Stochastic Simulation

Prof. Carolina Osorio
Dept. of Decision Sciences, HEC Montreal
60615A

Why we need simulation: an example

- Assume we have a large population of urban commuters across a city. Their travel time is integer-valued, independently and uniformly distributed within $[1, 100]$ minutes.
- You are carrying out an experiment where you pick two travelers from the city's population at random. You measure the network performance as the ratio between the travel times of the two people.
- What is the expected network performance?
- What is the intuitive answer?

Why we need simulation: an example

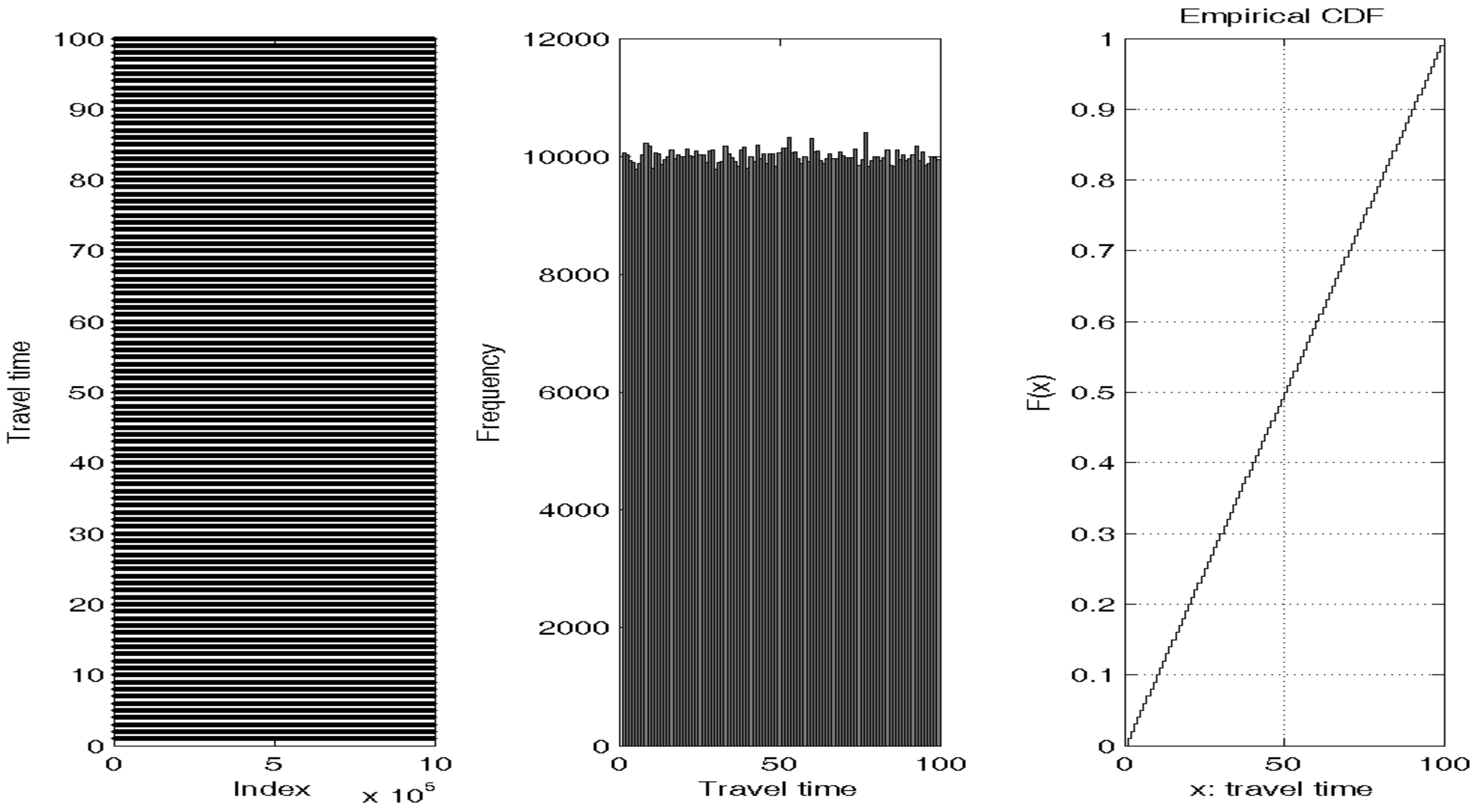
```
populationSize = 1000000;

travelTime1 = zeros(1,populationSize);
travelTime2 = zeros(1,populationSize);
ratio_travelTime = zeros(1,populationSize);

for j = 1:populationSize

   % randi([a,b]):  returns a pseudorandom integer scalar in [a,b]
   travelTime1(j) = randi([1,100]);
   travelTime2(j) = randi([1,100]);
   ratio_travelTime(j) = travelTime1(j)/travelTime2(j);

end;
```

