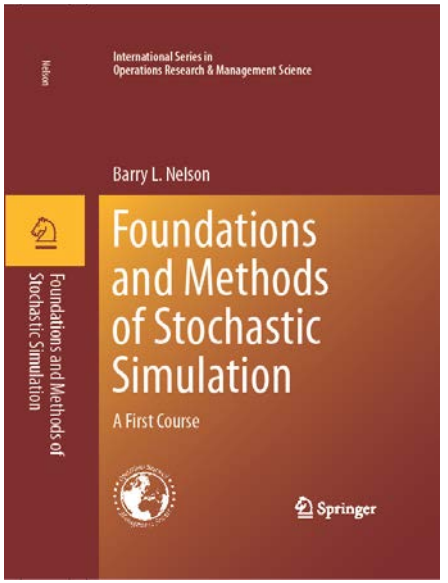


Chapters 1, 2 & 3: A Brief Introduction

©Barry L. Nelson

Northwestern University

July 2017

Why do we simulate?

- We typically choose to simulate a dynamic, stochastic system when the performance measure we want…
  - Is not analytically tractable.
  - Is not computationally tractable.
  - Cannot be approximated with a bound on the error.
- Let's start with a very simple example to remind us what we are doing and why.

Discrete-event simulation example

Two components work as an active and spare, so the system fails if both are failed.

The lifetime of a component is equally likely 1, 2, 3, 4, 5 or 6 days. Repair takes exactly 2.5 days (only one at a time). What can we say about the time to failure (TTF) for this system?

The *state* of the system is the number of functional components.
The *events* are the failure of a component and the completion of a repair.

	System	Event Calendar	
Clock	State	Next Failure	Next Repair
0	2		

	System	Event Calendar	
Clock	State	Next Failure	Next Repair
0	2	$0 + \mathbf{5} = 5$	∞
5	1	$5 + \mathbf{3} = 8$	$5 + 2.5 = 7.5$
7.5	2	8	∞
8	1	$8 + \mathbf{6} = 14$	$8 + 2.5 = 10.5$
10.5	2	14	∞
14	1	$14 + \mathbf{1} = 15$	$14 + 2.5 = 16.5$
15	0	∞	16.5

Features

- Simulated time (the simulation clock) jumps from event time to event time; this is called *next-event time advance*.
- The current state of the system, the event logic, and the list of future events are all we need to advance the system to the next state change.
- The simulation ends when a particular system state occurs. In other simulations termination may occur at a fixed time or event count.
- The system state over time is a *sample path* (output) from which we may extract performance measures. It is one realization of a stochastic process.

Outputs

Replications are statistically independent repetitions of the same model. We distinguish between *within-replication* and *across-replication* output data.

The time of system failure Y and the number of functional components $\{S(t); t \geq 0\}$ are *within-replication* outputs.

The times of system failure $Y_1, Y_2, \ldots, Y_n$, and the average number of functional components, $\bar{S}_1, \bar{S}_2, \ldots, \bar{S}_n$, from n replications are *across-replication* outputs.

Across-replication outputs should be independent (we made independent rolls of the dice) and identically distributed (we all followed the same rules).

Time averages

Notice that $\bar{S}$ is a *time-average* because $S(t)$ is a continuous-time output variable.

$$\bar{S} = \frac{1}{Y}\int_0^Y S(t)\,dt = \frac{1}{e_N - e_0}\sum_{i=1}^{N} S(e_{i-1}) \times (e_i - e_{i-1})$$

where $e_0, e_1, \ldots, e_N$ are the event times in a replication.

$$\begin{aligned}\bar{S} &= \frac{1}{15-0}[2(5-0) + 1(7.5-5) + 2(8-7.5) + 1(10.5-8) \\ &\quad +2(14-10.5) + 1(15-14)] = \frac{24}{15}\end{aligned}$$

Performance measures

We run simulations to *estimate* system performance, often to compare alternative designs for a system.

We justify this based on some version of the *strong law of large numbers*, either as the number of replications increases

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} Y_i = \mu \quad \text{(with probability 1)}$$

...or as the length of the replication goes to infinity (this would make sense if we did not stop the simulation when both components are failed)

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T S(t)\, dt = \theta \quad \text{(with probability 1)}$$

This book

Building Simulations: Modeling and programming

Doing Simulations Well: Experimental design and output analysis

Simulation "Theory:" Why simulations work

Simulation for Research: Get you ready to use simulation in your research, even if your research area is not simulation

Programming: Book uses VBA for Excel, but Java and Matlab versions are available on the book web site.

Structure of a discrete-event simulation

- **Main program**
  - Initialize state variables, clock, statistics
  - Schedule at least one future event
  - Simulation loop: Until ending condition met
    - Call Timer to find next event
    - Call event subprogram
  - Final update of statistics and report results
- **Timer subprogram**
  - Remove next event from event calendar
  - Advance clock
  - Return event type
- **Event subprograms**
  - Update state variables
  - Schedule any future events
  - Update statistics

Programming quick start

```
Dim Clock As Double          ' simulation clock
Dim NextFailure As Double    ' time of next failure event
Dim NextRepair As Double     ' time of next repair event
Dim S As Double              ' system state
Dim Slast As Double          ' previous value of the system state
Dim Tlast As Double          ' time of previous state change
Dim Area As Double           ' area under S(t) curve

Private Function Timer() As String
     Const Infinity = 1000000

' Determine the next event and advance time
    If NextFailure < NextRepair Then
        Timer = "Failure"
        Clock = NextFailure
        NextFailure = Infinity
    Else
        Timer = "Repair"
        Clock = NextRepair
        NextRepair = Infinity
    End If
End Function
```

Global declarations

```
Private Sub TTF()
' Program to generate a sample path for the TTF example
    Dim NextEvent As String
    Const Infinity = 1000000
    Rnd (-1)
    Randomize (1234)

' Initialize the state and statistical variables
    S = 2
    Slast = 2
    Clock = 0
    Tlast = 0
    Area = 0

' Schedule the initial failure event
    NextFailure = WorksheetFunction.Ceiling(6 * Rnd(), 1)
    NextRepair = Infinity

' Advance time and execute events until the system fails
    Do Until S = 0
        NextEvent = Timer
        Select Case NextEvent
        Case "Failure"
            Call Failure
        Case "Repair"
            Call Repair
        End Select
    Loop

' Display output
    MsgBox ("System failure at time " _
            & Clock & " with average # functional components " & Area / Clock)
    End
End Sub
```

```
Private Sub Failure()
' Failure event
' Update state and schedule future events
    S = S - 1
    If S = 1 Then
        NextFailure = Clock + WorksheetFunction.Ceiling(6 * Rnd(), 1)
        NextRepair = Clock + 2.5
    End If

' Update area under the S(t) curve
    Area = Area + Slast * (Clock - Tlast)
    Tlast = Clock
    Slast = S
End Sub

Private Sub Repair()
' Repair event
' Update state and schedule future events
    S = S + 1
    If S = 1 Then
        NextRepair = Clock + 2.5
        NextFailure = Clock + WorksheetFunction.Ceiling(6 * Rnd(), 1)
    End If

' Update area under the S(t) curve
    Area = Area + Slast * (Clock - Tlast)
    Tlast = Clock
    Slast = S
End Sub
```

Important Computer Simulation Concepts

Source of randomness, usually assumed i.i.d. U(0,1)

Input random variables with known distributions that we specify; ex: component TTF is DU(1,2,3,4,5,6)

Output random variables whose properties we want to estimate; ex: system TTF

Random-number generation

Even if we could get random numbers, we don't want them. We prefer that our results be repeatable, but appear to be random: *pseudorandom*.

Think of this as a long list with period P that repeats if we reach the end.

$$U_1, U_2, \ldots, U_P, U_1, U_2, \ldots$$

U's are consumed in order from wherever we start.

With a good pseudorandom number generator it does not matter where in the list we start (usually specified by a "seed" or "stream").

Random-variate generation

We specify an input distribution F_X we want.

Variate generation means finding an algorithm such that if $X = \texttt{algorithm}(U)$ and $U \sim U(0,1)$, then

$$\Pr\{X \le x\} = F_X(x)$$

An algorithm that always works is

$$X = F_X^{-1}(U) = \min\{x : F_X(x) \ge U\}$$

Proof for continuous case:
$\Pr\{X \le x\} = \Pr\{F_X^{-1}(U) \le x\} = \Pr\{U \le F_X(x)\} = F_X(x)$

Replications

We will typically make multiple replications to get better estimators and compute a measure of error (e.g., confidence interval).

Results across replications are (pseudo) i.i.d.

- "independent" because different (pseudo)random numbers are used if we initialize the generator OUTSIDE the replication loop.
- "identically distributed" because initial conditions and logic are always the same INSIDE the loop (e.g, 2 functional components at the beginning).

Thus standard large-sample statistics apply:
$Y_1, Y_2, \ldots, Y_{100}$ system times to failure: use $\bar{Y} \pm 1.96S/\sqrt{100}$

Replication version of TTF

```
Private Sub TTFRep()
' Program to generate a sample path for the TTF example
    Dim NextEvent As String
    Const Infinity = 1000000
    Rnd (-1)
    Randomize (1234)
' Define and initialize replication variables
    Dim Rep As Integer
    Dim SumS As Double, SumY As Double
    SumS = 0
    SumY = 0
    For Rep = 1 To 100
' Initialize the state and statistical variables
        S = 2
        Slast = 2
        Clock = 0
        Tlast = 0
        Area = 0
' Schedule the initial failure event
        NextFailure = WorksheetFunction.Ceiling(6 * Rnd(), 1)
        NextRepair = Infinity
' Advance time and execute events until the system fails
        Do Until S = 0
            NextEvent = Timer
            Select Case NextEvent
                Case "Failure"
                    Call Failure
                Case "Repair"
                    Call Repair
            End Select
        Loop
' Accumulate replication statistics
        SumS = SumS + Area / Clock
        SumY = SumY + Clock
    Next Rep
' Display output
    MsgBox ("Average failure at time " _
            & SumY / 100 & " with average # functional components " & SumS / 100)
    End
End Sub
```

Initialize random number generator

Canonical examples

- To illustrate things as we go along we will use 4 ½ small examples that are
  - Easy to simulate.
  - Don't actually need to be simulated (they are pretty tractable).
- This will allow us to see the issues that arise in general without getting buried in details.
- These examples are also useful for testing new ideas in simulation.

$M(t)/M/\infty$ queue

- A queue with "ample" (infinite) servers. Used to design systems that should have adequate capacity nearly always.
- Arrivals are described by a nonstationary Poisson arrival process with rate $\lambda(t)$.
- Service times are independent and identically distributed exponential with finite mean τ.
- Examples: parking lot for large mall; cell phone system.

More on *M(t)/M/∞*

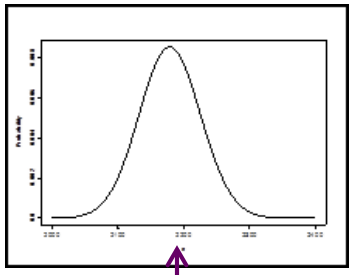
If $N(t)$ is the number of cars in the parking garage at time $t \geq 0$, then $N(t)$ has a Poisson distribution with mean $m(t)$, where $m(t)$ solves

$$\frac{d}{dt}m(t) = \lambda(t) - \frac{m(t)}{\tau}$$

with an appropriate initial condition like $m(0) = 0$ if the garage starts empty.

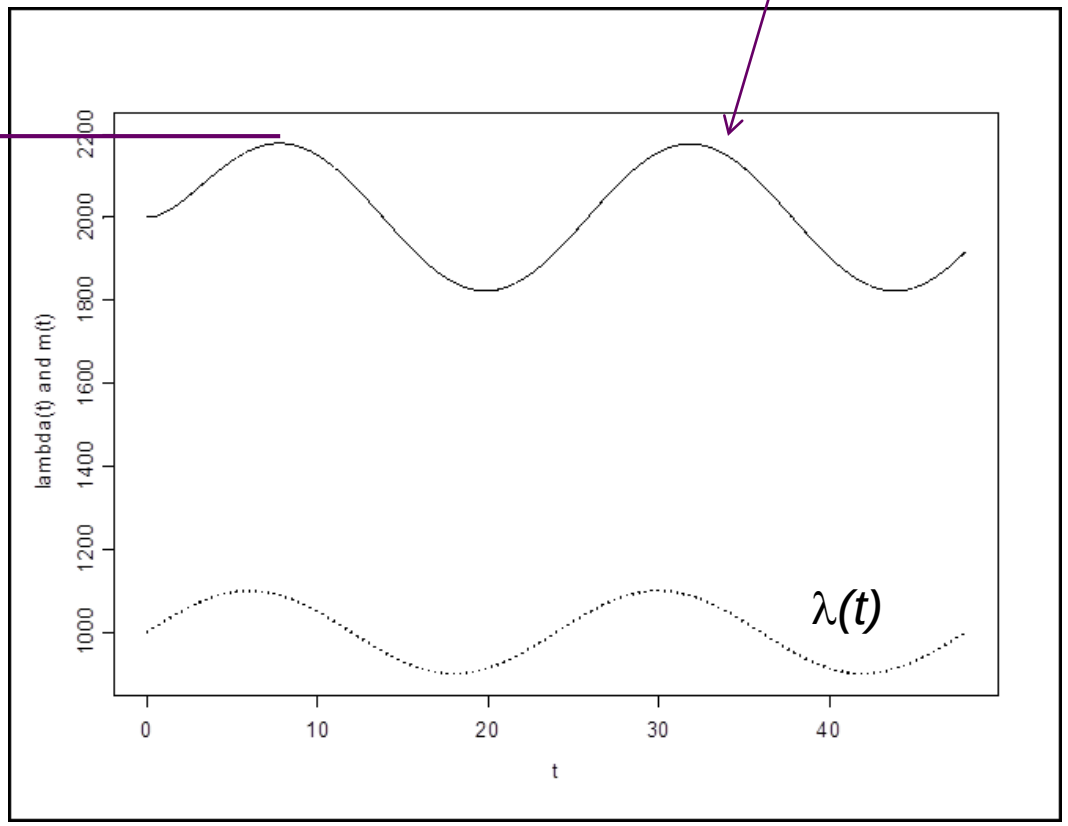
For capacity planning, solve for $m^\star = \max_t m(t)$ and choose c to control

$$\Pr\{\text{number of cars in the garage } \leq c\} = \sum_{n=0}^{c} \frac{\mathrm{e}^{-m^\star}(m^\star)^n}{n!}.$$



Distribution of number in the garage at mean *m**

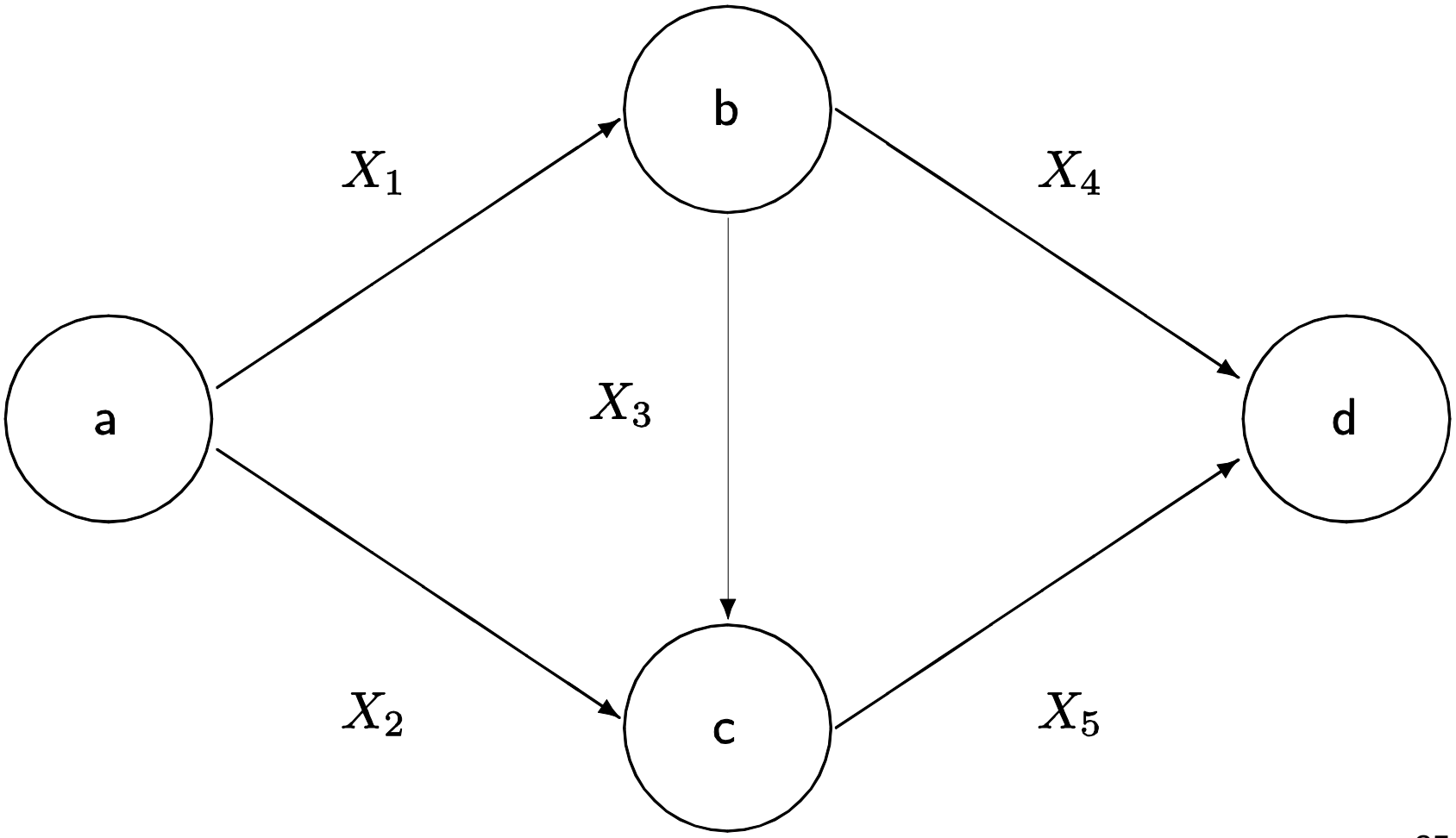
Mean number of cars in the garage *m(t)*



Simulation issues

- How do we simulate a nonstationary arrival process?
- How do we program a simulation that could (conceptually) have nearly an infinite number of "car departure" events?
- What relevant performance measure can we actually get out of a simulation?

Stochastic activity network (SAN)



More on the SAN

Let Y represent the time to complete the project, where X_i are the (random) activity times:

$$Y = \max\{X_1 + X_4, X_1 + X_3 + X_5, X_2 + X_5\}.$$

The project planners are interested in performance measures such as $\theta = \Pr\{Y > t_{\text{promised}}\}$ and $\mu = \mathrm{E}(Y)$.

