Stochastic simulation and resampling methods

Statistical modelling: Theory and practice

Gilles Guillot

gigu@dtu.dk

October 22, 2013

1. Simulations, what for?
2. Pseudo uniform random number generation
3. Beyond uniform numbers on $[0, 1]$
4. Random number generation in R
5. An application: the bootstrap
6. Exercises

Analysis of a stochastic model

- A stochastic model involving
  - D: something observed (or observable) about the system (data)
  - θ: a known or unknown parameter
  - $P(\theta, D)$: a probability model (*not* necessarilly known as a nice function)
- Three questions
  - One has observed some data D, what does it say about θ? (inference)
  - If θ is known to be equal to θ_0, what does it say about D? (prediction)
  - One has observed some data D, what does it say about future data value D^*? (joint inference and prediction)
- Stochastic simulations used to produce some "fake data" used to understand/learn something about the data generating process.
- Much more in June course Stochastic Simulation 02443.

Stochastic simulations as a tool in ordinary numerical analysis

- Integration in large dimension
- Optimization and equation solving

Pseudo uniform random number generation

- Computer = most deterministic thing on earth
- Idea here: produce deterministic sequence that mimics randomness
- Methods originate in von Neuman's work in theoretical physics during WWII coined "Monte Carlo" simulations
- Today's lecture:
  - basic idea used to produce uniform numbers in $[0, 1]$
  - a statistical application: the bootstrap method

Some examples of more or less "random uniform" sequences in $\{0, 1\}$ (not to be mixed up with $[0, 1]$!!)

Some examples of sequences in $\{0, 1\}$:

- 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...
- 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 ...
- 0 0 1 0 1 0 0 0 1 0 1 0 0 1 1 0 1 0 1 0 1...

- To look random and uniform on $\{0, 1\}$, a sequence must have the correct proportion of 0s and 1s.
- Pairs of outcome should have the correct proportion of $0, 0$ and $0, 1$ and $1, 1$.

Randomness relates to complexity: length of the shortest algorithm required to produce this sequence (Kolmogorov, Martin-Löf).

(Pseudo) Uniform random number generator on $\{0, 1\}$

Definition

A (uniform) RNG on $\{0, 1\}$ is an algorithm that returns numbers such that

- $\frac{\#0}{n} \longrightarrow 1/2$ as $n \longrightarrow \infty$
- And
  - $\frac{\#(0,0)}{n} \longrightarrow 1/4$ as $n \longrightarrow \infty$
  - $\frac{\#(0,1)}{n} \longrightarrow 1/4$ as $n \longrightarrow \infty$
  - $\frac{\#(1,0)}{n} \longrightarrow 1/4$ as $n \longrightarrow \infty$
  - $\frac{\#(1,1)}{n} \longrightarrow 1/4$ as $n \longrightarrow \infty$
- And so on for any order...

Such a sequence is also said to be k-uniform on $\{0, 1\}$ for any k and mimics what we expect of a sequence of i.i.d $b(1/2)$ random variables.

(Pseudo) Uniform random number generator on $[0, 1]$

Definition

A (uniform) RNG on $[0, 1]$ is an algorithm that returns numbers such that

- $\frac{\#x_i \in [a, b]}{n} \longrightarrow (b - a)$ as $n \longrightarrow \infty$
- $\frac{\#(x_i, x_j) \in [a, b] \times [c, d]}{n} \longrightarrow (b - a) \times (d - c)$ as $n \longrightarrow \infty$ for $a, b, c, d \in [0, 1]$
- And so on for any order...

Such a sequence is also said to be k-uniform on $[0, 1]$ for any k and mimics what we expect of a sequence of i.i.d $U([0, 1])$ random variables.

Pseudo random uniform numbers on $[0, 1]$: the congruential method.

The congruential method takes advantage of the erratic occurence of prime numbers.
It produce sequences in a discrete set $\{0, M\}$. To lie in $[0, 1]$, the sequence has to be re-scaled.