Why we need simulation: an example

- The average ratio_travelTime is significantly larger than 1.
- Simulation statistics:
  - Average travelTime1: 50.489
  - Average travelTime2: 50.483
  - Average ratio_travelTime: 2.61
- Average inputs do not give average outputs
- Hence the need to simulate in order to estimate the average output

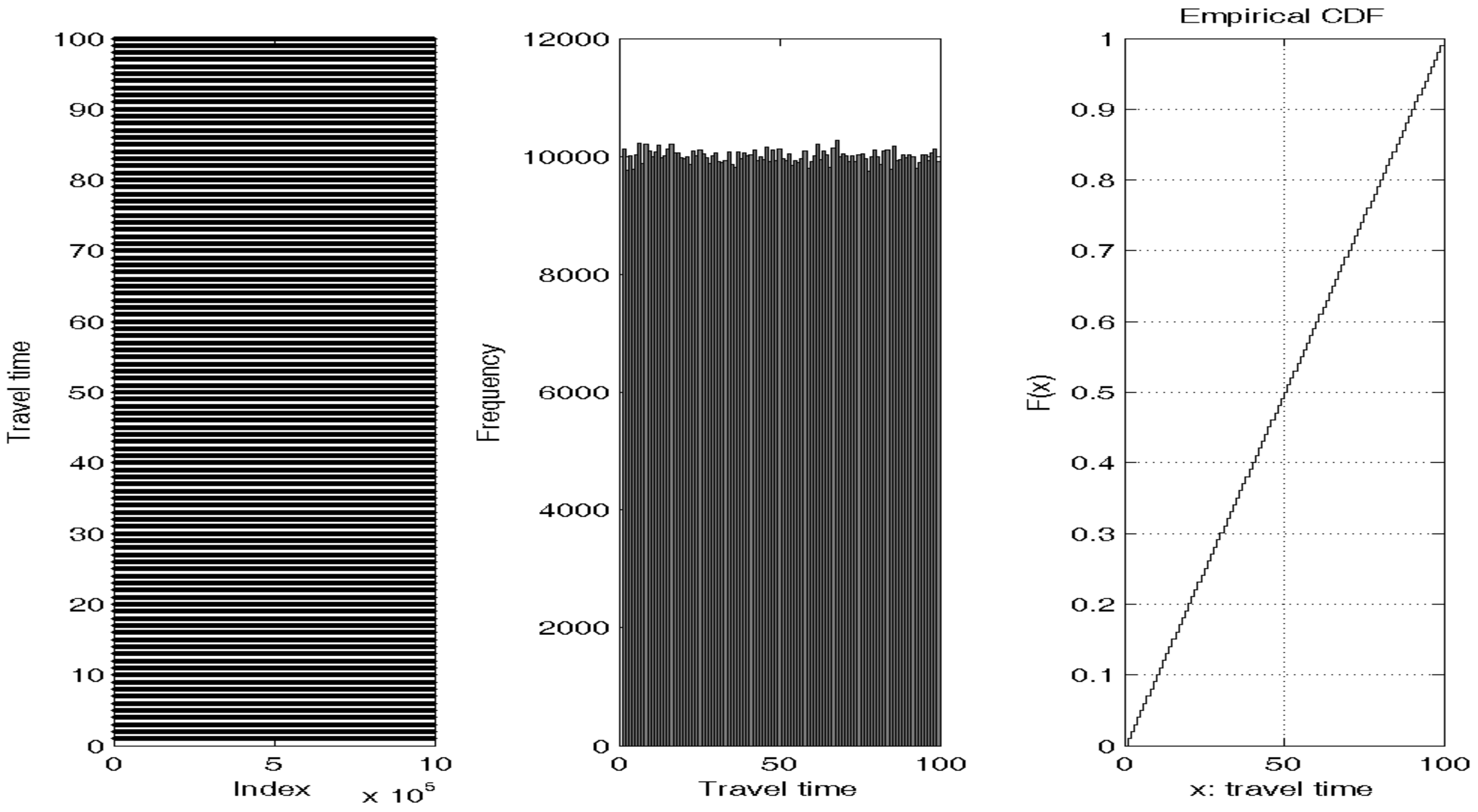
Why we need simulation: an example

- Results for travelTime1



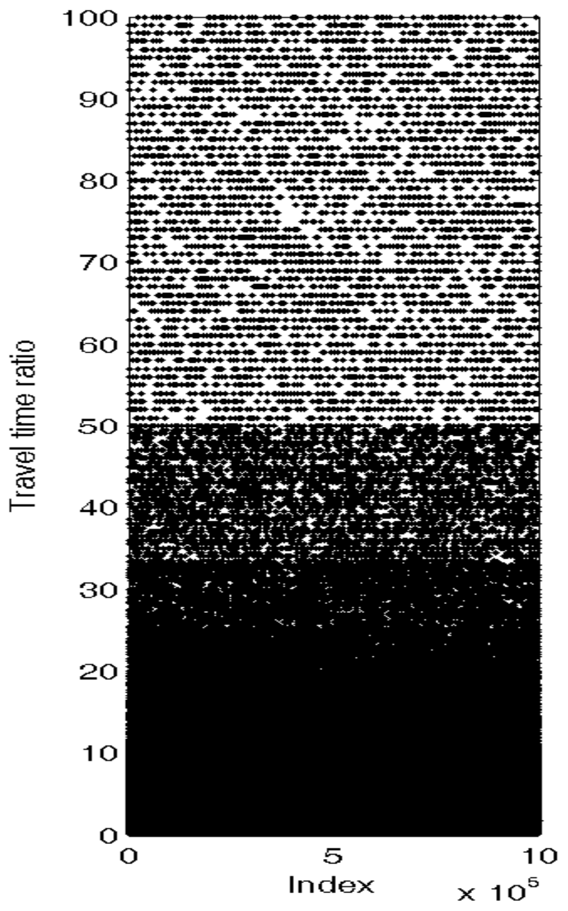
Why we need simulation: an example

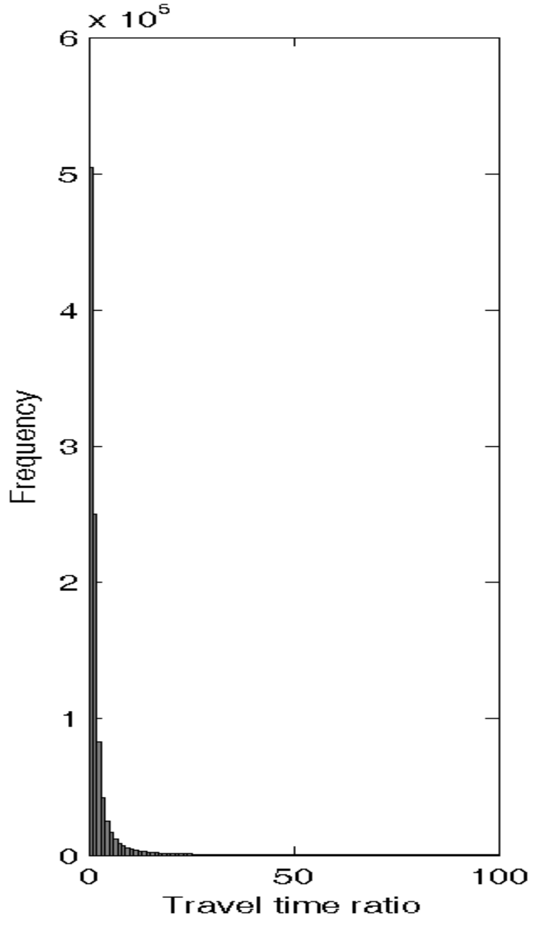
- Results for travelTime2

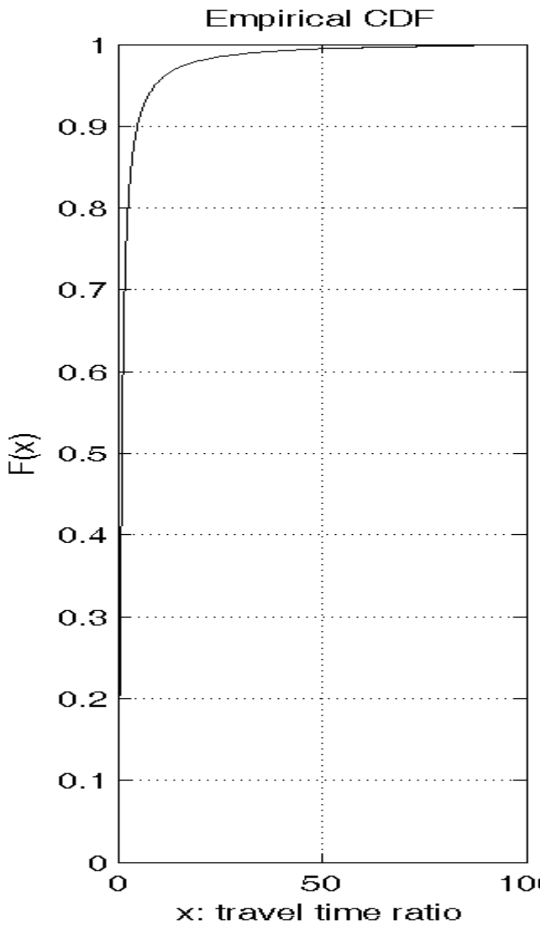


Why we need simulation: an example

