftarrow} abs(0:(n-1)/n - sort(data[1:n]));
  max2 {\leftarrow} abs(1:n/n - sort(data[1:n]));
  limit {\leftarrow} c(limit, max(pmax(max1, max2)));
}
```

Lastly, to produce the illustrative plot seen in figure 1, we plot the data using the built-in `plot` and `postscript` functions:

```
postscript("limit.eps");
plot((1:40)**2, limit, xlab=''n'', ylab=''limit'');
dev.off();
```

As seen in the figure, the limit converges rather quickly (although not uniformly).

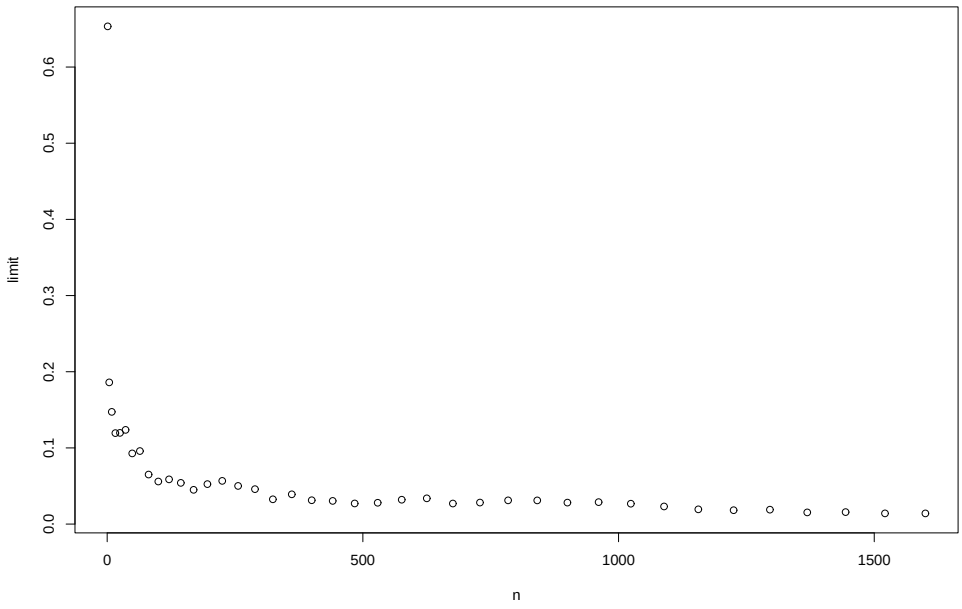

Figure 1: The convergence of the Glivenko-Cantelli theorem for the given data sample

2 Bootstrapping the properties of an estimator

The theoretical standard deviation, $\sqrt{\mathbf{Var}\{X\}}$, can be estimated by an estimator normally called the "sample standard deviation", defined as such:

$$\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

Since this is an estimator, it's of some interest to calculate the variance (and also bias) of this estimator. This can be done using either the *bootstrap* or *jackknife* methods. We're going to test both of them, on a sample of generated uniformly distributed data. First up is the bootstrap method, with $N = 200$ simulated observations. We start by generating these 200 observations, and calculating their sample standard deviation:

```
N <- 200; d <- c();
for(a in 1:N){
  sample <- data[as.integer(runif(n, 1, n+1))];
  d <- c(d, sd(sample));
}
varboot <- sum((d-mean(d))**2)/(N-1);
```

Here, `n` is defined to be either 20 or 100, as we are interested in the variance in both cases. This is the number of samples (n) in each observation. We also assume that the data in `boot2.dat` has been loaded into the variable `data`.

For $n = 20$, we obtain a value of roughly 0.0006 — for $n = 100$ this value has risen to roughly 0.0028. This is expected, since the sample size was increased by roughly the same factor.

Now, calculating the jackknife variance is somewhat more involved; we need to split the (only) sample we use and get n different sample standard deviances for these n different cut samples. After this, we simply calculate the jackknife variance:

```
sample <- data[1:n];
for(i in 1:n){
  a <- if(i == 1){ sample[2:n]
       }else if(i == n){ sample[1:(n-1)]
       }else{ c(sample[1:(i-1)],sample[(i+1):n])};
  j <- c(j, sd(a));
}
varjack <- (n-1)*sum((j-mean(j))**2)/n;
```

In the above code, we take advantage of the fact that the `if` statement has a return value in R. The values obtained using this method are 0.0026 for $n = 20$ and 0.0007 for $n = 100$ samples. This is close to the bootstrap method, which is to be expected; but since they are different methods (and operating on slightly different data sets) their values differ slightly.

Further, we would like to estimate the *bias* $\mathbf{E}\{\hat{\sigma}(X_1, \ldots, X_n)\} - \sigma(X)$. This cannot be calculated analytically, because the distribution is unkown. Therefore, we approximate the bias using the empirical distribution, enabling us to use the bootstrap method to estimate the bias. The actual formula for doing this is as follows:

$$\frac{1}{N}\sum_{i=1}^{N}\hat{\sigma}(\tilde{X}_1^{(i)}, \ldots, \tilde{X}_n^{(i)}) - \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}$$

Where n and N represent the sample size and number of samples, like before. Converting this to a programmatic R form, we get the following:

```
biasboot <- mean(d) - sqrt(mean(m));
```

Where `m` is calculated in the bootstrap loop above:

```
m <- c(m, (sample-mean(sample))**2);
```

Similarly, we can compute the jackknife bias using the formula $(n-1)(\hat{\theta}^{(\cdot)} - \hat{\theta})$. This is a bit easier to implement in R, since we already have all variables:

```
biasjack <- (n-1)*(mean(j)-sd(sample));
```

For $n = 20$, the bootstrap method gives a bias of 0.009 while the jackknife bias is -0.003. For $n = 100$, the values are 0.002 and -0.001, respectively.

It is also of interest to calculate a confidence interval for the bootstrap, since it to is an estimation. This is not neccesary for the jackknife method, since it is a *calculation* as opposed to an *estimation*.

The bootstrap confidence interval can be approximated using the data generated by our Monte Carlo method above. Since we have $N = 200$ observations of the estimator, we can choose a confidence interval so that $p\%$ of these observations fall outside the interval; this i easy to do in R:

```
p <- 0.05;
ca <- as.integer(length(d)*p/2);
cb <- length(d)-ca;
ci <- c(sort(d)[ca+1], sort(d)[cb]);
```

Here, `ci` is the confidence interval. Keep in mind that we have our observations in the variable `d`. The variable `p` represents the confidence level; in this case, $p = 0.05$ means we want the 95% confidence interval.

Using the data obtained above, a confidence interval of $[0.377, 0.567]$ was returned for $n = 20$, while a tighter confidence interval of $[0.486, 0.570]$ was returned for $n = 100$. This seems reasonable; a histogram of standard deviance for the simulated observations can be seen in Figure 2.

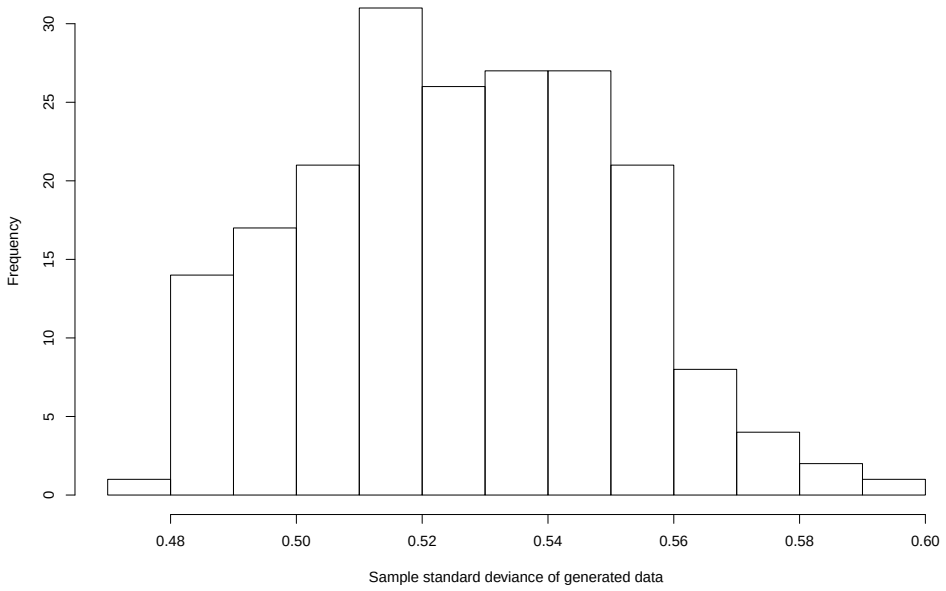

Figure 2: Sample standard deviance for simulated samples with $n = 100$