If all of the activity times are exponential 1 then $\Pr\{Y \leq t_p\} =$

$$\left(\frac{1}{2}t_p^2 - 3t_p - 3\right)\mathrm{e}^{-2t_p} + \left(-\frac{1}{2}t_p^2 - 3t_p + 3\right)\mathrm{e}^{-t_p} + 1 - \mathrm{e}^{-3t_p}.$$

Simulation issues

The SAN illustrates

- Performance measures such as means, probabilities and quantiles (if we want to know how to set a promise date) are all relevant.

  Note that

  $$\theta = \mathrm{E}[I(Y > t_p)]$$

  so a probability is also a mean, but a quantile is different.

- A situation where the natural design is to make replications, so the question is how many to make.

M/G/1 queue

- A single-queue, first-come-first-served service system with one server.
- Arrivals are described by a stationary Poisson arrival process (interarrivals exponential $1/\lambda$)
- Service times are independent and identically distributed with finite mean τ and standard deviation σ.
- Examples: video kiosk, ticket window, ATM machine, receptionist.

M/G/1 & Lindley's equation

$A_1, A_2, \ldots$ i.i.d. exponential interarrival times with mean $1/\lambda$.

$X_1, X_2, \ldots$ i.i.d. service times with mean τ and standard deviation σ.

If $Y_1, Y_2, \ldots$ are the successive waiting times in queue then

$$Y_i = \max\{0, Y_{i-1} + X_{i-1} - A_i\}, \ i = 1, 2, \ldots$$

where $Y_0 = X_0 = 0$.

Notice that we expect the $\{Y_i\}$ to be *neither independent nor identically distributed.*

What is known?

Provided $\lambda < \infty$, $\rho = \lambda\tau < 1$ and $\sigma < \infty$ then

1. The outputs $Y_1, Y_2, \ldots$ converge in distribution to a random variable Y.
2. The sample mean $\bar{Y}(m) = m^{-1} \sum_{i=1}^{m} Y_i$ converges with probability 1 to a constant μ.
3. The $\mathrm{E}(Y)$ and μ are equal, and are given by

$$\mu = \mathrm{E}(Y) = \frac{\lambda(\sigma^2 + \tau^2)}{2(1 - \lambda\tau)}.$$

What if our goal is to estimate μ (or some other property of Y)?

Simulation issues

- The quantities of interest are defined in the limit; easy (sometimes) mathematically; hard (nearly always) in simulation.
- The experiment design is not so clear: One very long run or multiple replications?
- As $\rho \to 1$ both the mean and the variability of *Y* explode (e.g., the standard deviation of *Y* grows as $1/(1-\rho)$).

The AR(1) surrogate model

While a lot is known (at least in the limit) about the $M/G/1$, the $\max$ operator makes transient behavior difficult to derive.

A nice stand-in that shares similar behavior is the AR(1)

$$Y_i = \mu + \varphi(Y_{i-1} - \mu) + X_i, \ i = 1, 2, \ldots$$

with $X_1, X_2, \ldots$ i.i.d. $(0, \sigma^2)$ and $|\varphi| < 1$.

The AR(1) is easy to analyze because

$$Y_i = \mu + \varphi^i(Y_0 - \mu) + \sum_{j=0}^{i-1} \varphi^j X_{i-j}.$$

Asian option

A "call" option is a contract giving the holder the right to purchase a stock for a fixed "strike price" at some time in the future.

If the stock's value increases well above the strike price, then the option is a good deal provided the purchase price for the option was not too high.

If offered a call option with strike price K and maturity T on a stock that is currently trading at $X(0)$, how much should one be willing to pay for this option?

For an Asian option it is

$$\nu = \mathrm{E}\left[\mathrm{e}^{-rT}\left(\bar{X}(T) - K\right)^{+}\right]$$

Simulation issues

Usually $X(t)$ is modeled as GBM, a continuous-time, continuous-state stochastic process, and

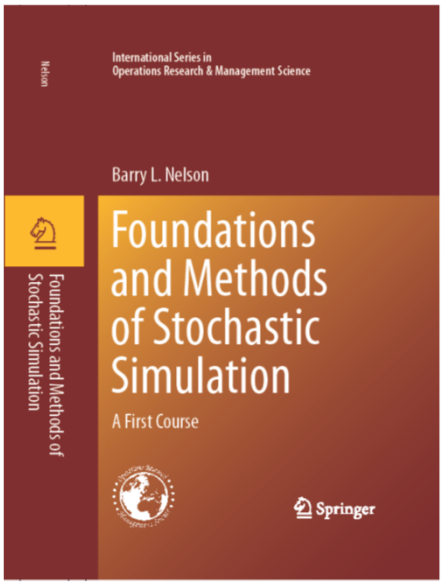
$$\bar{X}(T) = \frac{1}{T}\int_0^T X(t)\,\mathrm{d}t.$$

We cannot directly simulate this.

The natural approximation

$$\widehat{\bar{X}(T)} = \frac{1}{m}\sum_{i=1}^{m} X(i\Delta t)$$

with $\Delta t = T/m$ introduces bias, but we need highly accurate and precise estimates in financial applications.



Chapter 2.3.4 & 4: VBASim

©Barry L. Nelson

Northwestern University

December 2012

Object orientation

- Discrete-event simulations often contain multiple instances of similar objects:
  - Entities: things that come and go, like customers, messages, jobs, events
  - Queues: ordered lists of entities
  - Resources: scarce quantities that entities need, like servers, computers, machines
  - Statistics: information to be recorded on all of the above
- These are more naturally treated as “objects” for which we can have many instances.

VBASim

- A small collection of VBA class modules (objects) that support simulation.
  - You can easily customize and add to these
- A module containing declarations and a few useful subs.
- VBA implementations of the random-number and random-variate generation functions from simlib by Law & Kelton.

VBA class modules

- With VBA class modules we can define the template for an object.
- A Class Module contains…
  - Properties → “attributes” in simulation terminology
  - Methods → instructions about how to do things
- The key benefit of objects is that we can create multiple instances, each uniquely identified and with its own properties and methods.

```
' Generic continuous-time statistics object
' Note that CTStat should be called AFTER the value of the variable changes
Private Area As Double
Private Tlast As Double
Private Xlast As Double
Private TClear As Double

Private Sub Class_Initialize()
' Executes when CTStat object is created to initialize variables
    Area = 0
    Tlast = 0
    TClear = 0
    Xlast = 0
End Sub

Public Sub Record(X As Double)
' Update the CTStat from last time change and keep track of previous value
    Area = Area + Xlast * (Clock - Tlast)
    Tlast = Clock
    Xlast = X
End Sub

Function Mean() As Double
' Return the sample mean up through current time but do not update
    Mean = 0
    If (Clock - TClear) > 0 Then
        Mean = (Area + Xlast * (Clock - Tlast)) / (Clock - TClear)
    End If
End Function

Public Sub Clear()
' Clear statistics
    Area = 0
    Tlast = Clock
    TClear = Clock
End Sub
```

Properties: each CTStat will have its own copy

Automatically called when a New CTStat is created

Called from within your simulation to update the statistic

Called from within your simulation to report the sample mean

Called from within the simulation to reset the CTStat

CTStat object

Anatomy of a class module

- Declarations
  - Defines the attributes (Properties) the object has
  - **Private** if only used within the object; otherwise **Public**
  - Each instance of the object will have its own unique copy

```
' Generic continuous-time statistics object
' Note that CTStat should be called AFTER the value of the variable changes
Private Area As Double
Private Tlast As Double
Private Xlast As Double
Private TClear As Double
```

Anatomy of a class module

- Methods can be Subs, Functions, Property Let and Property Get
- To the best of my ability to tell, Property Get is the same as a Function

```
Dim QueueLength As New CTStat
QueueLength.Record(Q)
Xbar = QueueLength.Mean
```

```
Public Sub Record(X As Double)
' Update the CTStat from last time change and keep track of previous value
    Area = Area + Xlast * (Clock - Tlast)
    Tlast = Clock
    Xlast = X
End Sub

Function Mean() As Double
' Return the sample mean up through current time but do not update
    Mean = 0
    If (Clock - TClear) > 0 Then
        Mean = (Area + Xlast * (Clock - Tlast)) / (Clock - TClear)
    End If
End Function
```

Anatomy of a class module

- There are special methods that will be executed when an object is created or destroyed.
- To release the object name
  Set QueueLength = **Nothing**

```
Private Sub Class_Initialize()
' Executes when CTStat object is created to initialize variables
    Area = 0
    Tlast = 0
    TClear = 0
    Xlast = 0
End Sub


Private Sub Class_Terminate()
' Termination code goes here…
End Sub
```

VBA Collections

- A Collection is a VBA generalization of an array; it can store objects of the same class.
  - VBA itself uses lots of collections
  - Worksheets(“Sheet2”) or Worksheets(2)
- We use these in the event calendar management and queue management.

Collection syntax

`Dim` `Queue` `As New Collection`

`Queue.`**`Item(i)`** ← the object in position i

`Queue.`**`Count`** ← number of objects currently in the Queue collection

`Queue.`**`Remove(j)`** ← remove the object in position j of the Queue collection

`Queue.`**`Add`** `customer,` **`Before:=`**k ← insert object customer before object currently in position k (also has option **after**)

VBASim EventCalendar Class Module

```
' This class module creates an Event Calendar object

Private ThisCalendar As New Collection

Public Sub Schedule(addedEvent As EventNotice)
' Add EventNotice in EventTime order
    Dim i As Integer
    If ThisCalendar.Count = 0 Then 'no events in calendar
        ThisCalendar.Add addedEvent
    ElseIf ThisCalendar(ThisCalendar.Count).EventTime <= addedEvent.EventTime Then
                                   'added event after last event in calendar
        ThisCalendar.Add addedEvent, After:=ThisCalendar.Count
    Else                           'search for the correct place to insert the event
        For i = 1 To ThisCalendar.Count
            If ThisCalendar(i).EventTime > addedEvent.EventTime Then
                Exit For
            End If
        Next i
        ThisCalendar.Add addedEvent, before:=i
    End If
End Sub

Public Function Remove() As EventNotice
' Remove next event and return the EventNotice object
    If ThisCalendar.Count > 0 Then
        Set Remove = ThisCalendar.Item(1)
        ThisCalendar.Remove (1)
    End If
End Function

Function N() As Integer
' Return current number of events on the event calendar
    N = ThisCalendar.Count
End Function
```

Note: EventNotice is also a Class Module with two attributes: EventTime and EventType

M/G/1 Queue in VBASim

```
' Example illustrating use of VBASim for
' simulation of M/G/1 Queue.

' See VBASim module for generic declarations
' See Class Modules for the supporting VBASim classes

' Parameters we may want to change
Public MeanTBA As Double        ' mean time between arrivals
Public MeanST As Double         ' mean service time
Public Phases As Integer        ' number of phases in service distribution
Public RunLength As Double      ' run length
Public WarmUp As Double         ' "warm-up" time

' Global objects needed for simulation
' These will usually be queues and statistics

Dim Queue As New FIFOQueue   'customer queue
Dim Wait As New DTStat       'discrete-time statistics on customer waiting
Dim Server As New Resource   'server resource
```

```
Public Sub MG1()
    Dim Reps As Integer
    Dim NextEvent As EventNotice

    Call MyInit ' special initializations for this simulation
    For Reps = 1 To 10
        Call VBASimInit 'initialize VBASim for each replication
        Call Schedule("Arrival", Expon(MeanTBA, 1))
        Call Schedule("EndSimulation", RunLength)
        Call Schedule("ClearIt", WarmUp)
        Do
            Set NextEvent = Calendar.Remove
            Clock = NextEvent.EventTime
            Select Case NextEvent.EventType
            Case "Arrival"
                Call Arrival
            Case "EndOfService"
                Call EndOfService
            Case "ClearIt"
                Call ClearStats
            End Select
        Loop Until NextEvent.EventType = "EndSimulation"

' Write output report for each replication
        Call Report(Wait.Mean, "MG1", Reps + 1, 1)
        Call Report(Queue.Mean, "MG1", Reps + 1, 2)
        Call Report(Queue.NumQueue, "MG1", Reps + 1, 3)
        Call Report(Server.Mean, "MG1", Reps + 1, 4)
    Next Reps
    End ' ends execution, closes files, etc.
End Sub
```

```
Public Sub MyInit()

' Initialize the simulation
    Call InitializeRNSeed
    Server.SetUnits (1) ' set the number of servers to 1
    MeanTBA = 1
    MeanST = 0.8
    Phases = 3
    RunLength = 55000
    WarmUp = 5000

' Add queues, resources and statistics that need to be
' initialized between replications to the global collections

    TheDTStats.Add Wait
    TheQueues.Add Queue
    TheResources.Add Server

' Write headings for the output reports

    Call Report("Average Wait", "MG1", 1, 1)
    Call Report("Average Number in Queue", "MG1", 1, 2)
    Call Report("Number Remaining in Queue", "MG1", 1, 3)
    Call Report("Server Utilization", "MG1", 1, 4)

End Sub
```

VBASim will reinitialize any objects in these collections between replications; there is also a The CTStats collection.

```
Public Sub Arrival()
' Arrival event

' Schedule next arrival
    Call Schedule("Arrival", Expon(MeanTBA, 1))

' Process the newly arriving customer

    Dim Customer As New Entity
    Queue.Add Customer
    Set Customer = Nothing
' If server is not busy, start service by seizing the server

    If Server.Busy = 0 Then
        Server.Seize (1)
        Call Schedule("EndOfService", Erlang(Phases, MeanST, 2))
    End If

End Sub
```

Note that we dim a NEW Entity; "New" means not only declare, but also create

Seize is VBASim for "make busy this many units of the resource"

Without “New” this is only a declaration

```
Public Sub EndOfService()
' End of service event

' Remove departing customer from queue and record wait time

    Dim DepartingCustomer As Entity
    Set DepartingCustomer = Queue.Remove
    Wait.Record (Clock - DepartingCustomer.CreateTime)
    Set DepartingCustomer = Nothing    'be sure to free up memory

' Check to see if there is another customer; if yes start service
' otherwise free the server

    If Queue.NumQueue > 0 Then
        Call Schedule("EndOfService", Erlang(Phases, MeanST, 2))
    Else
        Server.Free (1)
    End If

End Sub
```

How did this get set?

Free is VBASim for “make idle this many units of the resource”

Using VBASim

- VBASim Module
  - Declarations
  - Subs: VBASimInit, Schedule, SchedulePlus, Report
- VBASim Class Modules
  - CTStat, DTStat
  - Entity
  - EventCalendar, EventNotice
  - FIFOQueue
  - Resource
- Changing and adding to VBASim

VBASim module: Delcarations

```
Public Clock As Double                      'simulation global clock
Public Calendar As New EventCalendar        'event calendar

' Set up Collections to be reinitialized between replications
Public TheCTStats As New Collection   ' continuous-time statistics
Public TheDTStats As New Collection   ' discrete-time statistics
Public TheQueues As New Collection    ' queues
Public TheResources As New Collection ' resources
```

- Everything in VBASim is "Public" so that it can be used from any module in the Workbook.
- The […] are collections of VBASim objects that will be reinitialized whenever `VBASimInit` is called.

VBASimInit

- Usage: `Call VBASimInit`
- Typically placed **inside** the replication loop
- Resets the `Clock`, `Calendar`, and all of `The[…]` collections

Schedule, SchedulePlus & Report

```
Public Sub Schedule(EventType As String, EventTime As Double)

Public Sub SchedulePlus(EventType As String, EventTime As
Double, TheObject as Object)

Public Sub Report(Output As Variant, WhichSheet As String, Row
As Integer, Column As Integer)
```

- Usage:

```
Call Schedule("Arrival", Expon(2,1))

Dim Customer as New Entity
Call SchedulePlus("Arrival", Expon(2,1), Customer)

Call Report(Queue.Mean, "Sheet2", 3, 5)
```

- Notice that EventTime is how far into the future the event is to occur, not the absolute time.
- The "Plus" version allows another object (usually an Entity) to be attached to the EventNotice.

Entity class module

- Usage

```
Dim Customer as New Entity

Delay = Clock – Customer.CreateTime
```

- You can add as many additional attributes as you need the entities to have to the Entity Class Module.

```
' This is a generic entity that has a single attribute CreateTime

Public CreateTime As Double

' Add additional problem specific attributes here


Private Sub Class_Initialize()
' Executes when Entity object is created to initialize variables
        CreateTime = Clock
End Sub
```

EventNotice class module

- Usage

```
Dim NextEvent as EventNotice

Set NextEvent = Calendar.Remove

Clock = NextEvent.EventTime
Select Case NextEvent.EventType

     Call EOS(NextEvent.WhichObject)
```

- The EventNotices are usually created by Schedule or SchedulePlus; you use them when advancing to the next event.

```
' This is a generic EventNotice object with EventTime, EventType
‘ and WhichObject attributes

Public EventTime As Double
Public EventType As String
Public WhichObject As Object

' Add additional problem specific attributes here
```

About the other class modules

- You are unlikely to modify the other class modules (although you may create your own variations using them as templates).
- The most important thing is to know how to use them.
- Remember:
  When you `Dim X as New Object`, a **pointer** is created to that (perhaps very complex) object. That pointer needs to be retained, either in a specific name (e.g., `TicketQueueStatistic`) or stored in a collection or else the object is lost.

CTStat

- Collects continuous-time statistics
- Methods: Record, Mean and Clear
- Usage

```
Dim TotalCustomerStats as New CTStat

TotalCustomerStats.Record(NumCust)

Call Report(TotalCustomerStats.Mean,“Sheet1”, 1,2)

TotalCustomerStats.Clear
```

Call AFTER the value has changed

DTStat

- Collects discrete-time statistics
- Methods: Record, Mean, StdDev, N and Clear
- Usage

```
Dim Wait as New DTStat

Wait.Record(Clock - Customer.CreateTime)

Call Report(Wait.Mean,“Sheet1”, 1,2)
Call Report(Wait.StdDev,“Sheet1”, 1,3)
Call Report(Wait.N, “Sheet1”, 1,4)

Wait.Clear
```

Resource

- Models resources and also keeps a CTStat on average number in use
- Properties: Busy [current number in use]
- Methods: SetUnits, Seize, Free, Mean
- Usage

```
Dim Server as New Resource

Server.SetUnits(5)        ‘ resource has capacity 5
Server.Seize(1)           ‘ make busy 1 unit of resource
Server.Free(1)            ‘ make idle 1 unit of resource

If Server.Busy = 5 Then …

Call Report(Server.Mean, “Sheet1”, 5,4)
```

FIFOQueue

- Models first-in-first-out queue, and also keeps a CTStat on average number in queue
- Methods: NumQueue, Add, Remove, Mean
- Usage

```
Dim Line as New FIFOQueue

Dim Shopper as New Entity
Line.Add Shopper

Dim DepartingShopper as Entity
Set DepartingShopper = Line.Remove

If Line.NumQueue = 0 Then…

Call Report(Line.Mean, “Sheet1”, 5,10)
```

Some notes…

- The CTStat's created by `FIFOQueue` and `Resource` are automatically added to `TheCTStats` collection, so they are reinitialized by `VBASimInit`.
- The most common change you will make is to add attributes to the Entity class.
- VBASim currently does not have a lot of error checking.