- Results for ratio_travelTime

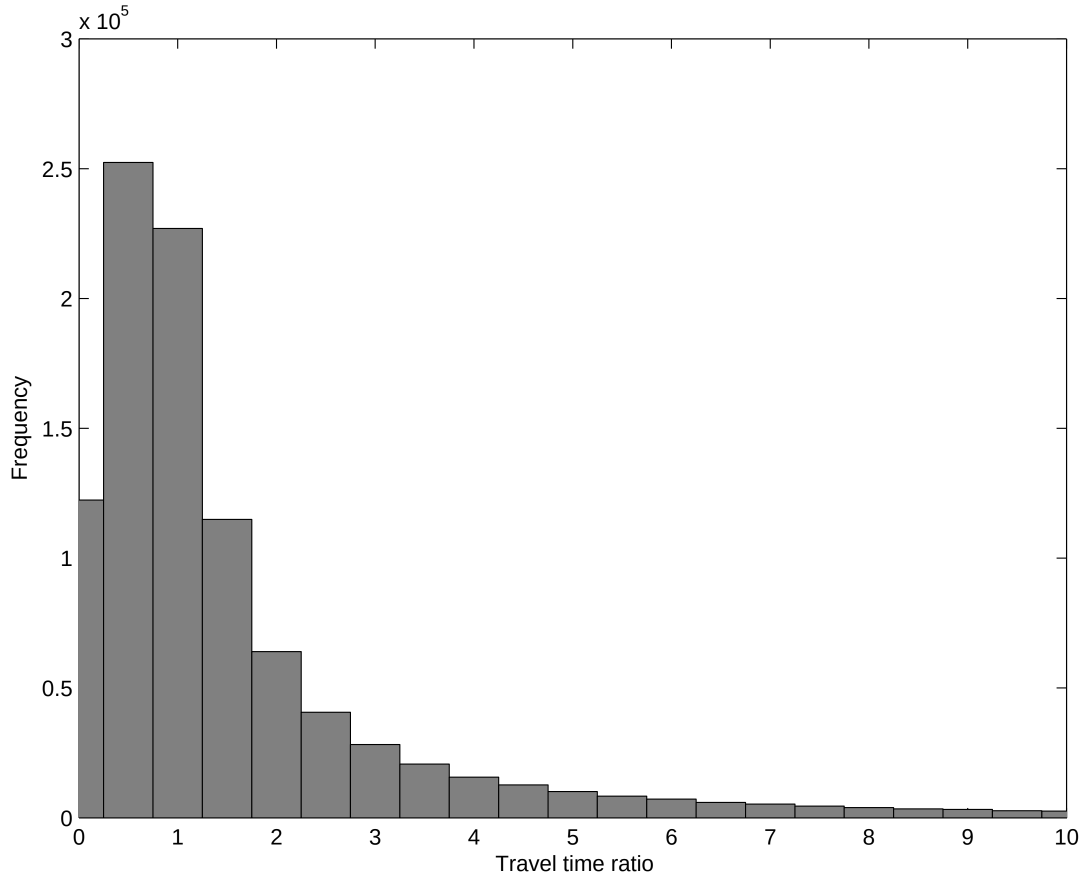






Why we need simulation: an example

- Results for ratio_travelTime



Uniform random numbers

- The sequence of numbers is not random!
- Given the initial number the entire sequence is known
- This is true for all mathematical algorithms for random number generation
- The goal of these algorithms is to generate a sequence of numbers that appear to be taken from the $U(0, 1)$ dbn
- Are also called pseudo-random numbers
- Think of them as a very long list of predetermined numbers

Uniform random numbers