Congruential method

1. Choose a, b, M in $\mathbb{N}$
2. Init x_1 at some arbitrary value in $\{0, ..., M-1\}$
3. Iterate $x_i := ax_{i-1} + b \ (mod\ M)$
4. $y_i = x_i/M$ (rescaling)

Some R code for the congruential method

```
require(gmp); require(gplots)
a <- 1000
b <- 0
M <- 2001179
isprime(M)
n <- 10000 #length of sequence
x <- rep(NA,n)

x[1] <- 220472 # init
for(i in 2:n)
  {
    x[i] <- (a*x[i-1]+b) %%M
  }
y <- x/M # rescaling
par(mfrow=c(1,2))
hist(y,prob=TRUE)
plot(y[-1],y[-n],pch=".")
```

Remarks on the congurential method

- Method given to produce numbers on $\{0, M\}$
  RNGs on $[0, 1]$ obtained by re-scaling
- Note the analogy with a roulette
- Algorithm is deterministic: same initial state produces same sequence
- Algorithm is periodic
- Fast
- Efficiency (ability to look random) depends strongly on a, b and M

Assessing the quality of a random number generator

- Histogram
- Bidimensional histogram
- Auto-correlation function: $\rho(h) = c(h)/Var(X)$
  where $c(h) = Cov(X_k, X_{k+h})$
- Formal statistical test: Kolmogorov-Smirnov (KS) test:
  - based on $\max |F(x) - F^\star(x)|$
  - returns a p-value
    - measures how much the data conflict with H_0
    - a small p-value indicates a conflict between data and H_0
    - a large p-value indicates the absence of conflict between data and H_0

A state-of-the-art method: the Mersenne twister

- Previous method in use untill late 80's
- Mersenne twister: algorithm proposed by Matsumoto and Nishimura in 1998
- Iterative method
- Based on a matrix linear recurrence
- Has period $2^{19937} - 1$ when using 32-bit words
- Name derives from the fact that period length is chosen to be a Mersenne prime number
- Best pseudo-RNG to date
- Default algorithm in R (cf `? RNG`), Splus and Matlab

Simulating non-uniform random numbers

Most common families of methods include:

- Rejection methods
- Transformation methods
- Markov chain methods

R functions for random number simulation

- All common distributions are implemented
- Arguments include the desired sample size and distribution parameters (sometimes several parametrizations available)
- Examples:
  - Continuous uniform $\mathcal{U}([a,b])$: `runif(n,min=a,max=b)`
  - Univariate normal $\mathcal{N}(\mu,\sigma)$: `rnorm(n,mean=mu,sd=sigma)`
  - Exponential $\mathcal{E}(\lambda)$: `rexp(n,rate=lambda)`
  - Discrete with arbitrary weights (including uniform):: `sample(size,x,replace,prob)`
  - See on-line help: `? Distributions`

The bootstrap

Introductory example: estimating the accuracy of an estimator of the mean

- Consider the problem of estimating the expectation μ of a distribution with unknown variance σ^2 from an i.i.d sample $X_1, ..., X_n$.
- A natural estimator of μ is $\hat{\mu} = \bar{X}_n$
- The accuracy of $\hat{\mu}$ can be assessed by computing $MSE(\hat{\mu}) = V(\hat{\mu})$ (since $E(\hat{\mu}) = \mu$)
  - We know that $V(\hat{\mu}) = \sigma^2/n$
  - σ^2 can be estimated by the sample variance $\sum(X_i - \bar{X}_n)^2/(n-1)$
  - Hence $\widehat{MSE(\hat{\mu})} = \sum(X_i - \bar{X}_n)^2/(n^2 - n)$

Now imagine for a while that we do not know that $V(\hat{\mu}) = \sigma^2/n$ or that $\widehat{\sigma^2} = \sum(X_i - \bar{X}_n)^2/(n-1)$

A fairly general situation:

- An unknown parameter θ
- An estimator $\hat{\theta}$
- No known formula for the variance (or MSE) of $\hat{\theta}$

Back to the variance of $\hat{\mu}$

- By definition:
  variance = "expected square of the difference with expectation"
- $V(\bar{X}_n)$ can be thought of as the number obtained as follows
  - Take a 1st i.i.d sample of size n, compute $\bar{X}_n^{(1)}$
  - Take a 2nd i.i.d sample of size n, compute $\bar{X}_n^{(2)}$
  - $\vdots$
  - Take a B-th i.i.d sample of size n, compute $\bar{X}_n^{(B)}$
- Compute $\widehat{V(\bar{X}_n)} = \frac{1}{B}\sum_{j=1}^{B}\left(\bar{X}_n^{(j)} - 1/B\sum_{j=1}^{B}\bar{X}_n^{(j)}\right)^2$

R illustration of the previous (useless) idea

```
n <- 15 ;x <- rnorm(n=n,mean=2,sd=1)
func1 <- function()
{
  col <- col + 1
  y <- rnorm(n=n,mean=2,sd=1)
  points(y,rep(0,n),col=col,cex=2)
  abline(v=mean(y),col=col,pch=16,cex=4)
  col
}
col <- 1
plot(x,rep(0,n),type="p",xlab="Data",
     ylab="",xlim=c(-1,4),cex=2)
abline(v=mean(x),col=1,pch=16,cex=4)
col <- func1()
```

- The previous quantity estimates $V(\hat{\mu})$ more and more accurately as B increases (law of large numbers)
- Little problem: we do not have B samples of size n, we have only one!
- There is a get around ... just pull your bootstraps!