A note on creating new objects

- Consider the following code

```
Dim Queue as New FIFOQueue
Dim Customer as New Entity
Customer.SomeAttribute = 10
Queue.Add Customer
Dim Customer as New Entity
Customer.SomeAttribute = 11
Queue.Add Customer
```

- Surprisingly, this code puts 2 of the same entity (both with `SomeAttribute = 11`) in the Queue.
- This is because `Dim…New` only creates a new object if the target pointer variable (Customer here) is currently unassigned.

- Correct approach:

```
Dim Queue as New FIFOQueue
Dim Customer as New Entity
Customer.SomeAttribute = 10
Queue.Add Customer
Set Customer = Nothing
Dim Customer as New Entity
Customer.SomeAttribute = 11
Queue.Add Customer
Set Customer = Nothing
```

- Note that the `Customer` is not lost, because it has been placed in the `Queue` (a collection); that is, its reference is being maintained in another way.
- When `Dim…New` encounters an unassigned pointer variable it creates a new object.

Using RNG

- `Call InitializeRNSeed()`
  - Call **once** at the beginning of the simulation to initialize the pseudorandom-number generator
- `lcgrand(Stream)`
  - Pseudorandom-number generator
  - Streams 1-100
- `Expon, Erlang, Random_integer, Normal, Lognormal, Triangular`
  - Arguments are distribution parameters first, with last argument being the stream number 1-100
  - Ex: `Expon(15.2, 7)`

M/G/5 Queue

- What would we have to change to make this a single waiting line, but multiple server queue?
- Let's modify the M/G/1 code…

Changes in Sub MG1

```
Do
        Set NextEvent = Calendar.Remove
        Clock = NextEvent.EventTime
        Select Case NextEvent.EventType
        Case "Arrival"
            Call Arrival
        Case "EndOfService"
            Call EndOfService(NextEvent.WhichObject)
        Case "ClearIt"
            Call ClearStats
        End Select
Loop Until NextEvent.EventType = "EndSimulation"
```

The key difference is that the Queue will now only contain the customers waiting for service, but not those in service. The ones in service will be passed along with the Event Notice.

Changes in Sub Arrival

```
Sub Arrival()
' Arrival event
' Schedule next arrival
    Call Schedule("Arrival", Expon(MeanTBA, 1))
' Process the newly arriving customer
    Dim Customer As New Entity
' If server is not busy, start service by seizing the server
    If Server.Busy < NumServers Then
        Server.Seize (1)
        Call SchedulePlus("EndOfService", _
                    Erlang(Phases, MeanST, 2), Customer)
    Else
        Queue.Add Customer
    End If
    Set Customer = Nothing
End Sub
```

Changes in Sub EndOfService

```
Sub EndOfService(DepartingCustomer As Entity)
' End of service event
' record wait time of departing customer
    Wait.Record (Clock - DepartingCustomer.CreateTime)
    Set DepartingCustomer = Nothing
' Check to see if there is another customer;
' if yes start service otherwise free the server
    If Queue.NumQueue > 0 Then
        Dim NextCustomer As Entity
        Set NextCustomer = Queue.Remove
        Call SchedulePlus("EndOfService", _
              Erlang(Phases, MeanST, 2), NextCustomer)
        Set NextCustomer = Nothing
    Else
        Server.Free (1)
    End If

End Sub
```

Changes in Sub MyInit

```
NumServers = 5
Server.SetUnits (NumServers) ' set the number of servers
```

Written in this way, the simulation can look at any number of servers simply by changing one line of code.

Fax Center Simulation

```
' Parameters we may want to change

Dim MeanRegular As Double       ' mean entry time regular faxes
Dim VarRegular As Double        ' variance entry time regular faxes
Dim MeanSpecial As Double       ' mean entry time special faxes
Dim VarSpecial As Double        ' variance entry time special faxes
Dim RunLength As Double         ' length of the working day
Dim NumAgents As Integer        ' number of regular agents
Dim NumSpecialists As Integer   ' number of special agents
Dim NumAgentsPM As Integer      ' number of regular agents after noon
Dim NumSpecialistsPM As Integer ' number of special agents after noon


' Global objects needed for simulation

Dim RegularQ As New FIFOQueue       ' queue for all faxes
Dim SpecialQ As New FIFOQueue       ' queue for special faxes
Dim RegularWait As New DTStat       ' discrete-time statistics on fax waiting
Dim SpecialWait As New DTStat       ' discrete-time statistics on special fax waiting
Dim Regular10 As New DTStat         ' discrete-time statistics on < 10 minutes threshold
Dim Special10 As New DTStat         ' discrete-time statistics on < 10 minutes threshold
Dim Agents As New Resource          ' entry agents resource
Dim Specialists As New Resource     ' specialists resource
Dim ARate(1 To 8) As Double         ' arrival rates
Dim MaxRate As Double               ' maximum arrival rate
Dim Period As Double                ' period for which arrival rate stays constant
Dim NPeriods As Integer             ' number of periods in a "day"
```

```
Public Sub FaxCenterSim()
    Dim Reps As Integer
    Dim NextEvent As EventNotice
    ' Read in staffing policy
    NumAgents = Worksheets("Fax").Cells(25, 5)
    NumAgentsPM = Worksheets("Fax").Cells(25, 6)
    NumSpecialists = Worksheets("Fax").Cells(26, 5)
    NumSpecialistsPM = Worksheets("Fax").Cells(26, 6)
    Call MyInit

    For Reps = 1 To 10
        Call VBASimInit
        Agents.SetUnits (NumAgents)
        Specialists.SetUnits (NumSpecialists)
        Call Schedule("Arrival", NSPP_Fax(ARate, MaxRate, NPeriods, Period, 1))
        Call Schedule("ChangeStaff", 4 * 60)
        Do
            Set NextEvent = Calendar.Remove
            Clock = NextEvent.EventTime
            Select Case NextEvent.EventType
            Case "Arrival"
                Call Arrival
            Case "EndOfEntry"
                Call EndOfEntry(NextEvent.WhichObject)
            Case "EndOfEntrySpecial"
                Call EndOfEntrySpecial(NextEvent.WhichObject)
            Case "ChangeStaff"
                Agents.SetUnits (NumAgentsPM)
                Specialists.SetUnits (NumSpecialistsPM)
            End Select
        Loop Until Calendar.N = 0 ' stop when event calendar empty
```

```
 ' Write output report for each replication

         Call Report(RegularWait.Mean, "Fax", Reps + 1, 1)
         Call Report(RegularQ.Mean, "Fax", Reps + 1, 2)
         Call Report(Agents.Mean, "Fax", Reps + 1, 3)
         Call Report(SpecialWait.Mean, "Fax", Reps + 1, 4)
         Call Report(SpecialQ.Mean, "Fax", Reps + 1, 5)
         Call Report(Specialists.Mean, "Fax", Reps + 1, 6)
         Call Report(Regular10.Mean, "Fax", Reps + 1, 7)
         Call Report(Special10.Mean, "Fax", Reps + 1, 8)
         Call Report(Clock, "Fax", Reps + 1, 9)
     Next Reps
     End
 End Sub
```

```
Private Sub Arrival()

' Schedule next fax arrival if < 4 PM
    If Clock < RunLength Then
        Call Schedule("Arrival", NSPP_Fax(ARate, MaxRate, NPeriods, Period, 1))
    Else
        Exit Sub
    End If
' Process the newly arriving Fax
    Dim Fax As New Entity
    If Agents.Busy < Agents.NumberOfUnits Then
        Agents.Seize (1)
        Call SchedulePlus("EndOfEntry", Normal(MeanRegular, VarRegular, 2), Fax)
    Else
        RegularQ.Add Fax
    End If
    Set Fax = Nothing
End Sub
```

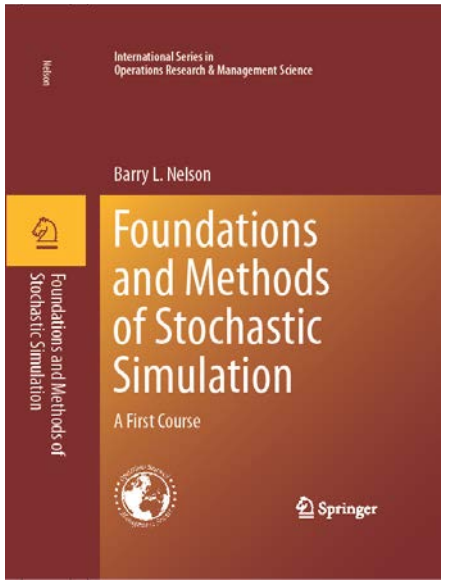
```
Private Sub EndOfEntry(DepartingFax As Entity)
    Dim Wait As Double

' Record wait time if regular; move on if special

    If Uniform(0, 1, 3) < 0.2 Then
        Call SpecialArrival(DepartingFax)
    Else
        Wait = Clock - DepartingFax.CreateTime
        RegularWait.Record (Wait)
        If Wait < 10 Then
            Regular10.Record (1)
        Else
            Regular10.Record (0)
        End If
    End If
    Set DepartingFax = Nothing

' Check to see if there is another Fax; if yes start entry
' otherwise free the agent

    If RegularQ.NumQueue > 0 And Agents.NumberOfUnits >= Agents.Busy Then
        Set DepartingFax = RegularQ.Remove
        Call SchedulePlus("EndOfEntry", Normal(MeanRegular, VarRegular, 2), DepartingFax)
        Set DepartingFax = Nothing
    Else
        Agents.Free (1)
    End If

End Sub
```

```
Private Sub SpecialArrival(SpecialFax As Entity)

' If special agent available, start entry by seizing the special agent

    If Specialists.Busy < Specialists.NumberOfUnits Then
        Specialists.Seize (1)
        Call SchedulePlus("EndOfEntrySpecial", Normal(MeanSpecial, VarSpecial, 4), SpecialFax)
    Else
        SpecialQ.Add SpecialFax
    End If
    Set SpecialFax = Nothing

End Sub
```

```
Private Sub EndOfEntrySpecial(DepartingFax As Entity)
    Dim Wait As Double

' Record wait time and indicator if < 10 minutes

    Wait = Clock - DepartingFax.CreateTime
    SpecialWait.Record (Wait)
    If Wait < 10 Then
        Special10.Record (1)
    Else
        Special10.Record (0)
    End If
    Set DepartingFax = Nothing

' Check to see if there is another Fax; if yes start entry
' otherwise free the specialist

    If SpecialQ.NumQueue > 0 And Specialists.NumberOfUnits >= Specialists.Busy Then
        Set DepartingFax = SpecialQ.Remove
        Call SchedulePlus("EndOfEntrySpecial", Normal(MeanSpecial, VarSpecial, 4), DepartingFax)
        Set DepartingFax = Nothing
    Else
        Specialists.Free (1)
    End If

End Sub
```

```
Private Sub MyInit()
' Initialize the simulation
    Call InitializeRNSeed
    MeanRegular = 2.5
    VarRegular = 1#
    MeanSpecial = 4
    VarSpecial = 1#
    RunLength = 480
' Add queues, resources and statistics that need to be
' initialized between replications to the global collections
    TheDTStats.Add RegularWait
    TheDTStats.Add SpecialWait
    TheDTStats.Add Regular10
    TheDTStats.Add Special10
    TheQueues.Add RegularQ
    TheQueues.Add SpecialQ
    TheResources.Add Agents
    TheResources.Add Specialists
' Write headings for the output reports
    Call Report("Ave Reg Wait", "Fax", 1, 1)
    Call Report("Ave Num Reg Q", "Fax", 1, 2)
    Call Report("Agents Busy", "Fax", 1, 3)
    Call Report("Ave Spec Wait", "Fax", 1, 4)
    Call Report("Ave Num Spec Q", "Fax", 1, 5)
    Call Report("Specialists Busy", "Fax", 1, 6)
    Call Report("Reg < 10", "Fax", 1, 7)
    Call Report("Spec < 10", "Fax", 1, 8)
    Call Report("End Time", "Fax", 1, 9)
' Arrival process data
    NPeriods = 8
    Period = 60
    MaxRate = 6.24
    ARate(1) = 4.37
    ARate(2) = 6.24
    ARate(3) = 5.29
    ARate(4) = 2.97
    ARate(5) = 2.03
    ARate(6) = 2.79
    ARate(7) = 2.36
    ARate(8) = 1.04
End Sub
```

```
Private Function NSPP_Fax(ARate() As Double, MaxRate As Double, NPeriods As Integer, _
                Period As Double, Stream As Integer) As Double
' This function generates interarrival times from a NSPP with piecewise constant
' arrival rate over a fixed time of Period*NPeriod time units

' ARate = array of arrival rates over a common length Period
' MaxRate = maximum value of ARate
' Period = time units between (possible) changes in arrival rate
' NPeriods = number of time periods in ARate

Dim i As Integer
Dim PossibleArrival As Double

PossibleArrival = Clock + Expon(1 / MaxRate, Stream)
i = WorksheetFunction.Min(NPeriods, WorksheetFunction.Ceiling(PossibleArrival / Period, 1))

Do Until Uniform(0, 1, Stream) < ARate(i) / MaxRate
    PossibleArrival = PossibleArrival + Expon(1 / MaxRate, Stream)
    i = WorksheetFunction.Min(NPeriods, WorksheetFunction.Ceiling(PossibleArrival / Period, 1))
Loop

NSPP_Fax = PossibleArrival - Clock

End Function
```



Chapter 5.2: Simulation as a Stochastic Process

©Barry L. Nelson

Northwestern University

July 2017

Simulated asymptotics

- Good news: Simulation problems can often be treated with "large sample" statistical reasoning because data are cheap and we want high precision.
- Bad news: We don't always have the "i.i.d." output data to which our standard asymptotics apply.
- Here we review/introduce some important ideas that run throughout the remainder of the book.

Thought experiment…

- Suppose each of us is simulating the *M/G/1* queue for one replication and reporting the average waiting time.
- We each independently and randomly select our starting random number seed.
  - Assume a *perfect* generator with infinite seeds.
- We report our averages at four different numbers of customers: m = 10, 100, 1000 and 10,000.
- If we collected the results from all of us (plus enough other people to total 1000), what would we see?

Specifics

We each independently simulate

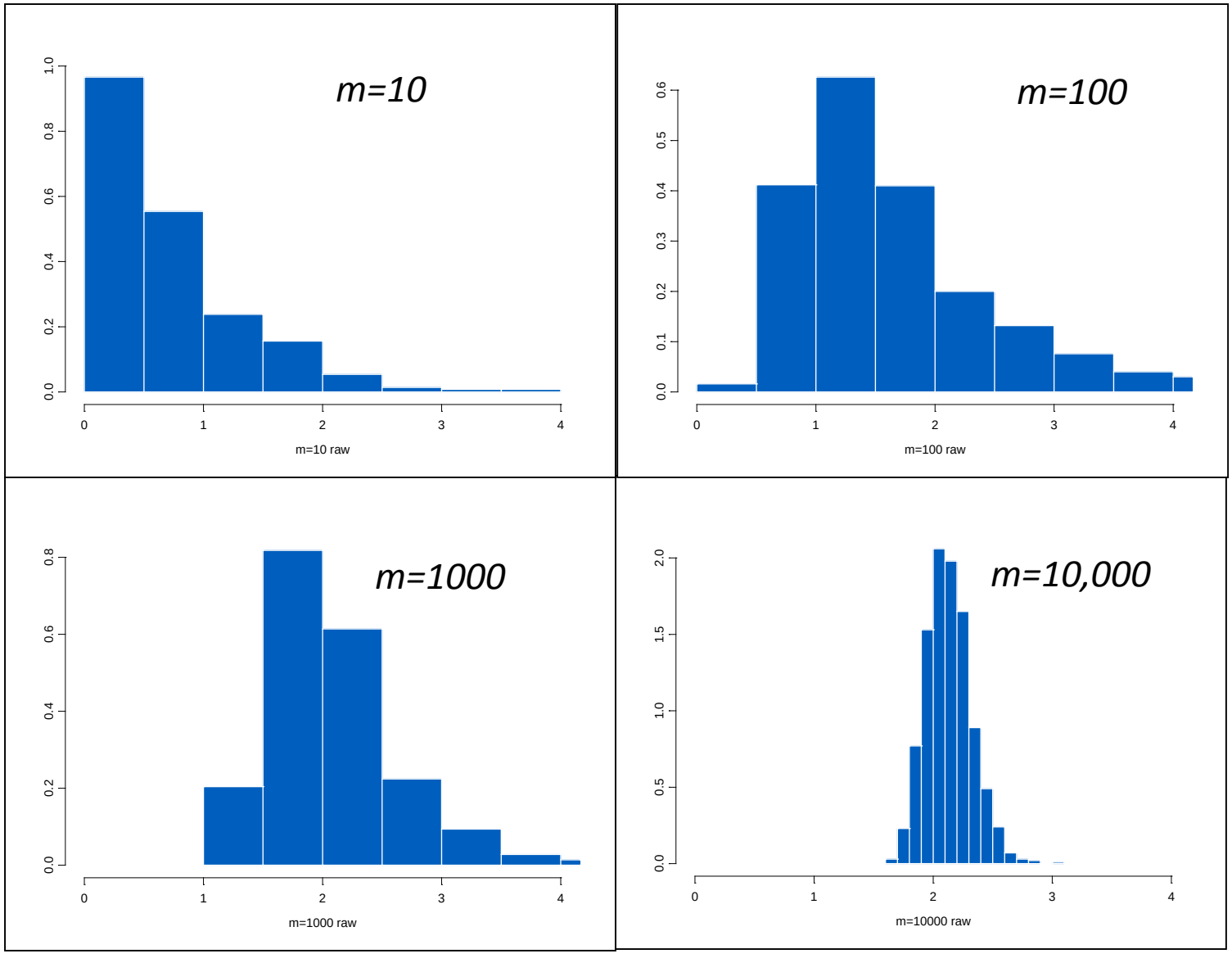
$$Y_i = \max\{0, Y_{i-1} + X_{i-1} - A_i\}$$