- Pseudorandom number generator: a *deterministic* algorithm for generating a sequence of numbers that approximates the properties of random numbers
- A simple example: the middle-square method, John von Neumann (1949)

```
1. Choose a k-digit integer
2. Square the integer. If the result does not have 2k-digits,
   then add 0's to the left of it to obtain  a 2k-digit result.
3. Extract the k middle digits and return to Step 2.
```

- Numerical example
- The middle-square approach is not a good algorithm:
  - Often has a small period: e.g., initial digit 3567, then the period is only 46.
  - If the middle digits are all 0, then all future numbers are 0.

Uniform random numbers

- Pseudorandom number generator: a *deterministic* algorithm for generating a sequence of numbers that approximates the properties of random numbers

- What is a good random number generator?
  1. a long (maximum) period
  2. covers unit interval uniformly
  3. same for higher dimensional unit-cubes
  4. does not show obvious patterns
  5. fools statistical tests when used in simulations

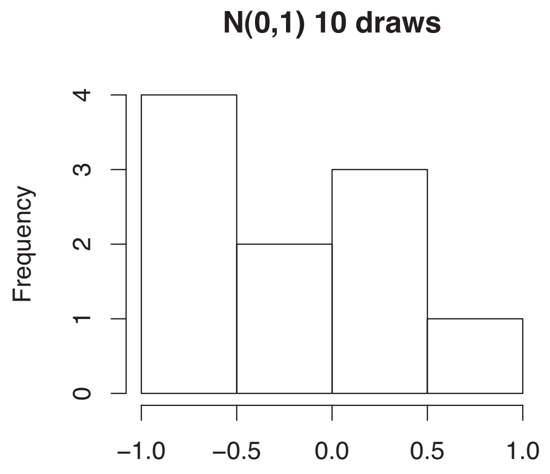
Uniform random numbers

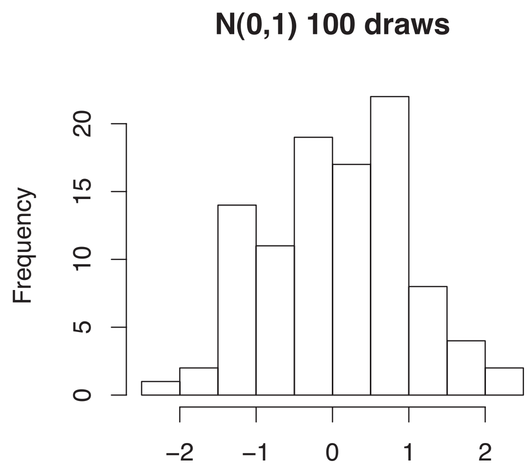
- Most softwares use linear-congruential generators (LCG)
  Recursive algorithm:
  $$r_{n+1} = (ar_n + c) \text{ mod } m$$
  where a, c and m are positive integers ($a < m, c < m$).
  Choose r_0, the seed.

