Advanced Quantitative Methods: Bootstrap and simulation

Johan A. Elkink

University College Dublin

27 April 2018

Outline

Simulation and bootstrapping

Used for:

- Gaining **intuition** about distributions and sampling
- Providing **distribution** information not directly available
- Acquiring **uncertainly** estimates

Both simulation and bootstrapping are **numerical approximations** of the quantities we are interested in. Run the same code twice, and you get different answers!

Bootstrapping vs simulation

Simulation: taking random draws from the estimated parameters and their distribution. E.g. all maximum likelihood estimates have a normal distribution, so you take draws from the multivariate normal distribution defined by mean $\hat{\boldsymbol{\theta}}$ and variance-covariance matrix $V(\hat{\boldsymbol{\theta}}) = -(\mathbf{H}^{-1})$.

Bootstrapping: taking random samples, with replacement, from the original data and then re-estimate the model. The distribution of some statistic across iterations will be the sampling distribution of that statistic.

Outline

Outline

Principle

For many models the estimated parameters or statistics will have a normal distribution.

- $\bar{x}$ as estimator of μ_x has a normal distribution with standard error $\sigma_x/\sqrt{n}$.
- $\hat{\beta}^{OLS}$ as an estimator of $\boldsymbol{\beta}$ has a normal distribution with standard error $\sqrt{V(\hat{\beta}^{OLS})}$.
- $\hat{\boldsymbol{\theta}}^{ML}$ as an estimator of $\boldsymbol{\theta}$ has a normal distribution asymptotically with standard error $\sqrt{-(\mathbf{H}^{-1}_{\hat{\boldsymbol{\theta}}^{ML}})}$.

We can take random draws from this multivariate normal distribution, rather than just the estimated parameters, to translate the estimation uncertainty into prediction uncertainty.

Example

$$\pi_i = \frac{1}{1 + e^{-x_i \beta}}$$

Say, we find $\hat{\beta} = .4$ and we have a value we are interested in $x = 3$, then $\hat{\pi} = \hat{P}r(y = 1 | x = 3) = \frac{1}{1+e^{-3\times .4}} = .77$.

However, we know that this $\hat{\beta}$ is not exact, but that there is some uncertainty around this estimate, due to our sample being of finite size. So we estimate $V(\hat{\beta})$ using the Hessian matrix. Say, $V(\hat{\beta}) = .1$. In the case of a bivariate analysis like this, you can just take the boundaries: $\hat{\beta} - 1.96\sqrt{.1} \Longrightarrow \hat{\pi} = .34$ and $\hat{\beta} + 1.96\sqrt{.1} \Longrightarrow \hat{\pi} = .96$. So $\hat{\beta}$ is somewhere between .34 and .96.

Multivariate normal distributions

The confidence interval is easy to calculate for univariate normal distributions, but becomes difficult for multivariate ones.

Instead, we can run simulations:

1. Estimate model.
2. Draw random $\boldsymbol{\theta}^*$ from $N(\hat{\boldsymbol{\theta}}, -(\mathbf{H}^{-1}))$.
3. Predict $\mathbf{y}^*$ given $\boldsymbol{\theta}^*$.
4. Repeat m times (i.e. m random draws and predictions).
5. Look at distribution of $\mathbf{y}^*$'s.

Outline

1. Introduction
2. Simulation
   - Principle
   - Implementation
3. Bootstrap
   - Principle
   - Implementation
4. Monte Carlo studies

Implementation

In library `MASS` there is a function `mvrnorm()` to draw random numbers from a multivariate normal distribution.

```
mlest <- optim(...)
m <- 100
x.star <- c(...) ## some point of interest
theta.star <- mvrnorm(m, mlest$par,
        sqrt(-mlest$hessian))
pi.star <- 1/(1+exp(-x.star %*% t(theta.star)))
pi.star <- pi.star[1,] ## convert matrix to vector

se.pi <- sd(pi.star)
ci.pi <- c(mean(pi.star) - 1.96 * se.pi,
           mean(pi.star) + 1.96 * se.pi)
```

Example (logit)

$\hat{\pi}$ cannot really be normally distributed (values outside the 0-1 range are not allowed), so a more nonparametric approach is better:

```
ci.pi <- quantile(pi.star, c(.025, .975))
```

You can also do a more graphical inspection:

```
plot(density(pi.star))
```

Outline

Outline

Bootstrapping

The simulation just described makes strong parametric assumptions about the estimated parameters.

A more flexible approach is bootstrapping.

Like with simulated parameters, the purpose of bootstrapping is to get an estimate of the uncertainty of an estimate.

The bootstrap is a **nonparametric** approach to estimating the standard error on your estimate, since no assumptions are made about the distribution of the underlying data or of the sampling distribution of the parameters.

Bootstrapping

1. Take a random sample of the original data, of the same size, sampled *with replacement*.
2. Estimate $\hat{\theta}^*$ on the new sample.
3. Repeat m times.
4. Take the distribution of the m $\hat{\theta}^*$'s as the sampling distribution of $\hat{\theta}$.

For estimating the standard error, m should be at least 200 or so.

For estimating a confidence interval (one that is not based on just estimating the standard error), m should be at least 1000 or so.

Bootstrap estimates of uncertainty can be **biased**, but are generally **consistent**.

Outline

Bootstrapping: example

```r
mean.age <- NULL
for (i in 1:1000) {
  age.bootstrap <- sample(age, 100, replace=TRUE)
  mean.age[i] <- mean(age.bootstrap)
}
summary(mean.age)
```

```r
quantile(mean.age, c(.05, .95))
```

Outline

Monte Carlo studies

Used to study **small-sample properties** of estimators when only asymptotics are known.

Based on computer **simulations** and due to computational costs only recently becoming common.

1. Model the data-generating process (DGP)
2. Generate artificial data sets
3. Create estimates of the underlying parameters using the estimator you are testing
4. Assess the estimator's efficiency, bias and MSE relative to the (known) data
5. ... or, for tests, check proportion Type I and Type II errors

Monte Carlo: example

In R, the linear model is estimated with the following command:

```r
summary(m <- lm(y ~ x))
```

or explicitly without constant:

```r
summary(m <- lm(y ~ 0 + x))
```

We can test this estimator by creating fake datasets, following the steps outlined.

Monte Carlo example: DGP

Data generating process (DGP):

$$\begin{aligned} x_i &\sim N(5, 2) \\ \varepsilon_i &\sim N(0, 1) \\ y_i &= 3x_i + \varepsilon_i \end{aligned}$$

Monte Carlo example: artificial datasets

Artificial datasets (R datasets of N cases each):

```
N <- 50
R <- 1000
for (i in 1:R) {
        x <- rnorm(N, 5, 2)
        e <- rnorm(N, 0, 1)
        y <- 3 * x + e
}
```

Monte Carlo example: estimations

Insert the estimation itself:

```
N <- 50
R <- 1000
estimates <- rep(NA, R)
for (i in 1:R) {
        x <- rnorm(N, 5, 2)
        e <- rnorm(N, 0, 1)
        y <- 3 * x + e
        estimates[i] <- coef(lm(y ~ 0 + x))
}
```

Monte Carlo example: assessment

- Efficiency

```
plot(density(estimates))
sd(estimates)
```

- Bias

```
mean(estimates - 3)
```

- Mean squared error (MSE)

```
var(estimates) + mean(estimates - 3)^2
```