with $Y_0 = X_0 = 0$, and we report

$$\bar{Y}(m) = \frac{1}{m} \sum_{i=1}^{m} Y_i$$

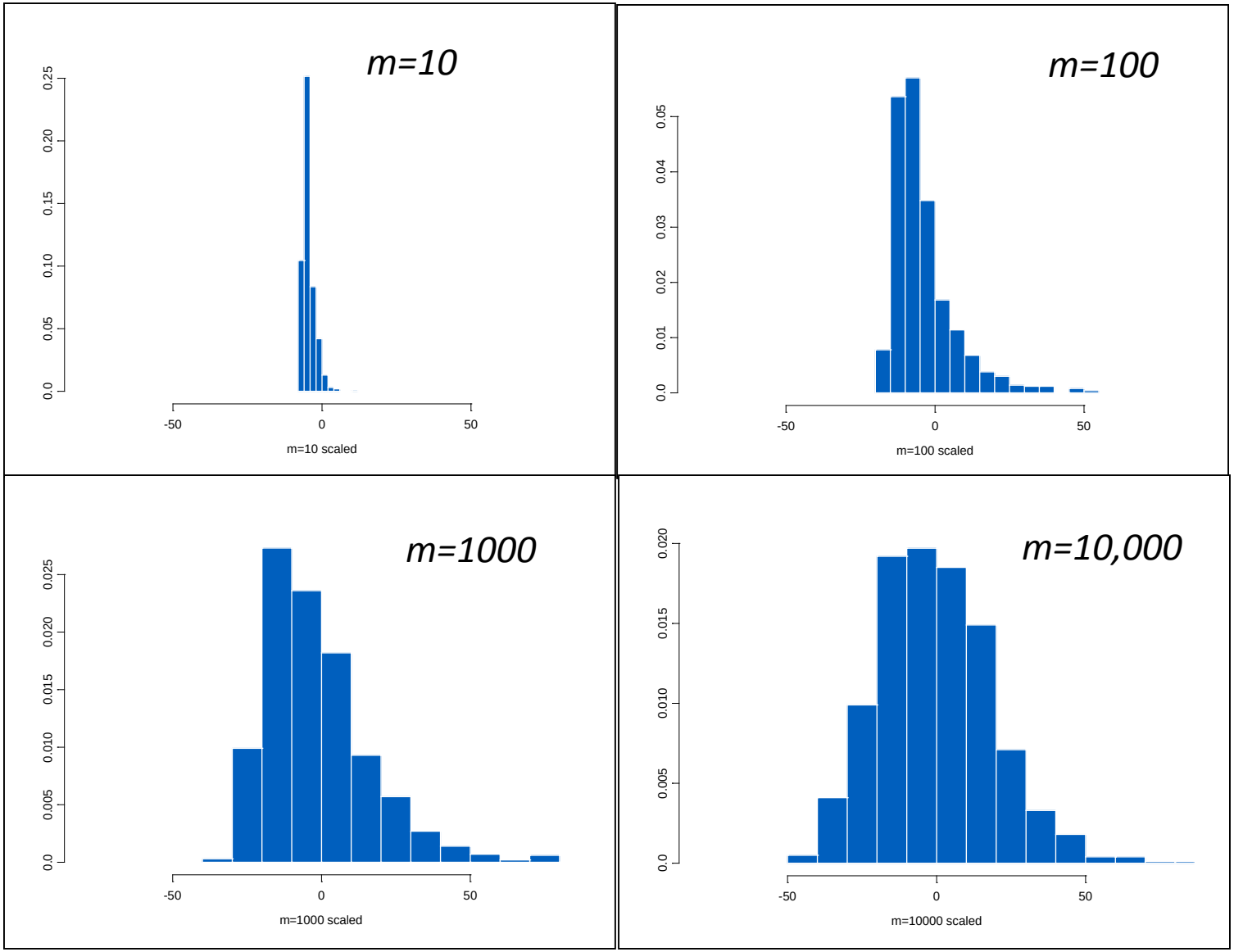
at $m = 10, 100, 1000$ and $10{,}000$.

The parameters I have chosen imply that the steady-state mean is $\mu = 2.133$.



Notice that the 1000 averages are looking more and more normally distributed, but also converging to a single value.

Here we are plotting 1000 values of $\sqrt{m}\left(\overline{Y}(m) - 2.133\right)$



Notice that the scaled and centered averages are looking more and more like a mean 0 stable normal distribution, not converging to a single value.

Summaries of the 1000 averages

1000 averages

0.301432	1.316943	2.181791	2.12834	
0.951291	1.442282	1.60648	2.197148	
0.020483	0.612363	1.68078	2.326293	
0.978482	1.62803	1.738985	2.430449	
1.405882	1.406136	1.362369	2.087307	
0.289488	3.895345	2.148725	2.234362	
0.845869	0.841387	2.062226	2.049821	
0.212686	0.544943	2.001026	2.172981	
0.107063	2.15477	1.470807	2.067342	
1.392107	2.882288	1.67284	2.204039	
1.258184	2.791777	1.684099	1.959779	
1.078646	4.330491	2.746	2.065909	
0.55582	1.02201	1.957363	2.099637	
0.28467	5.880592	2.241638	2.280259	
0.717877	1.738929	2.082655	2.129813	**average**
-1.41512	-0.39407	-0.05034	-0.00319	**bias = average - 2.133**
0.914	0.742	0.359	0.01	**probability > 0.5 off**
m=10	**m=100**	**m=1000**	**m=10000**	

Convergence

- This example illustrates various modes of convergence:
  - Convergence in distribution
    - Scaled average converges to a stable normal distribution.
  - Convergence in probability
    - Chance the average is far away from the mean goes to 0.
  - Convergence with probability 1 (“almost sure convergence”)
    - For "almost all" random number seeds the average converges to the mean.
- But your data were NOT i.i.d.
- We need to understand these modes and when we can expect them to hold in our simulations.

Familiar stuff: Across-rep asymptotics

Strong law of large numbers: If $Z_1, Z_2, \ldots$ are i.i.d. with $\mathrm{E}(Z_1) < \infty$, then

$$\frac{1}{n}\sum_{i=1}^{n} Z_i \xrightarrow{a.s.} \mathrm{E}(Z_1).$$

Central limit theorem: If $Z_1, Z_2, \ldots$ are i.i.d. with $\mathrm{E}(Z_1^2) < \infty$, then

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} Z_i - \mathrm{E}(Z_1)\right) \xrightarrow{D} \gamma\,\mathrm{N}(0,1)$$

where $\gamma^2 = \mathrm{Var}(Z_1)$.

Is E(Z_1) what we want?

SAN: $Z_1 = I(Y_1 > t_p)$ when we want to estimate $\Pr\{Y > t_p\}$? OK

M/G/1:

$$Z_1 = \frac{1}{m} \sum_{j=1}^{m} Y_{1j}$$

when we want to estimate the steady-state mean μ? SLLN and CLT still apply across reps, but convergence is to the wrong value.

This leads to thinking about within-replication asymptotics, as suggested by our group experiment.

Within-replication asymptotics

When might we expect a SLLN or CLT to apply within a single (long) replication? Roughly,...

- Neither the logic nor the input processes change over time, and no periodic (cyclic) behavior.
- The state space of the simulation does not decompose into distinct subsets.
- The state of the simulation in the distant future is effectively independent of the state in the past.

Examples: Stable Markovian queues; irreducible, aperiodic, positive recurrent Markov chains; time series processes; and regenerative processes.

Regenerative processes & steady state

Let Y_t be the output process ($t = 0, 1, 2, \ldots$ or $t \geq 0$).

Suppose in the same simulation we can identify a renewal process $\{S_i, i = 0, 1, 2, \ldots\}$:

$$\begin{aligned} S_0 &= 0 \\ S_i &= A_1 + A_2 + \cdots + A_i \end{aligned}$$

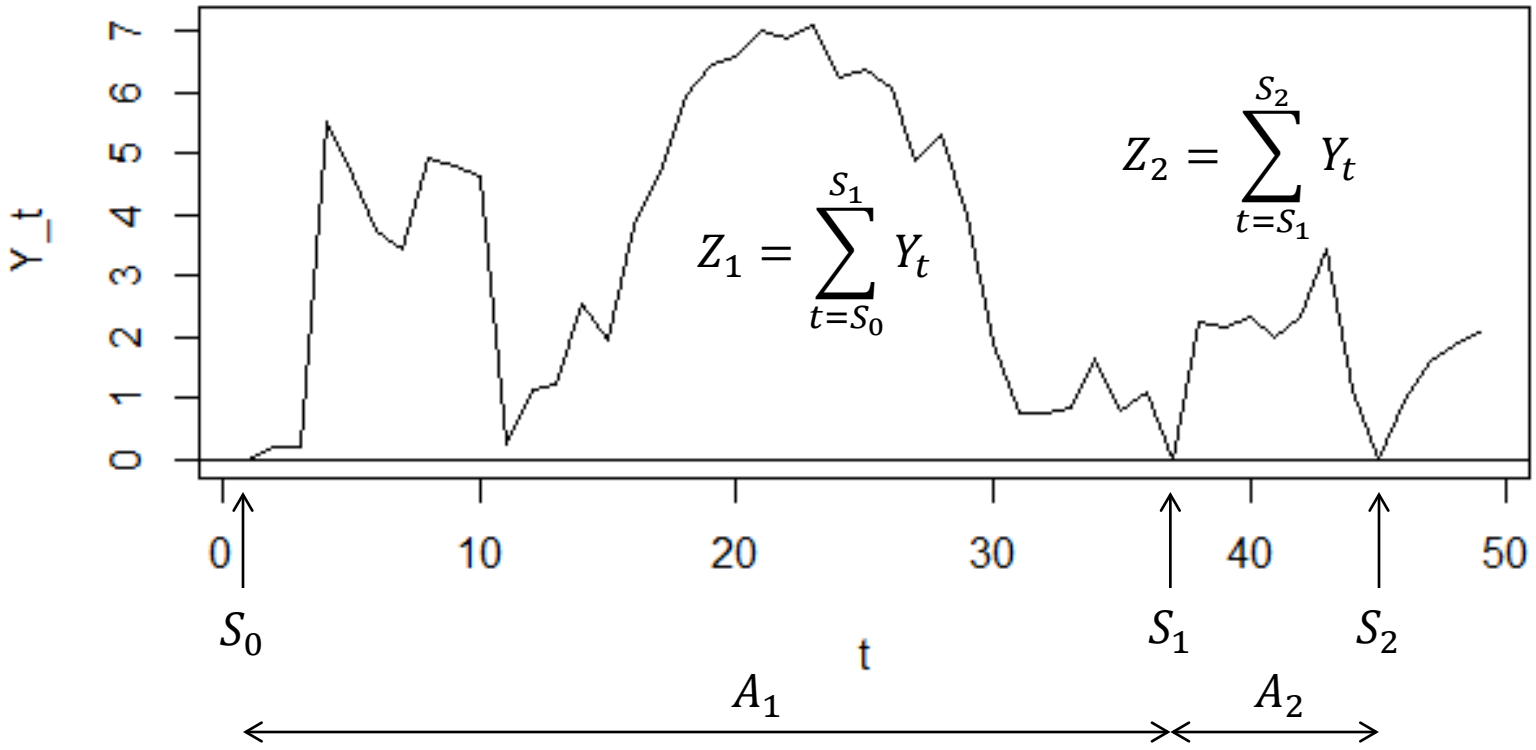
where A_i are i.i.d. with $\Pr\{A_i = 0\} < 1$ and $\Pr\{A_i < \infty\} = 1$.

The output $\{Y_t\}$ is a *regenerative process* if

$$\{Y_t; S_i \leq t < S_{i+1}\}, i = 0, 1, 2, \ldots$$

are i.i.d.

Regeneration in GI/G/1 queue



Let the cumulative output during the ith cycle be

$$Z_i = \int_{S_{i-1}}^{S_i} Y_t \, dt$$

The integral becomes a sum for discrete-time output processes.

If Y_t is regenerative, $\mathrm{E}(|Z_1|) < \infty$, $\mathrm{E}(A_1) < \infty$ (and A_1 is aperiodic for discrete-time processes) then

- $Y_t \xrightarrow{D} Y$ as $t \to \infty$
- $\lim_{t\to\infty} t^{-1} \int_0^t Y_s \, ds \xrightarrow{a.s.} \mathrm{E}(Z_1)/\mathrm{E}(A_1)$
- $\mu = \mathrm{E}(Y) = \mathrm{E}(Z_1)/\mathrm{E}(A_1)$

Regenerative structure establishes that there is a steady-state, and the sample mean satisfies a SLLN for the steady-state mean.

Also a CLT?

Let $V_i = Z_i - \mu A_i$. Notice that the V_i are i.i.d.

If it is also the case that $\mathrm{E}(V_1^2) < \infty$, then

- $\lim_{t\to\infty} \sqrt{t}(\bar{Y}_t - \mu) \xrightarrow{D} \gamma\, \mathrm{N}(0,1)$, where $\bar{Y}_t = t^{-1} \int_0^t Y_s\, ds$
- The asymptotic variance $\gamma^2 = \mathrm{E}(V_1^2)/\mathrm{E}(A_1)$.

Therefore, the sample mean satisfies a CLT, and the asymptotic variance γ^2 (more on this later) can be expressed in terms of properties of a regenerative cycle.

Practical value

Establishing the existence of steady state using the regenerative argument means identifying the renewal process and arguing that $\{Y_t; S_i \leq t < S_{i+1}\}$ are i.i.d.

Typically $S_1, S_2, \ldots$ are times such that the state of the simulation is identical, and the probability distributions of all pending events are also the same.

Also need to establish that $\Pr\{A_i < \infty\} = 1$.

Example: For the $M/G/1$ queue let S_i be the ith time an arriving customer finds the system completely empty. Does this work?

Properties of steady-state estimators: Bias

Even though Y_i has a steady state, the estimator $\bar{Y}(m)$ may (usually is) biased. Define the *asymptotic bias*

$$\begin{aligned}
\beta &= \lim_{m\to\infty} m\left(\mathrm{E}\left(\bar{Y}(m)\right) - \mu\right) \\
&= \lim_{m\to\infty} m\left(\mathrm{E}\left(\frac{1}{m}\sum_{i=1}^{m} Y_i\right) - \mu\right) \\
&= \sum_{i=1}^{\infty}\left(\mathrm{E}(Y_i) - \mu\right)
\end{aligned}$$

For the bias in $\bar{Y}(m)$ to disappear, the bias of Y_i must fall off fast enough so that $\beta < \infty$.

Thus, for large m, $\mathrm{Bias}\left(\bar{Y}(m)\right) \approx \beta/m$.

Properties of steady-state estimators: Variance

For any random variables $Y_1, Y_2, \ldots, Y_m$,

$$\mathrm{Var}\left(\bar{Y}(m)\right) = \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \mathrm{Cov}(Y_i, Y_j).$$

What we hope in steady-state simulation is that the dependence of $Y_1, Y_2, \ldots$ also stabilizes for large m.

The standard assumption is that beyond some point the process is *covariance stationary*, meaning $\sigma^2 = \mathrm{Var}(Y_m)$ and $\rho_k = \mathrm{Corr}(Y_m, Y_{m+k})$ are no longer a function of m.

$$\mathrm{Var}\left(\bar{Y}(m)\right) = \frac{\sigma^2}{m}\left(1 + 2\sum_{k=1}^{m-1}\left(1 - \frac{k}{m}\right)\rho_k\right).$$

Asymptotic variance & MSE

For a covariance stationary process the *asymptotic variance* is

$$\gamma^2 = \lim_{m\to\infty} m\mathrm{Var}\left(\bar{Y}(m)\right) = \lim_{m\to\infty} \sigma^2 \left(1 + 2\sum_{k=1}^{m-1}\left(1 - \frac{k}{m}\right)\rho_k\right)$$

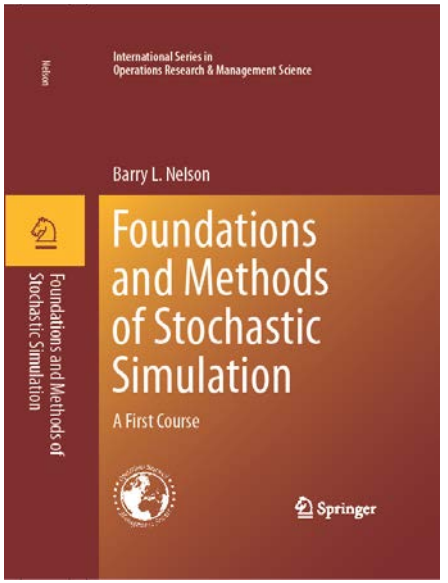
$$= \sigma^2\left(1 + 2\sum_{k=1}^{\infty}\rho_k\right).$$

For the $\mathrm{Var}\left(\bar{Y}(m)\right)$ to go to 0 we need $\gamma^2 < \infty$.

A combined measure that treats bias and variance equally is the *mean squared error*

$$\mathrm{MSE}\left(\bar{Y}(m)\right) = \mathrm{Bias}^2\left(\bar{Y}(m)\right) + \mathrm{Var}\left(\bar{Y}(m)\right)$$

$$\approx \frac{\beta^2}{m^2} + \frac{\gamma^2}{m}.$$



Chapter 6: Simulation Input

©Barry L. Nelson

Northwestern University

July 2017

Simulation input

Input modeling: Selecting (and perhaps fitting) the probability models that represent the uncertainty.

- Inference approach
- Matching approach
- Special topic: Nonstationary arrival processes

Random-variate generation: Representing the inputs as transformations of i.i.d. $U(0,1)$ random variables.

Inversion, rejection and particular properties.

Random-number generation: Producing a good approximation to realizations of i.i.d. $U(0,1)$ random variates.

Combined generators for exceptionally long period.

An input modeling story

Recall the hospital kiosk which we modeled/simulated as a single-server queue. There are two input processes:

Patient arrival process: The hospital has records from which we can extract arrival counts over time intervals (why might they not have actual arrival *times*?).

Process physics suggests a Poisson arrival process.

Service time at the kiosk: The vendor has conducted a trial time study that provides data.

With no obvious process physics, we use distribution fitting software that combines MLEs with some measure of "best fit."

What can go wrong?

Poisson arrival process

The MLE for the arrival rate λ is $\widehat{\lambda} = N(T)/T$ assuming a *stationary* Poisson process. How could we detect nonstationarity?

Suppose we believe daily stationarity and do a goodness-of-fit test for being Poisson. Why might it reject?

1. The distribution of the data is substantially different from Poisson; e.g., includes scheduled arrivals.
2. The arrival rate might differ by day of the week, even if stationary Poisson each day.
3. We have so much data that the test has power against even meaningless deviations from Poisson.

Service process

"Best fit" usually means fit a large collection of distributions using MLEs and choose the one with the best summary statistic (e.g.,smallest test score or largest p-value).

Concerns?

- Are the data actually relevant? Similar to real patients? Will there be learning? Will there be different types of patients by time of day?
- There is no statistical meaning to selecting the distribution having the smallest test score from among an arbitrary collection. This is a heuristic.

View through the queue

Recall that for the $M/G/1$ queue

$$\mathrm{E}(Y) = \frac{\lambda(\sigma^2 + \tau^2)}{2(1 - \lambda\tau)}$$

- Notice that the service-time distribution does not matter *as long as it gets the mean and variance right*. Fortunately this is often true.
- Although the service-time distribution is not critical, the arrival process being Poisson is. So we want good process physics to force a distribution.
- The parameter estimates will almost certainly be wrong: $\widehat{\lambda} \neq \lambda$, $\widehat{\tau} \neq \tau$, and $\widehat{\sigma}^2 \neq \sigma^2$. In fact, we could have $\widehat{\lambda}\widehat{\tau} > 1$.

Univariate input models

$X_1, X_2, \ldots, X_m$ i.i.d real-world input data with a stable (unchanging) distribution F_X.

$F(x; \boldsymbol{\theta})$ is a parametric family with parameter vector $\boldsymbol{\theta}$ to model F_X; e.g., $\boldsymbol{\theta} = (\mu, \sigma^2)$.