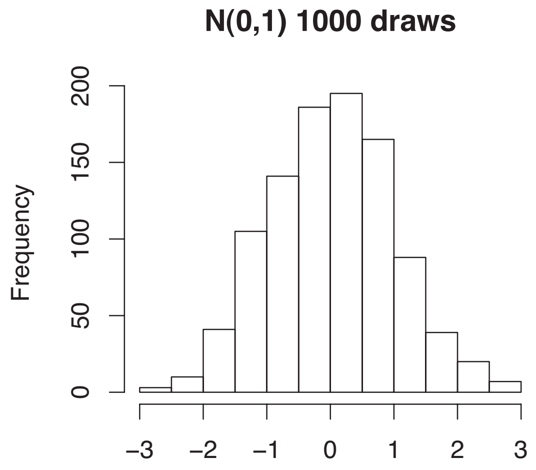
- Choice of a, c, m is not trivial
- Typically: choice such that LCG is of full period m. This can be achieved if:
  1. m and c are relatively prime (i.e., the only positive integer that divides them both is 1)
  2. For any prime number q that divides m, q divides $a - 1$
  3. If 4 divides m, then 4 divides $a - 1$
- Example of commercial software:
  $m = 2^{31}, \quad a = 1103515245, \quad c = 12345$

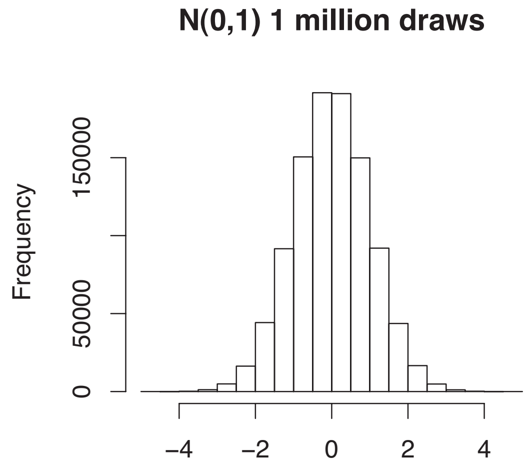
Replications

- The simulation outputs are random variables.
- Running one simulation replication provides one realization of these r.v.









Input uncertainty

- Carefull on overestimating the added value of running a large number of replications !
- Parking lot example: M/M/∞
- Goal: estimate the steady-state expected number of users in the system
- Known input distributions, but unknown parameters λ, μ.
- We have access to m i.i.d observations from the “real-world”. We use them to derive estimates: $\hat{\lambda}, \hat{\mu}$.
- We run n simulation replications and derive the sample mean estimator:

$$\hat{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

- Then:

$$E[\hat{Y} \mid \hat{\lambda}, \hat{\mu}] = \frac{\hat{\lambda}}{\hat{\mu}} \quad Var[\hat{Y} \mid \hat{\lambda}, \hat{\mu}] = \frac{\hat{\lambda}}{n\hat{\mu}}$$

- But $\hat{\lambda} \neq \lambda, \ \hat{\mu} \neq \mu$

Input uncertainty

- Since $\hat{\lambda} \neq \lambda, \ \hat{\mu} \neq \mu$

$$E[\hat{Y}] = \frac{m}{m-1}\frac{\lambda}{\mu}$$

$$Var[\hat{Y}] \approx \frac{\lambda}{n\mu} + \frac{2(\lambda/\mu)^2}{m}$$

- Source: Nelson (2013) section 7.2.1.