Estimating $V(\hat{\mu})$ with the so-called *bootstrap estimator* (Efron, 1979)

- We have a single dataset $(X_1, ..., X_n)$
- We can pretend to have B different samples of size n:
  - sample $Y_1^{(1)}, ..., Y_n^{(1)}$ in $(X_1, ..., X_n)$ uniformly with replacement, compute $\bar{Y}_n^{(1)}$
  - sample $Y_1^{(2)}, ..., Y_n^{(2)}$ in $(X_1, ..., X_n)$ uniformly with replacement, compute $\bar{Y}_n^{(2)}$
  - $\vdots$
  - $\vdots$
  - sample $Y_1^{(B)}, ..., Y_n^{(B)}$ in $(X_1, ..., X_n)$ uniformly with replacement, compute $\bar{Y}_n^{(B)}$

    Note that $Y_1^{(b)}, ..., Y_n^{(b)}$ usually have ties
- Compute $\widehat{V(\bar{X}_n)} = \frac{1}{B} \sum_{j=1}^{B} \left(\bar{Y}_n^{(j)} - 1/B \sum_{j=1}^{B} \bar{Y}_n^{(j)} \right)^2$

R illustration of the bootstrap idea

```
func2 <- function()
  {
    col <- col + 1
    y <- sample(x=x,size=n,replace=TRUE)
    points(y,rep(0,n),col=col,cex=2)
    abline(v=mean(y),col=col,pch=16,cex=4)
    col
  }
col <- 1
plot(x,rep(0,n),type="p",xlab="Data",
     ylab="",xlim=c(-1,4),cex=2)
abline(v=mean(x),col=1,pch=16,cex=4)
col <- func2()
```

Key idea underlying bootsrtap techniques:

The true unknown probability distribution F of X is approximated by the empirical distribution F^*.
(Recall that F^* is the discrete distribution putting an equal weight $1/n$ on each observed data value.)

This approximation makes sense as soon as n (sample size) is large.

Numerical comparison

```
sd <- 1
n <- 200 ; x <- rnorm(n=n,mean=2,sd=sd)
B <- 10000
X.theo <- X.boot <- matrix(nrow=n,ncol=B)
for(b in 1:B)
  {
    X.theo[,b] <- rnorm(n=n,mean=2,sd=1)
    X.boot[,b] <- sample(x=x,size=n,replace=TRUE)
  }
mean.theo <- apply(X=X.theo,MARGIN=2,FUN=mean)
mean.boot <- apply(X=X.boot,MARGIN=2,FUN=mean)
par(mfrow=c(1,2))
hist(mean.theo,prob=TRUE)
lines(density(mean.theo),col=2,lwd=2); lines(density(mean.boot),lwd=2,col=3
hist(mean.boot,prob=TRUE)
lines(density(mean.theo),col=2,lwd=2); lines(density(mean.boot),lwd=2,col=3
var(mean.theo)
var(mean.boot)
sd^2/n # truth
```

Summary

- A collection of several samples can be mimicked by resampling the data
- A theoretical parameter that can be expressed as an expectation can be estimated by an average over the "fake" samples
- The idea is very general (see example for the median below)

Bootstrap-based confidence intervals

Problem: give a C.I. for a parameter θ on the basis of a sample $X_1, ..., X_n$
Solution:

- Build an estimator $\hat{\theta}$
- Assume that $\hat{\theta} \sim \mathcal{N}$
- Estimate $V(\hat{\theta})$ by the bootstrap estimator $\widehat{V(\hat{\theta})}$
- Build the normal-based C.I. = $\left[\hat{\theta} + z_{\alpha/2}\sqrt{\widehat{V(\hat{\theta})}} \; ; \; \hat{\theta} + z_{1-\alpha/2}\sqrt{\widehat{V(\hat{\theta})}}\right]$

Take home messages

1. Stochastic simulations useful in the analyis of stat. models and ordinary numerical analysis
2. Computers can produce numbers that look random
3. Congruence method fast but outcome of varying quality. Default algorithm implemented in R and Matlab highly reliable
4. Lack of data can be supplemented by resampled data which involves (simple) simulations

References

- Bootstrap: Chapter 8 of *All of Statistics*, Larry Wasserman, Springer, 2004. Available on Google books.
- *Introducing Monte Carlo Methods with R*, C.P. Robert & G. Casella Springerlink