"Fitting:" Use some estimator $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\theta}}(X_1, X_2, \ldots, X_m)$.

$\widehat{X} \sim F(x; \widehat{\boldsymbol{\theta}})$, a simulated value from the fitted distribution (what we actually use in the simulation).

Inference vs. Matching

Inference: Treats input modeling as statistical inference about the true distribution F_X, and employs methods that have good statistical properties.

MLEs for $\boldsymbol{\theta}$ and testing H_0: $F(x; \widehat{\boldsymbol{\theta}}) = F_X$ arise here.

Matching: Get properties of $\widehat{X}$ to closely match those of $X_1, X_2, \ldots, X_m$; e.g.,

$$\mathrm{E}\left(\widehat{X} \,|X_1, X_2, \ldots, X_m\right) = \bar{X}.$$

Notice the expectation is with respect to $F(x; \widehat{\boldsymbol{\theta}})$.

Depending on the application, other properties will be important.

Inference approach

The inference paradigm is justified by having strong process physics that supports the choice of parametric family.

Many (perhaps most) probability distributions are derived from some basic process physics (e.g., summing, multiplication, minimum) leading to the distribution.

These are beautifully captured in a series of books by Johnson, Kotz and coauthors.

See also `www.math.wm.edu/~leemis/chart/UDR/UDR.html`

Example: Weibull vs. gamma

Both are used in reliability. Both can have increasing, decreasing or constant hazard: $h(t) = f_T(t)/(1 - F_T(t))$.

So how could we choose?

Look at the tails of the density functions with common scale parameter $\beta = 1$:

Weibull	gamma
$\alpha t^{\alpha-1} \mathrm{e}^{-t^\alpha}$	$\Gamma(\alpha)^{-1} t^{\alpha-1} \mathrm{e}^{-t}$

Gamma eventually has exponential tail (constant failure rate), but Weibull will not unless $\alpha = 1$ which makes it exponential *everywhere*.

Matching approach

The basic idea is to choose parameters for $F(x; \boldsymbol{\theta})$ to match sample properties of the real-world data $X_1, X_2, \ldots, X_m$.

The most common choice is to match central *moments*:

$\mu_X = \mathrm{E}(X)$	mean
$\sigma_X^2 = \mathrm{E}[(X - \mu_X)^2]$	variance
$\alpha_3 = \mathrm{E}[(X - \mu_X)^3]/\sigma_X^3$	skewness
$\alpha_4 = \mathrm{E}[(X - \mu_X)^4]/\sigma_X^4$	kurtosis

Symmetric distributions have $\alpha_3 = 0$.

The normal distribution has kurtosis $\alpha_4 = 3$.

$\alpha_4 - 3$ is called the excess kurtosis.

If $\alpha_3 < \infty$ and $\alpha_4 < \infty$ then $\alpha_4 > 1 + \alpha_3^2$.

Where the action is

Suppose that the first four central moments of X, $(\mu_X, \sigma_X^2, \alpha_3, \alpha_4)$, exist. Let

$$X' = \mu + \sigma \left(\frac{X - \mu_X}{\sigma_X} \right).$$

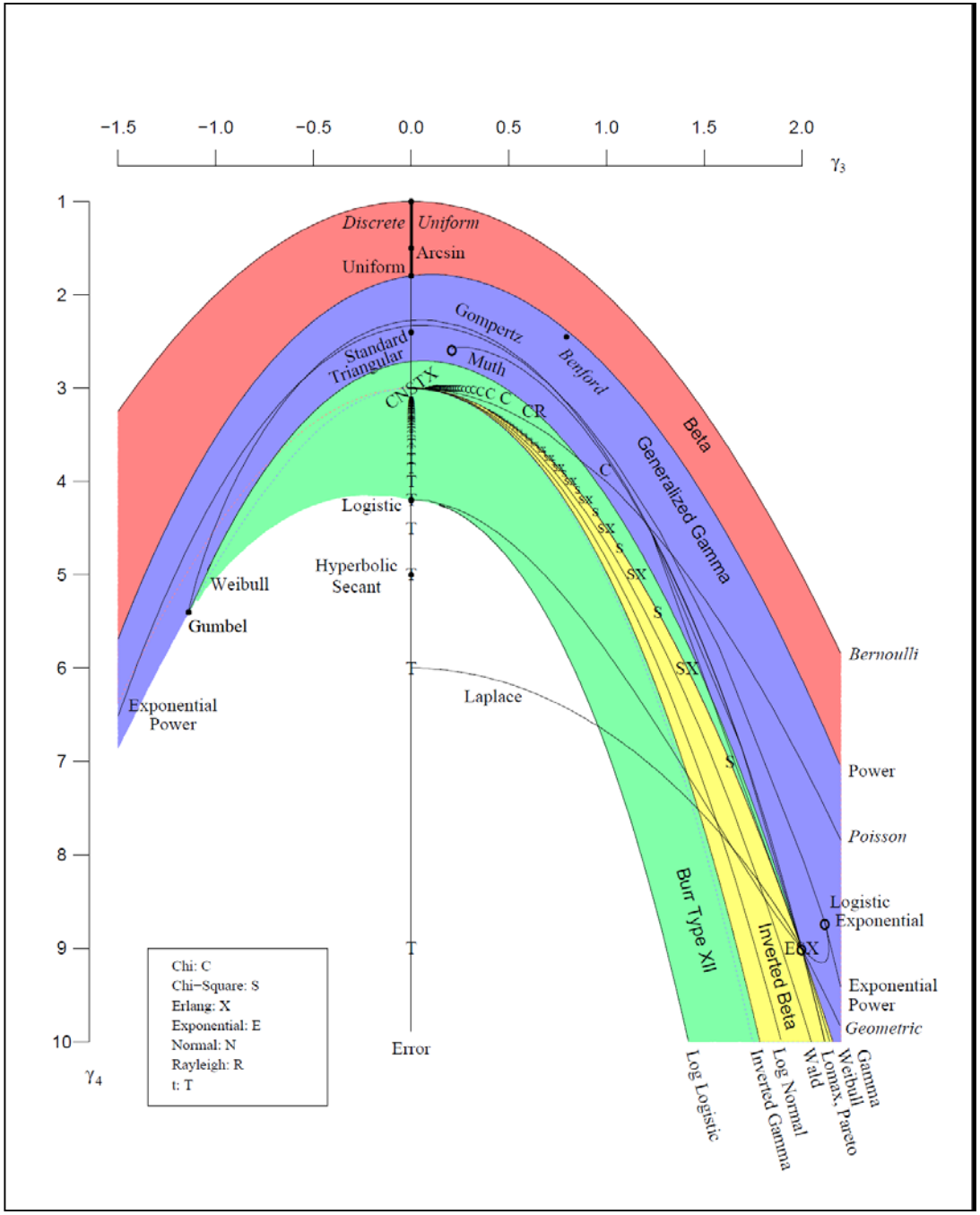
Then X' has mean μ and variance σ^2, but the *same* skewness and kurtosis as X.

So matching mean and variance is easy; the challenge is in matching α_3 and α_4.

There are distributions (e.g., Johnson, GLD) designed to cover all or nearly all of the (α_3^2, α_4) plane.

Skewness vs. Kurtosis for common univariate distributions

Vargo et al. 2010. *Journal of Quality Technology* **42** (3): 1-11.



http://www.math.wm.edu/~leemis/2010jqt.Plot4.pdf

Moment matching approach

1. Compute the sample standardized central moments of the input data:

$$
\begin{aligned}
\bar{X} &= \frac{1}{m}\sum_{i=1}^{m} X_i \\
\widehat{\sigma}^2 &= \frac{1}{m}\sum_{i=1}^{m} (X_i - \bar{X})^2 \\
\widehat{\alpha}_3 &= \frac{1}{m}\sum_{i=1}^{m} (X_i - \bar{X})^3 / \widehat{\sigma}^3 \\
\widehat{\alpha}_4 &= \frac{1}{m}\sum_{i=1}^{m} (X_i - \bar{X})^4 / \widehat{\sigma}^4.
\end{aligned}
$$

2. Express central moments of $F(\cdot;\boldsymbol{\theta})$ as functions of $\boldsymbol{\theta}$: $\mu(\boldsymbol{\theta}), \sigma^2(\boldsymbol{\theta}), \alpha_3(\boldsymbol{\theta})$ and $\alpha_4(\boldsymbol{\theta})$.

3. Match the moments by solving the system of equations for $\boldsymbol{\theta}$:

$$\begin{aligned} \mu(\boldsymbol{\theta}) &= \bar{X} \\ \sigma^2(\boldsymbol{\theta}) &= \widehat{\sigma}^2 \\ \alpha_3(\boldsymbol{\theta}) &= \widehat{\alpha}_3 \\ \alpha_4(\boldsymbol{\theta}) &= \widehat{\alpha}_4 \end{aligned}$$

Example: The $\mathrm{gamma}(\alpha, \beta)$ distribution has $\mu(\alpha, \beta) = \alpha\beta$ and $\sigma^2(\alpha, \beta) = \alpha\beta^2$, so solve $\widehat{\alpha}\widehat{\beta} = \bar{X}$ and $\widehat{\alpha}\widehat{\beta}^2 = \widehat{\sigma}^2$.

FYI: $\alpha_3 = 2/\sqrt{\alpha}$ and $\alpha_4 = 3 + 6/\alpha$ for the gamma.

Other things to match

Let $\widehat{F}$ be the empirical cdf (ecdf) of the data

$$\widehat{F}(x) = \frac{1}{m}\sum_{i=1}^{m} I(X_i \le x)$$

and let $X_{(1)} \le X_{(2)} \le \cdots \le X_{(m)}$ be the order statistics (sorted data).

Then we could try to pick $\widehat{\boldsymbol{\theta}}$ so that

$$F(X_{(i)}; \widehat{\boldsymbol{\theta}}) \approx \widehat{F}(X_{(i)}) = \frac{i}{m}$$

The book describes a least squares approach for the GLD.

Empirical distributions

Using the ecdf $\widehat{F}$ is easy (is this inverse cdf?):

1. Generate $U \sim U(0,1)$
2. Set $i = \lceil mU \rceil$
3. Return $\widehat{X} = X_{(i)}$

The ecdf is unbiased for F_X and matches sample properties; e.g., for $\widehat{X} \sim \widehat{F}$

$$\mathrm{E}\left(\widehat{X} \,\middle|\, X_1, \ldots, X_m\right) = \sum_{i=1}^{m} X_{(i)} \frac{1}{m} = \bar{X}.$$

Interpolated ecdf

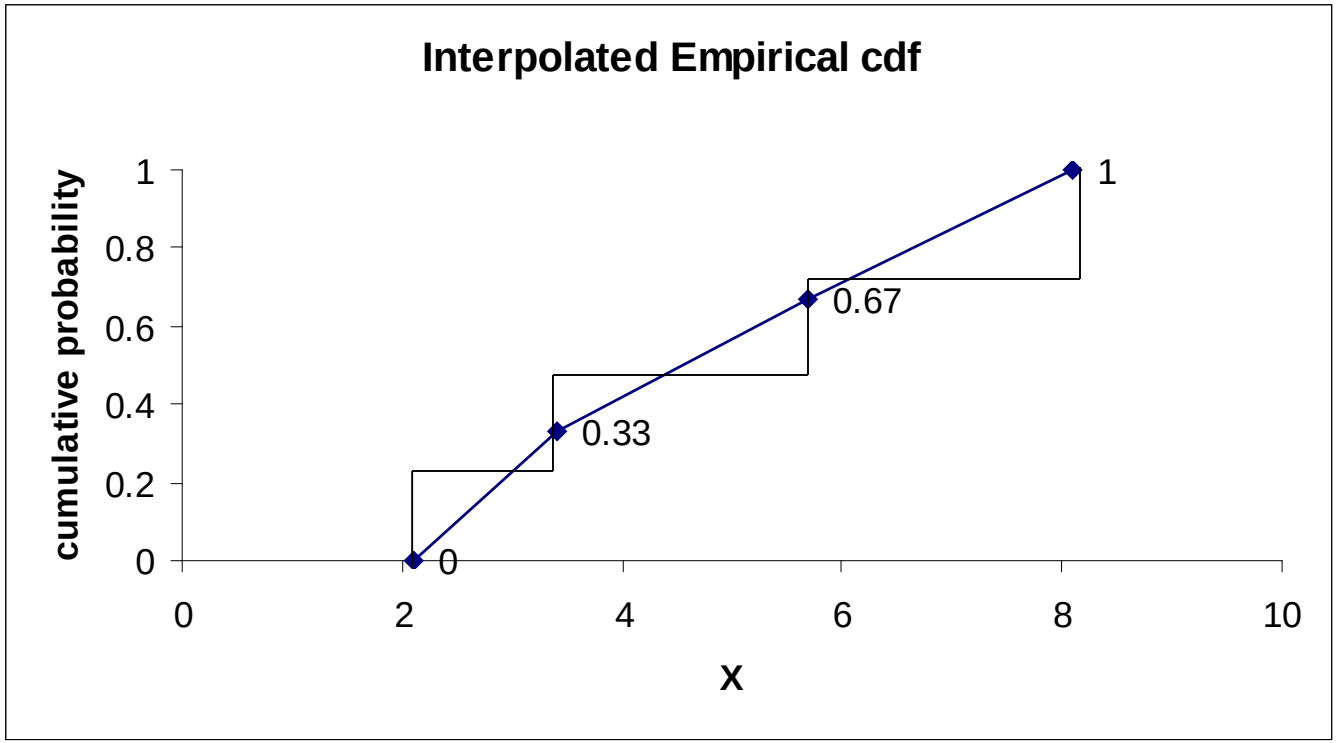
We can fill in the gaps with linear interpolation:

$$\tilde{F}(x) = \begin{cases} 0, & x < X_{(1)} \\ \dfrac{i-1}{m-1} + \dfrac{x - X_{(i)}}{(m-1)(X_{(i+1)} - X_{(i)})}, & X_{(i)} \le x < X_{(i+1)} \\ 1, & x \ge X_{(m)}. \end{cases}$$

Inversion:

1. Generate $U \sim U(0,1)$
2. Set $i = \lceil (m-1)U \rceil$
3. Return $\tilde{X} = X_{(i)} + (m-1)(X_{(i+1)} - X_{(i)}) \left(U - \dfrac{i-1}{m-1} \right)$

Empirical cdf and interpolated empirical cdf for data points $X = \{5.7, 2.1, 3.4, 8.1\}$



Notice that the interpolated ecdf fills in the gaps between observed values, but the domain is still limited to the smallest and largest observed values (no tails).

Properties of the interpolated ecdf

$\tilde{F}$ is no longer unbiased, nor does it exactly match sample properties. But it is consistent. For any fixed x, let

$$\bar{F}(x) = \frac{1}{m-1}\sum_{i=1}^{m} I(X_i \le x) = \frac{m}{m-1}\widehat{F}(x).$$

Then $\bar{F}(x) \xrightarrow{a.s.} F_X(x)$ because $m/(m-1) \to 1$. Also for $X_{(i)} \le x < X_{(i+1)}$ we have

$$0 \le \frac{x - X_{(i)}}{(m-1)(X_{(i+1)} - X_{(i)})} < \frac{1}{m-1}.$$

Therefore, $\bar{F}(x) - \dfrac{1}{m-1} \le \tilde{F}(x) < \bar{F}(x)$.

Since both the lower and upper bounds on $\tilde{F}(x)$ converge a.s. to $F_X(x)$, so does $\tilde{F}(x)$.

Input modeling without data

1. Use the process physics to suggest a good choice, then translate subjective information people can give you into parameters.

   Example: If given the 30th and 80th percentiles $x_{0.3}$ and $x_{0.8}$ of a normal distribution, then solve

$$\begin{aligned} \mu + z_{0.3}\sigma &= x_{0.3} \\ \mu + z_{0.8}\sigma &= x_{0.8} \end{aligned}$$

2. Use a distribution with intuitive parameters; e.g., the triangular distribution uses minimum, maximum and most likely values.

Nonstationary arrival processes

Arrival processes are fundamental to many OR simulations. In addition to scheduled arrivals (how would you simulate?), the most common model is random *interarrival* times.

Our background in queueing theory tends to make us think about *stationary renewal arrival processes*, meaning i.i.d. interarrival times.

When the arrival rate changes dramatically over time, approximating it as stationary (e.g., average rate, max rate) may miss a critical aspect of system behavior.

Renewal arrivals

Interarrival times: $\tilde{A}_1, \tilde{A}_2, \ldots$ i.i.d. nonnegative with distribution G (finite mean, variance and has density)

Arrival times: $\tilde{S}_n = \begin{cases} 0, & n = 0 \\ \sum_{i=1}^{n} \tilde{A}_i = \tilde{S}_{n-1} + \tilde{A}_n, & n = 1, 2, \ldots \end{cases}$

Arrival-counting process: $\tilde{N}(t) = \max\{n \geq 0 : \tilde{S}_n \leq t\}$

Arrival rate: $\lim_{t\to\infty} \dfrac{\mathrm{E}(\tilde{N}(t))}{t} = \tilde{\lambda} = \dfrac{1}{\mathrm{E}(\tilde{A}_i)}$

Equilibrium renewal: If $\tilde{A}_1 \sim G_e(t) = \tilde{\lambda} \displaystyle\int_0^t (1 - G(s))\, ds$

then $\dfrac{\mathrm{E}(\tilde{N}(t))}{t} = \tilde{\lambda}$ or $\mathrm{E}(\tilde{N}(t)) = \tilde{\lambda} t, \forall t.$

Nonstationary arrivals

To generalize to a time varying rate, let $N(t)$ be an arrival-counting process (possibly nonstationary) and

$$\Lambda(t) = \mathrm{E}(N(t))$$

the expected number of arrivals by time t (the"integrated rate function").

Then define the *arrival rate* to be

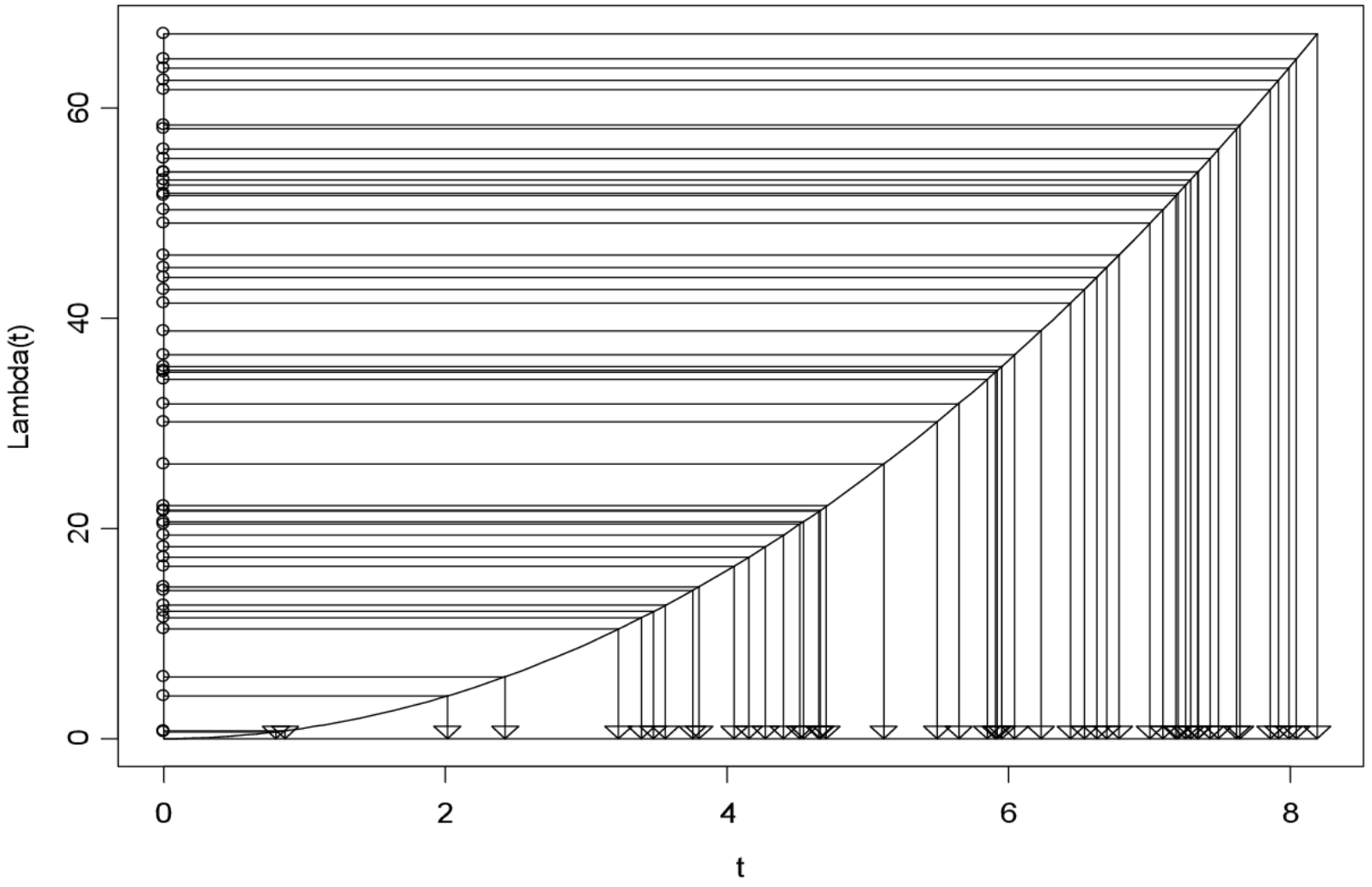
$$\lambda(t) = \frac{d}{dt}\Lambda(t)$$

To get arrival rate $\lambda(t)$, we transform equilibrium renewal arrivals with rate $\tilde{\lambda}$.

Inverting $\Lambda(t)$

1. Set index $n = 1$ and $\tilde{S}_0 = 0$
2. Generate $\tilde{A}_n$ with rate $\tilde{\lambda} = 1$
3. Let
   (a) $\tilde{S}_n = \tilde{S}_{n-1} + \tilde{A}_n$
   (b) $S_n = \Lambda^{-1}(\tilde{S}_n)$
   (c) $A_n = S_n - S_{n-1}$
4. $n = n + 1$
5. Go to Step 2

$$\lambda(t) = 2t \rightarrow \Lambda(t) = t^2 \rightarrow \Lambda^{-1}(s) = \sqrt{s}$$



Proof

Remember that $E(\tilde{N}(s)) = 1 \cdot s$

$$\begin{aligned} \mathrm{E}(N(t)) &= \mathrm{E}\left[\mathrm{E}\left(N(t) \left| \tilde{N}\left(\Lambda(t)\right)\right.\right)\right] \\ &= \mathrm{E}\left[\tilde{N}\left(\Lambda(t)\right)\right] \\ &= 1 \cdot \Lambda(t) = \Lambda(t) \end{aligned}$$

This method is ideal when $\Lambda(t)$ is easily invertible.

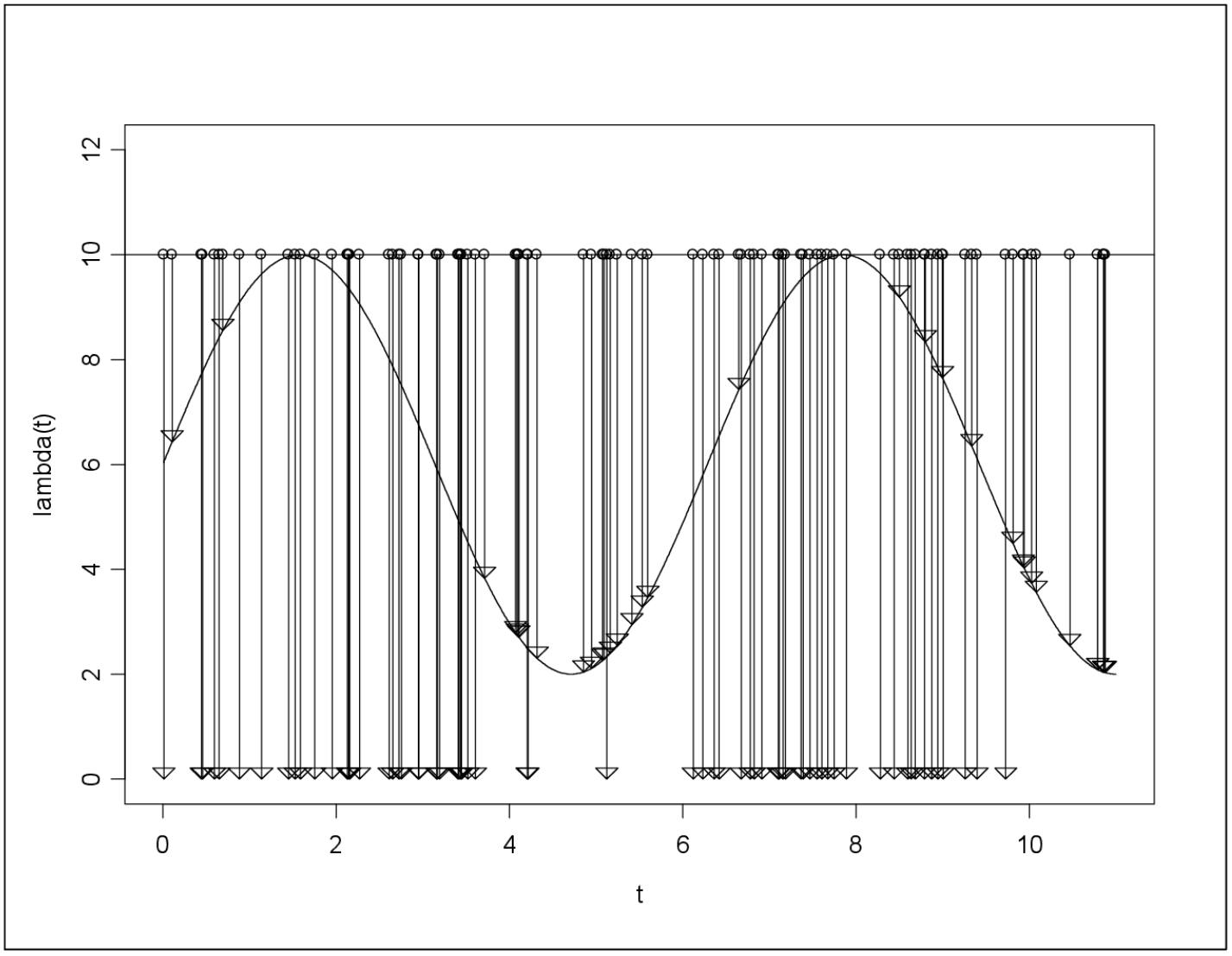
Any equilibrium renewal arrival process with rate 1 achieves the desired $\Lambda(t)$, but the specific choice affects other aspects of the arrivals (e.g., variability).

Thinning

1. Set indices $n = 1$ and $k = 1$ and $\tilde{S}_0 = 0$
2. Generate $\tilde{A}_n$ with rate $\tilde{\lambda} = \max_t \lambda(t)$, and let $\tilde{S}_n = \tilde{S}_{n-1} + \tilde{A}_n$
3. Generate $U \sim U(0, 1)$
4. If $U \leq \lambda(\tilde{S}_n)/\tilde{\lambda}$ then
   (a) $S_k = \tilde{S}_n$
   (b) $A_k = S_k - S_{k-1}$
   (c) $k = k + 1$

   Endif
5. $n = n + 1$
6. Go to Step 2

$\lambda(t) = 6 + 4\sin(t)$



Thinning with exponential base

1. Set indices $n = 1$ and $k = 1$ and $\tilde{S}_0 = 0$
2. Generate $V \sim U(0, 1)$, set $\tilde{A}_n = -\ln(1 - V) \cdot 10$, and let $\tilde{S}_n = \tilde{S}_{n-1} + \tilde{A}_n$
3. Generate $U \sim U(0, 1)$
4. If $U \leq (6 + 4\sin(\tilde{S}_n))/10$ then
   (a) $S_k = \tilde{S}_n$
   (b) $A_k = S_k - S_{k-1}$
   (c) $k = k + 1$

   Endif
5. $n = n + 1$
6. Go to Step 2

Exponential interarrival time

When $G(t) = 1 - e^{-\tilde{\lambda}t}$ then...

1. We call it a nonstationary Poisson process.
2. $G_e(t) = G(t)$ (memoryless property).
3. Inversion and thinning are probabilistically equivalent.
4. $\dfrac{\mathrm{Var}\left(N(t)\right)}{\mathrm{E}\left(N(t)\right)} = 1$ for all $t \geq 0$.

   This contrasts with general inversion for which

$$\frac{\mathrm{Var}\left(N(t)\right)}{\mathrm{E}\left(N(t)\right)} \approx \sigma_A^2 \text{ for large } t$$

Fitting $\lambda(t)$ or $\Lambda(t)$

A lot of interesting work has been done on fitting parametric arrival rate models; e.g.,

$$\lambda(t) = \exp\left\{\sum_{i=0}^{m} \alpha_i t^i + \beta \sin(\omega t + \phi)\right\}$$

If we stay nonparametric, then typically...

- Fit $\Lambda(t)$ is we have actual arrival times.
- Fit a piecewise-constant $\lambda(t)$ if we have only counts over intervals.

Fitting $\Lambda(t)$

Data

$$\{T_{ij}; i = 1, 2, \ldots, C_j(T)\},\ j = 1, 2, \ldots, k$$

where $C_j(t)$ is the cumulative number of arrivals by time t on the jth realization, $0 \leq t \leq T$.

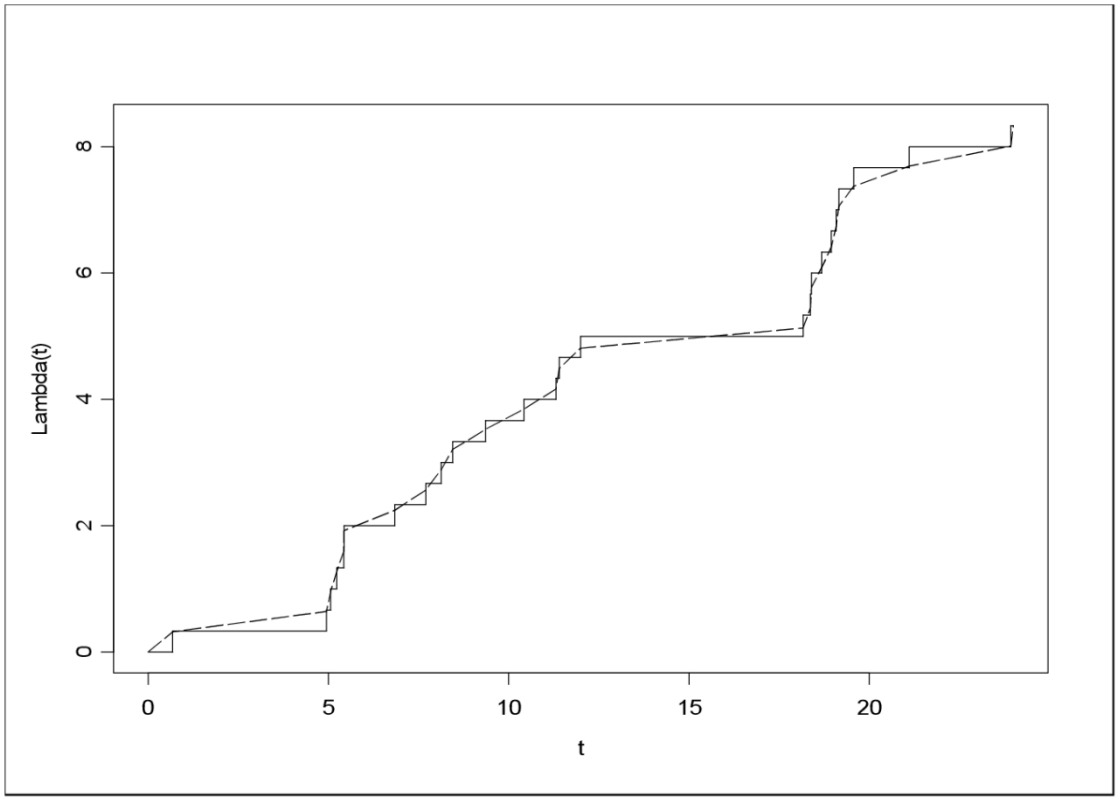
Example: Arrivals of emergency calls to 911 for a $T = 24$ hour period over $k = 3$ Mondays.

Merge these into a single data set of $C = \sum_{j=1}^{k} C_j(T)$ arrivals and sort

$$0 = T_{(0)} < T_{(1)} \leq T_{(2)} \leq \cdots \leq T_{(C)} < T_{(C+1)} = T$$

Linearly interpolated $\Lambda(t)$

$$\widehat{\Lambda}(t) = \left(\frac{C}{C+1}\right)\left\{\frac{i}{k} + \frac{1}{k}\left(\frac{t - T_{(i)}}{T_{(i+1)} - T_{(i)}}\right)\right\} \text{ when } T_{(i)} < t \leq T_{(i+1)}$$



Emergency call data for $k = 3$
Mondays over $T = 24$ hours.

Jumps $(C/(C+1))(1/k)$ at each arrival time.

$\widehat{\Lambda}(t) \xrightarrow{a.s.} \Lambda(t)$ as $k \to \infty$.

Is easily simulated using inversion.

Algorithm for inversion of $\Lambda(t)$

1. Set $n = 1$ and $S_0 = 0$
2. Generate $\tilde{S}_1 \sim G_e$
3. While $\tilde{S}_n \leq C/k$ do
   (a) $m = \left\lfloor \left(\frac{C+1}{C} \right) k\tilde{S}_n \right\rfloor$
   (b) $S_n = T_{(m)} + \left(T_{(m+1)} - T_{(m)}\right) \left(\left(\frac{C+1}{C} \right) k\tilde{S}_n - m \right)$
   (c) $A_n = S_n - S_{n-1}$
   (d) $n = n + 1$
   (e) Generate $\tilde{A}_n \sim G$
   (f) $\tilde{S}_n = \tilde{S}_{n-1} + \tilde{A}_n$

   Loop

Fitting piecewise constant $\lambda(t)$

Now assume arrival rate is constant over intervals of length $\delta > 0$ for which we have arrival counts.

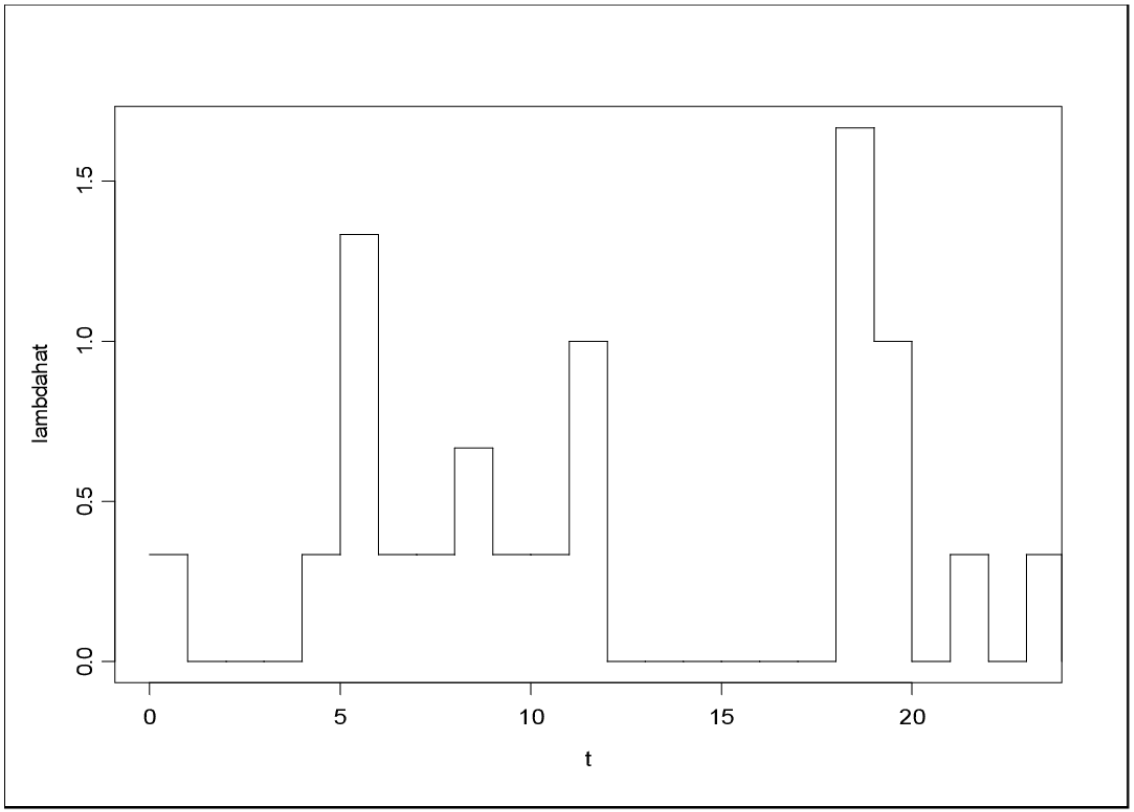
The arrival rate for an interval is just the average number of arrivals from the k realizations that fell in that interval per unit time:

$$\widehat{\lambda}(t) = \frac{1}{k\delta} \sum_{j=1}^{k} \left[\underbrace{C_j\left(\ell(t+\delta)\right) - C_j\left(\ell(t)\right)} \right]$$

where $\ell(t) = \lfloor t/\delta \rfloor \delta$.

Count of the number of arrivals on realization j for $i\delta < t \leq (i+1)\delta$ when $i = \lfloor t/\delta \rfloor$.

Piecewise constant $\lambda(t)$



Emergency call data for $k = 3$
Mondays with $\delta = 1$ hour.

Sparse arrivals can lead to $\widehat{\lambda}(t) = 0$ if δ too small.

δ too large can miss critical behavior.

Sensible when only have counts over fixed intervals; then use the natural δ.

Generating random variates

We know **inversion** is general. Even without a closed-form inverse, we can treat it as a root-finding problem:

$$\text{solve for } X\text{: } U = F_X(X) = \int_{-\infty}^{X} f_X(x)\,dx$$

Or use approximate inverses, such as

$$F_Z^{-1}(U) \approx \frac{U^{0.1349} + (1-U)^{0.1349}}{0.1975}$$

the GLD approximation to the standard normal inverse cdf which has about 1 decimal place accuracy.

Plus there are tricks to make inversion faster for discrete distributions.

Rejection

Applies when we can express

$$\Pr\{X \leq x\} = \Pr\{V \leq x | \mathcal{A}\}$$

where V is easy to generate and $\mathcal{A}$ is some event.

1. Generate V
2. If $\mathcal{A}$ occurs, then return $X = V$
   Otherwise, reject V and go to Step 1

Example: Generate $\{1, 2, 3, 4, 5\}$ equally likely by rolling a die but rejecting 6's.

An approach for $f_X(x)$ a density

Suppose $m(x) \geq f_X(x)$ for all x ("majorizes"). Then a density with the same shape is

$$g(x) = \frac{m(x)}{\int_{-\infty}^{\infty} m(y)\, dy} = \frac{m(x)}{c}$$

Rejection:

1. Generate $V \sim g$ (needs to be easy)
2. Generate $U \sim U(0, 1)$
3. If $U \leq f_X(V)/m(V)$ then return $X = V$
   Otherwise go to Step 1

Intuition

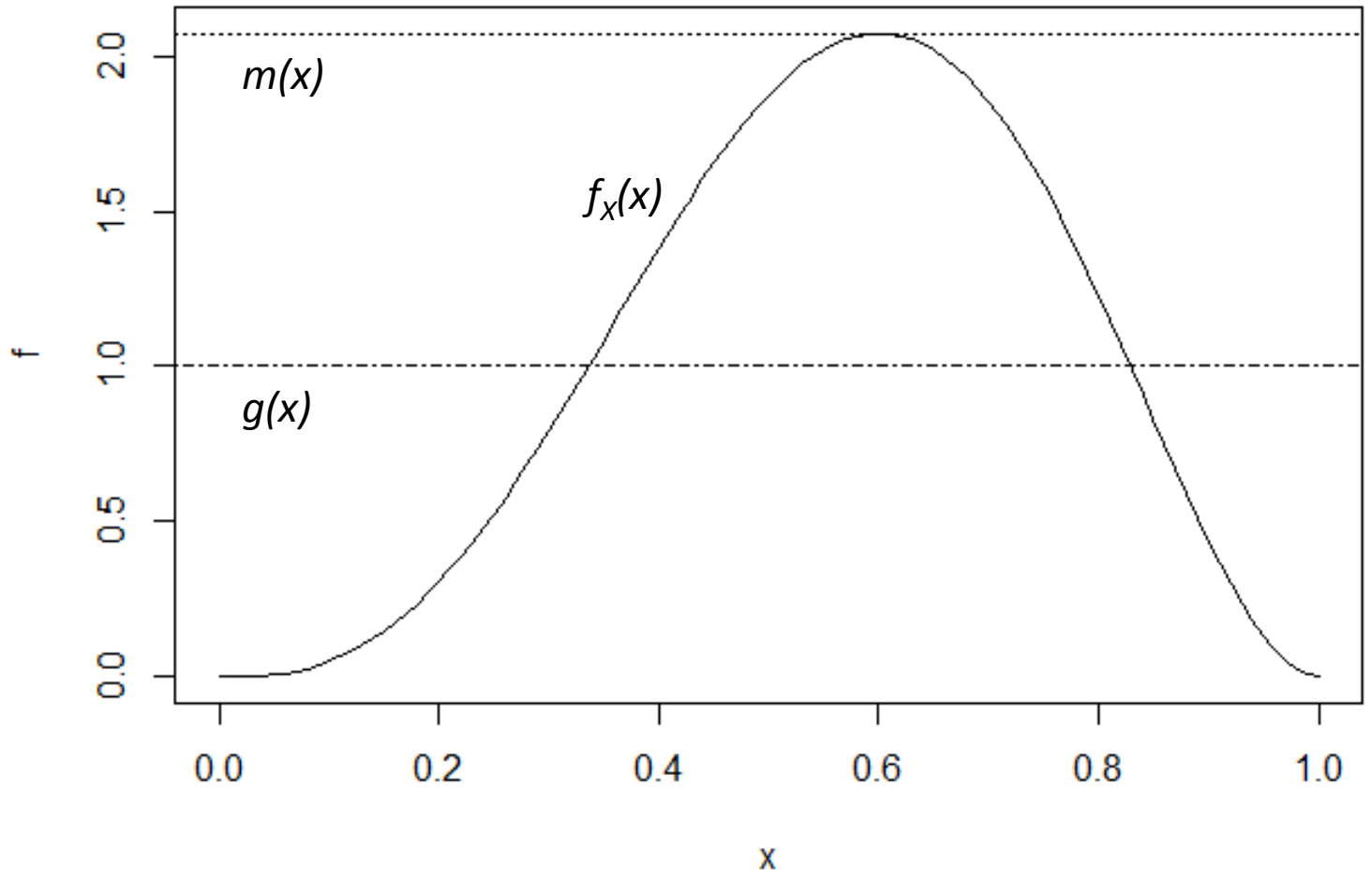
Notice that if $m(x) = f_X(x)$, then $g(x) = f_X(x)$ and we accept every time, as we should.

The density g redistributes the probability, making some x's more likely than f_X does, and others less likely.

Since $g \propto m$, where g makes x's more likely than f_X, then $m(x)$ will be big, $f_X(x)/m(x)$ will be small and we tend to reject.

Similarly where g makes x's less likely than f_X, then $m(x)$ will be small, $f_X(x)/m(x)$ will be big and we tend to accept.

Beta majorized by uniform



Beta distribution example

The beta density $f_X(x) = x^{\alpha_1-1}(1-x)^{\alpha_2-1}/\mathrm{B}(\alpha_1,\alpha_2)$ has its mode at $x^\star = (\alpha_1-1)/(\alpha_1+\alpha_2-2)$ when $\alpha_1 > 1$, $\alpha_2 > 1$. Use $m(x) = f_X(x^\star)$.

0. Compute $f^\star = f_X\left(\frac{\alpha_1-1}{\alpha_1+\alpha_2-2}\right)$
1. Generate $V \sim g = U(0,1)$
2. Generate $U \sim U(0,1)$
3. If $U \le f_X(V)/m(V) = [V^{\alpha_1-1}(1-V)^{\alpha_2-1}/\mathrm{B}(\alpha_1,\alpha_2)]/f^\star$ then return $X = V$

   Otherwise go to Step 1

Key proof steps

Need to show that

$$\Pr\{V \le x | \mathcal{A}\} = \frac{\Pr\{V \le x, U \le f_X(V)/m(V)\}}{\Pr\{U \le f_X(V)/m(V)\}} = F_X(x).$$

$$\begin{aligned}
&\Pr\{V \le x, U \le f_X(V)/m(V)\} \\
&= \int_{-\infty}^{\infty} \Pr\{y \le x, U \le f_X(y)/m(y) | V = y\} g(y)\, dy \\
&= \int_{-\infty}^{x} \Pr\{U \le f_X(y)/m(y)\} g(y)\, dy \\
&= \frac{1}{c} \int_{-\infty}^{x} f_X(y)\, dy = \frac{1}{c} F_X(x)
\end{aligned}$$

Rejection notes

- Number of trials to get one X is geometric with expected value c (can you see why?).
- The beta example had $c = 2.0736$ which is very bad. Good rejection algorithms have c very close to 1. These algorithms tend to majorize f_X in a piecewise manner.
- We can develop a corresponding rejection algorithm for discrete distributions.

Particular properties

Exploit relationships among distributions:

Normal (μ, σ^2)**:** $X = \mu + \sigma \cdot \mathrm{N}(0,1)$

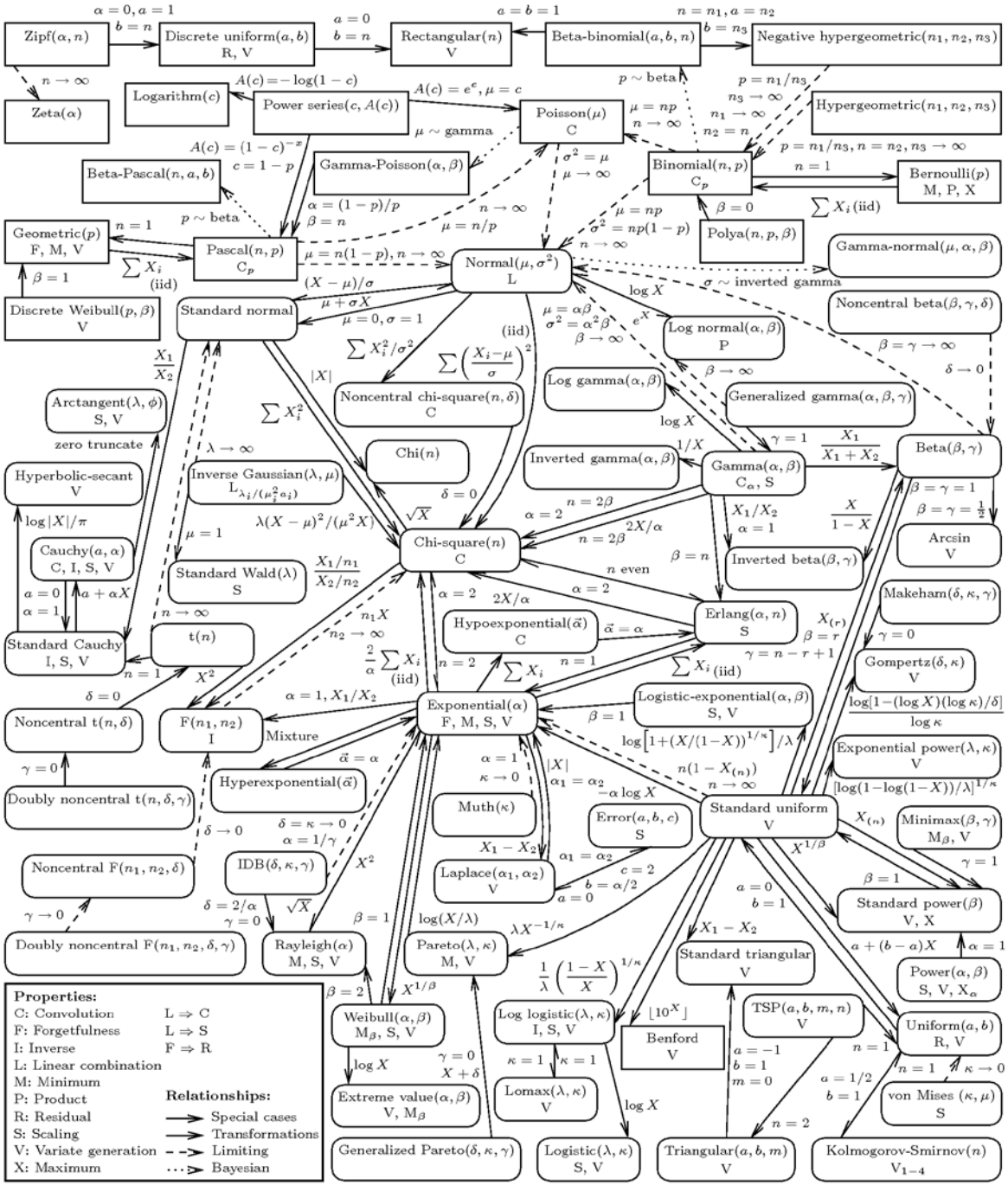
Lognormal: $Y = \exp(X)$

Beta: If $X_i \sim \mathrm{gamma}(\alpha_i, \beta)$ independent, then $X = X_1/(X_1 + X_2)$ is $\mathrm{beta}(\alpha_1, \alpha_2)$.

But avoid approximations and use exact methods.

Example: $Z = \sum_{i=1}^{12} U_i - 6$ for standard normal.

An interactive version of this chart can be found at www.math.wm.edu/~leemis/chart/UDR/UDR.html



Generating pseudorandom numbers

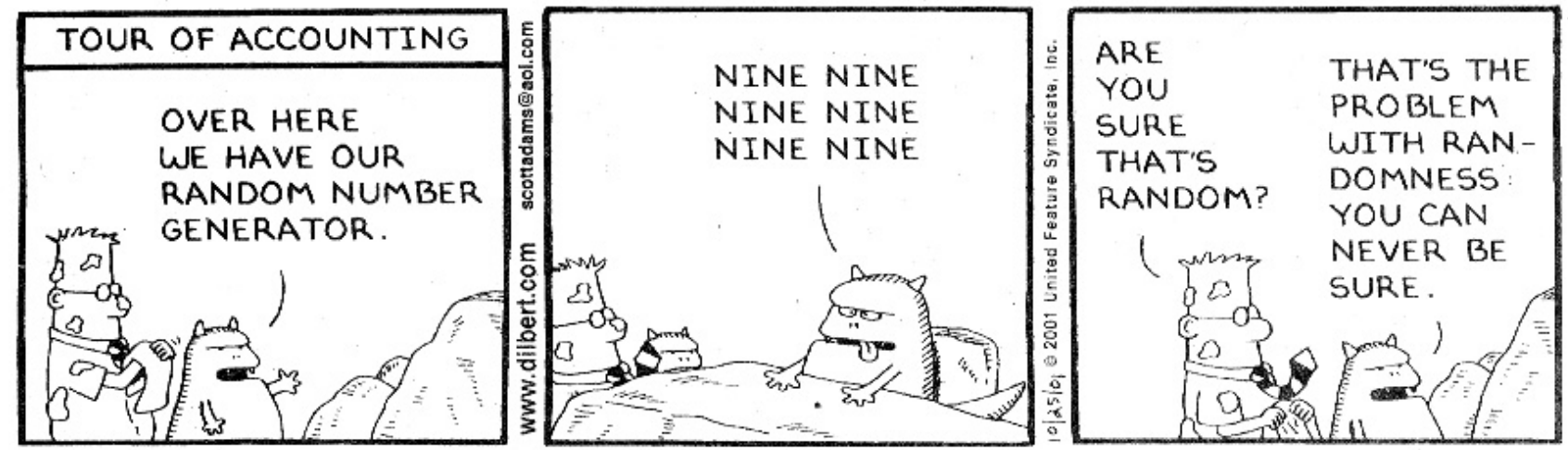
The theory of simulation is based on having a source of i.i.d. $U(0,1)$ random variates. In reality we use a deterministic, algorithm-generated list that repeats

$$u_1, u_2, u_3, \ldots, u_i, u_{i+1}, \ldots, u_{P-1}, u_P, u_1, u_2, u_3, \ldots$$

Producing good pseudorandom number generators (RNG) is *hard*.

On the interval $(0,1)$ the values $u_1, u_2, u_3, \ldots$ can appear i.i.d. uniform, but in $(0,1)^d$ the points $(u_{id+1}, u_{id+2}, \ldots, u_{id+d})$ may appear very non-random.

Modern generators have periods longer than we can test.


DILBERT
TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
www.dilbert.com scottadams@aol.com
NINE NINE NINE NINE NINE NINE
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RAN-DOMNESS: YOU CAN NEVER BE SURE.
© 2001 United Feature Syndicate, Inc.

Building block: MCG

Multiplicative congruential generator (MCG):

$$\begin{aligned} z_i &= a z_{i-1} \mod m \\ u_i &= \frac{z_i}{m} \end{aligned}$$

In VBASim

$$\begin{aligned} z_i &= 630{,}360{,}016 z_{i-1} \mod 2^{31}-1 \\ u_i &= \frac{z_i}{2^{31}-1} \end{aligned}$$

Maximal period is $P = m - 1 \approx 2$ billion generating $\{0, 1/m, 2/m, \ldots, (m-1)/m\}$. Why?

We don't want 0 or 1. Why?

Extending the period

Multiple recursive generators (MRG): Max $P = m^K - 1$

$$z_i = (a_1 z_{i-1} + a_2 z_{i-2} + \cdots + a_K z_{i-K}) \mod m$$

$$u_i = \begin{cases} \dfrac{z_i}{m+1}, & z_i > 0 \\ \dfrac{m}{m+1}, & \text{otherwise.} \end{cases}$$

Combined generators: Max $P = (P_1 P_2 \cdots P_J)/2^{J-1}$

$$z_i = (\delta_1 z_{i,1} + \delta_2 z_{i,2} + \cdots + \delta_J z_{i,J}) \mod m_1$$

$$u_i = \begin{cases} \dfrac{z_i}{m_1+1}, & z_i > 0 \\ \dfrac{m_1}{m_1+1}, & \text{otherwise} \end{cases}$$

Each $z_{i,j}$ a MRG.

L'Ecuyer's MRG32k3a

Uses $J = 2$ MRGs of order $K = 3$.

$$
\begin{aligned}
z_{i,1} &= (1{,}403{,}580\, z_{i-2,1} - 810{,}728\, z_{i-3,1}) \mod 2^{32} - 209 \\
z_{i,2} &= (527{,}612\, z_{i-1,2} - 1{,}370{,}589\, z_{i-3,2}) \mod 2^{32} - 22{,}853 \\
z_i &= (z_{i,1} - z_{i,2}) \mod 2^{32} - 209.
\end{aligned}
$$

Has $P \approx 3 \times 10^{57}$. If you could generate 2 billion U's per second, it would take longer than the age of the universe to exhaust this generator!

A great deal of theory and computation was required to prove good properties.

Proper use of RNGs

To get RNGs started we need a *seed*. A single seed z_0 is needed for a MCG; several seeds are needed for a combined MRG.

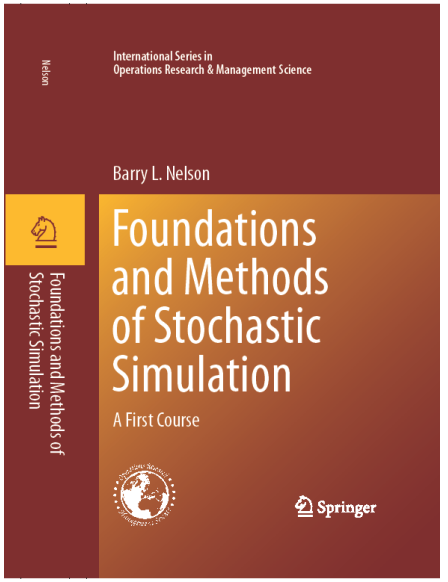
RNGs provide multiple *streams* (pointers to seeds) spaced far apart in the sequence. Why?

VBASim: `Expon(Mean, Stream)`

Provided we make enough replications or long enough runs, no seed/stream is better than any other in a good RNG.

Streams exist to help us make *comparisons*. More later.

Question: Why don't we randomly sample the starting seed?



Chapter7: Simulation Output

©Barry L. Nelson

Northwestern University

February 2013

Simulation output

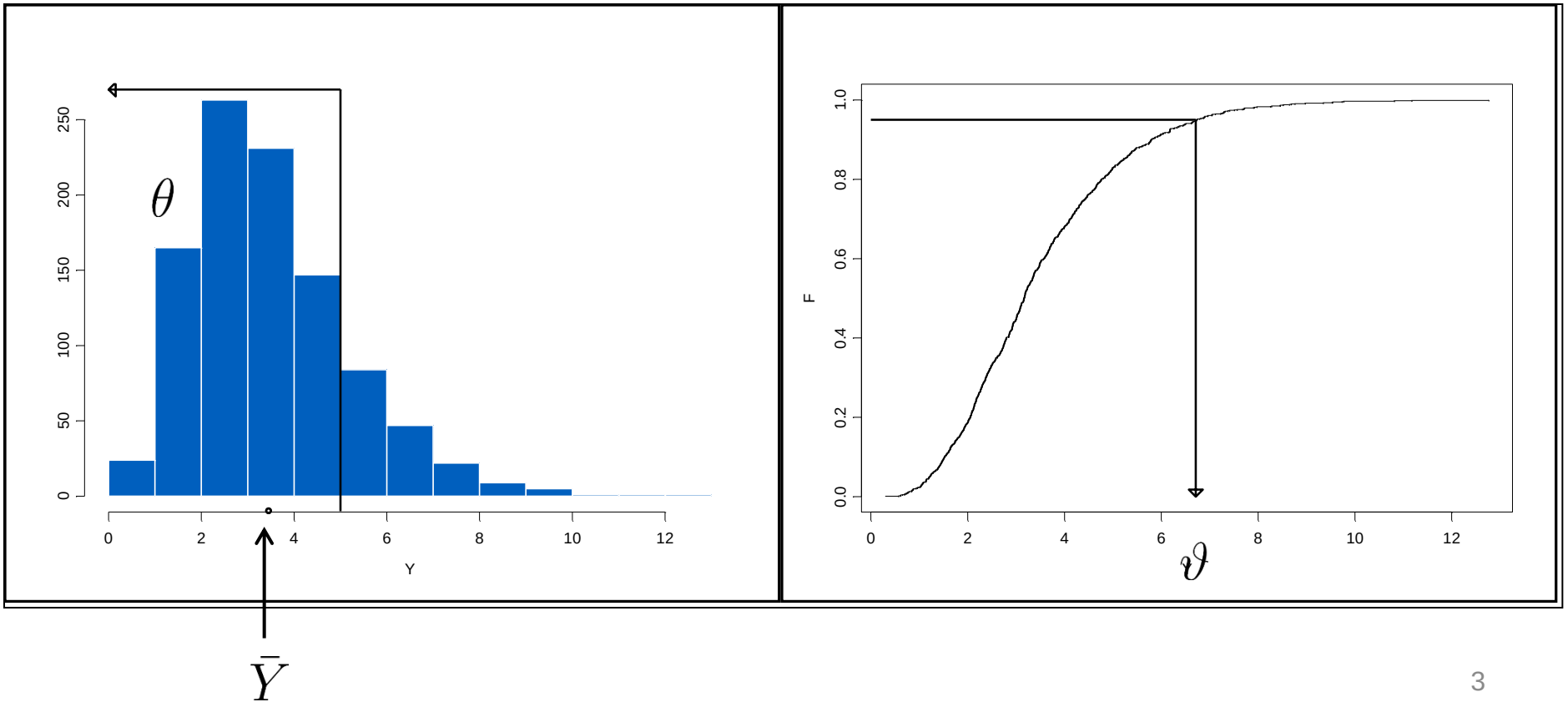
Suppose we make $n = 1000$ replications of Y, the time to complete the SAN. Let $Y_{(1)} \leq \cdots \leq Y_{(n)}$ be the *order statistics*.

What performance measures might be relevant?

- Mean time to complete the project, $\mu = \mathrm{E}(Y)$ estimated by the sample mean $\bar{Y} = \sum_{i=1}^{1000} Y_i/1000$.
- Probability we complete the project in 5 days $\theta = F_Y(5)$, estimated by $\widehat{F}(5) = \#\{Y_i \leq 5\}/1000$.
- The 0.95 quantile $\vartheta = F_Y^{-1}(0.95)$, which is the date we can promise and be 95% sure of making it, estimated by $\widehat{F}^{-1}(0.95) = Y_{(950)}$.

Visualization

Visualizing means, probabilities and quantiles using the histogram and empirical cdf of 1000 SAN project completion times.



What measures are relevant?

For project planning, μ really does not make much sense.

The project will almost certainly not complete in exactly μ days, and it may not even be the most likely value.

If we think of the mean as the "long-run average," then it is most relevant *when the long-run average is what we will see* rather than a one-time outcome.

For the hospital information kiosk ($M/G/1$ queue), the long-run average waiting time is meaningful because the kiosk will serve many patients and visitors.

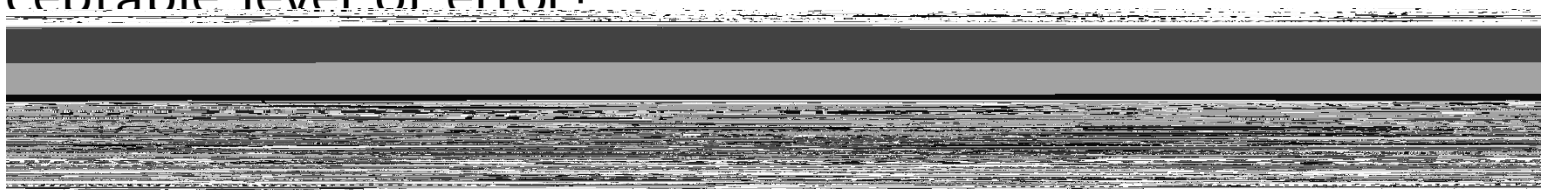
Q: Is $S(Y)$ relevant for the SAN? What about $S(Y)/\sqrt{1000}$?

Measures of error

No matter what performance measure we estimate, we need a *measure of error* (MOE) to establish how good it is.

Without an MOE, we cannot know if *any* of the digits in the estimate can be believed.

MOEs are also useful for experiment design: What number of replications and/or runlength is needed to attain an acceptable level of error?



CI for the mean

CI for the probability

The estimator of θ is still a sample mean

$$\widehat{F}(y) = \frac{1}{n}\sum_{i=1}^{n} I(Y_i \leq y)$$

CI for the q quantile

The q quantile $\vartheta = F_Y^{-1}(q)$ implies that $\Pr\{Y \leq \vartheta\} = q$.

Suppose we observe i.i.d. $Y_1, Y_2, \ldots, Y_n$. Then

Normal approximation

In general $\widehat{\vartheta} = \widehat{F}^{-1}(q) = Y_{(\lceil nq \rceil)}$ with CI $[Y_{(\ell)}, Y_{(u)}]$.

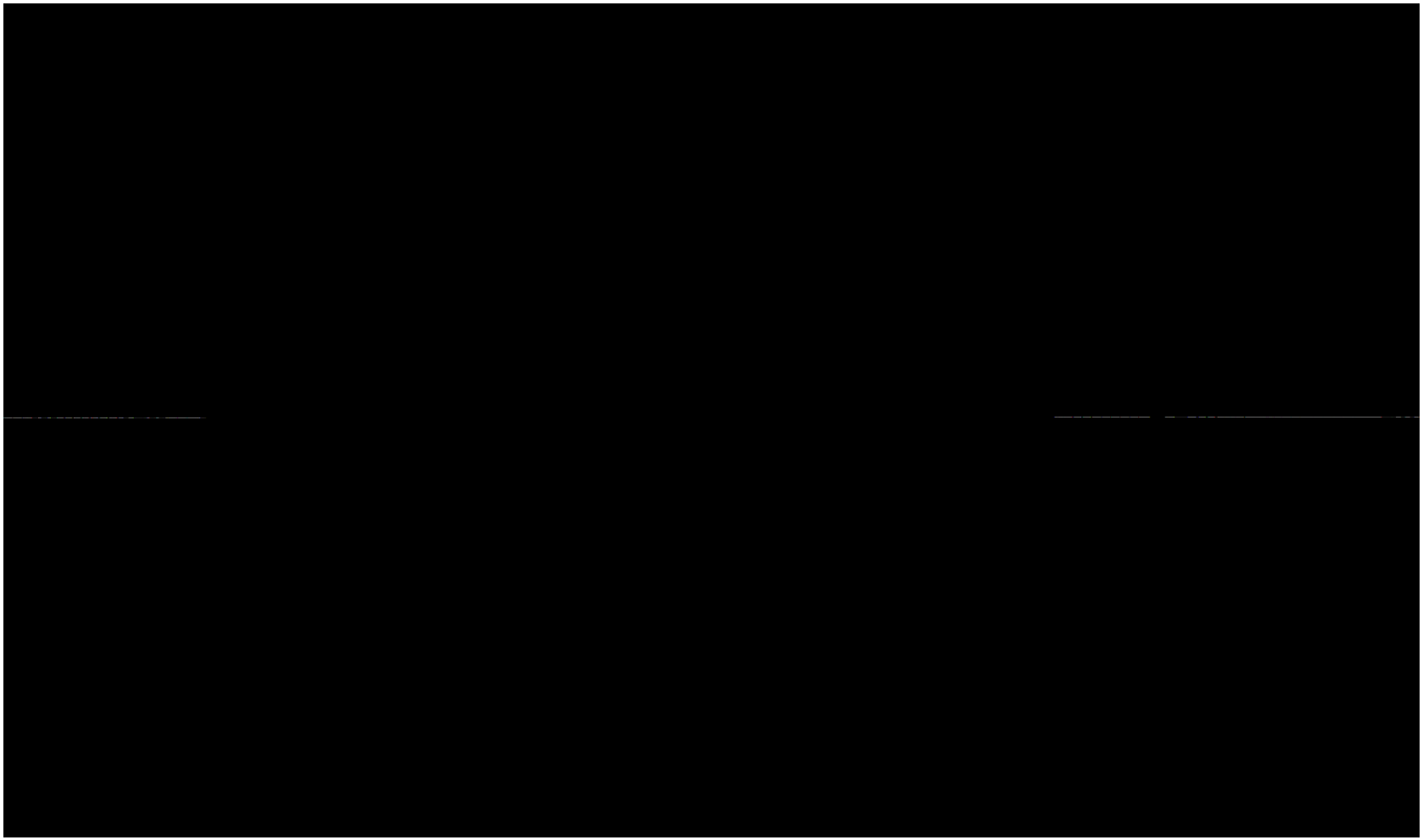
How hard is quantile estimation?

Just as probability estimation is relatively more difficult as

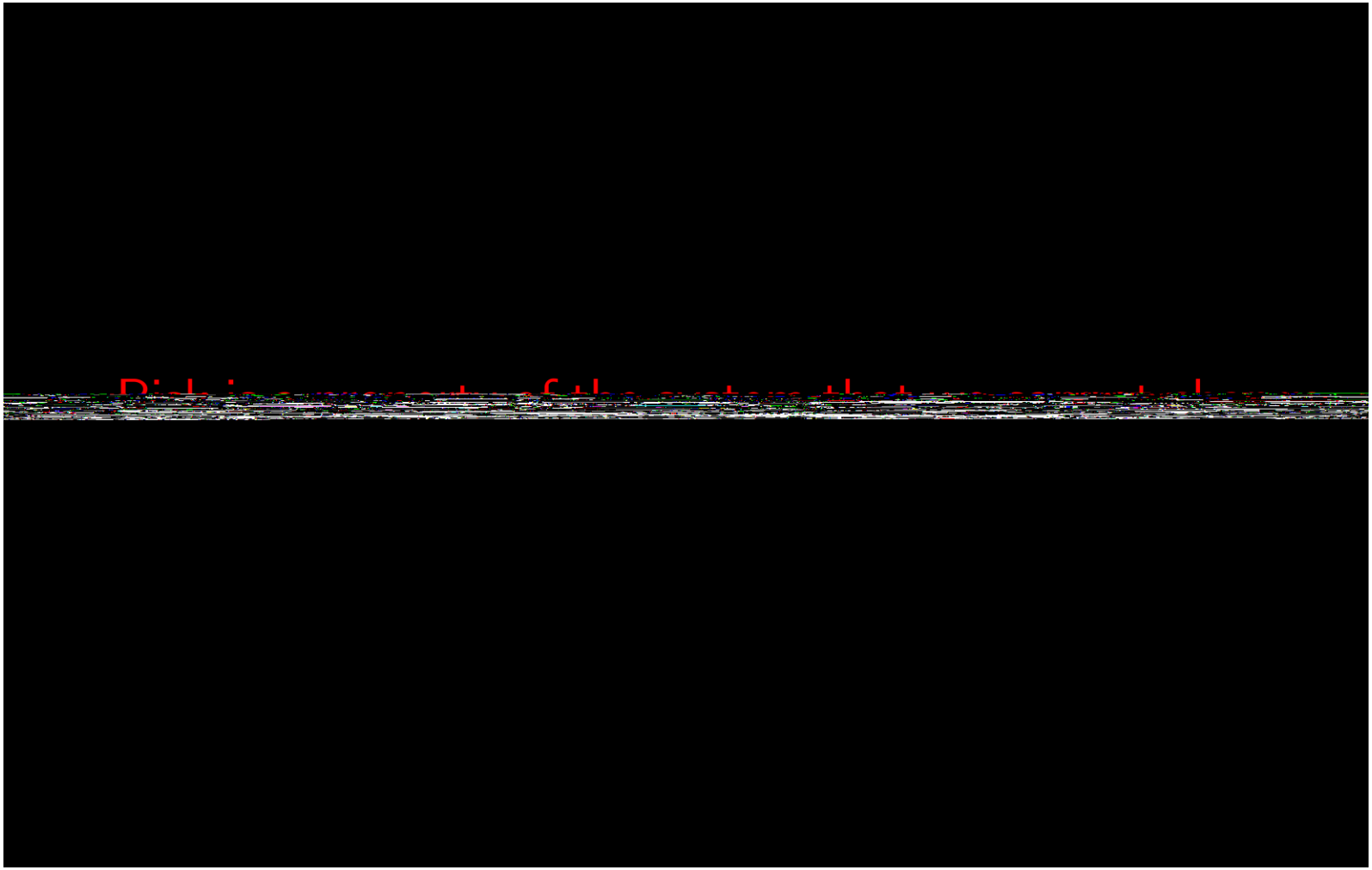
Example: Extreme quantiles

Suppose $f_Y(y) = \mathrm{e}^{-y}, y \geq 0$. Then $\vartheta = -\ln(1-q)$ so

$$\mathrm{se}\left(\widehat{\vartheta}\right) \approx \sqrt{\frac{q(1-q)}{[\quad\quad]^2}} = \sqrt{\frac{q}{(1-\quad)}}.$$



Risk vs. Error

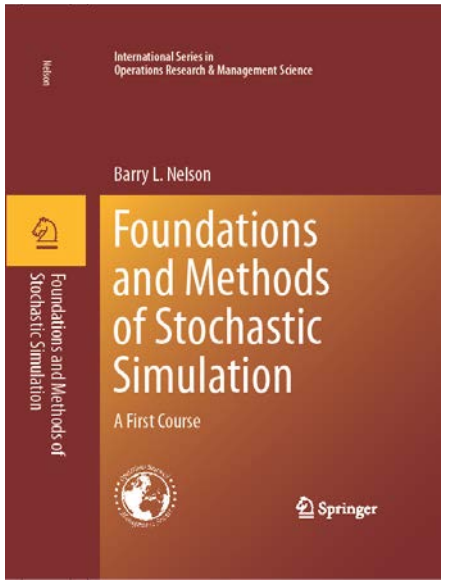


Idea: Bootstrap

1. Given real-world data $\{X_1, X_2, \ldots, X_m\}$, do:
2. For i from 1 to b
   (a) Generate the bootstrap sample $X_1^\star \quad X_2^\star \qquad X_m^\star$

M/M/ f: More details

We have $\{A_1, \ldots, A_{100}\}$ interarrival times and $\{X_1, \ldots, X_{100}\}$



Chapter 8.2, 8.3 & 8.4: Steady-state Simulation Simulation Optimization

©Barry L. Nelson

Northwestern University

February 2015

Extended asymptotic analysis

$$\mathrm{MSE}\left(\bar{Y}(n,m,d)\right) \approx \frac{\beta(d)^2}{(m-d)^2} + \frac{\gamma^2}{n(m-d)}$$

Fixed-precision problem

Fixed-budget problem

Given a budget of N observations, solve

Deletion with a fixed budget

Calculating MSER

Batch means variance estimator

The natural estimator of $\mathrm{Var}\left(\bar{Y}(b)\right)$ is

$$S^2(k) = \frac{1}{k-1}\sum^{k}\left(\bar{Y}_i(b) - \bar{Y}(m)\right)^2.$$

When will this "work"?

We need $b \geq b^\star$ (equivalently $k \leq k^\star$) where

$$\sigma^2 \left(1 + 2 \sum^{b^\star} \rho_i\right) \approx \gamma^2 = \sigma^2 \left(1 + 2 \sum^{\infty} \rho_i\right)$$

Simplest (sensible) algorithm

1. Make a single replication of length $m = N$.
2. Apply MSER to obtain a deletion point d (or for an even simpler procedure, set $d = 0$).

Why "$10 \leq k \leq 30$"?

Summary

1. There is a tradeoff between bias and variance, and both

Simulation Optimization

Simulation optimization formulation

$$\min \theta(\mathbf{x})$$

Experiment design for SO

$$\widehat{\theta}(\mathbf{x}; T, n, \mathbf{U})$$

$\mathbf{x}$ defines the scenario, and may be categorical, discrete-

Example: SAN resource allocation

Suppose activity i is exponentially distributed with mean τ_i

Solution for 1: Convergence

Solution for 3: Clean up

Common random numbers

Lessons learned

- Two aspects of this example are important more generally:
  1. **Monotonicity** of random numbers → inputs → outputs.
     - The inverse cdf method helps here.
  2. **Synchronization** of how the random numbers are used by each scenario.
     - Random number "streams" help here, but may not be enough.

Design & analysis for correct selection

Suppose there are only K scenarios $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K$, we would like a guarantee of selecting the best among them and there is time to simulate them all.

Foundation

Foundation

Example: Adaptive hyperbox algorithm

Infinitesimal perturbation analysis

Suppose that we fix the pseudorandom numbers. If $A(x_\cdot)$ is on

IPA

Choice of gradient estimator

SAN example

The output from replication i of the SAN simulation can be

If we replace the $\max$ operator by three inequality constraints then the SAA problem can be formulated as a linear program:

$$
\begin{aligned}
\min \quad & \frac{1}{n}\sum_{j=1}^{n} y_j \\
y_j \;&\geq\; -\ln(1-u_{1j})x_1 - \ln(1-u_{4j})x_4 \\
y_j \;&\geq\; -\ln(1-u_{1j})x_1 - \ln(1-u_{3j})x_3 - \ln(1-u_{5j})x_5 \\
y_j \;&\geq\; -\ln(1-u_{2j})x_2 - \ln(1-u_{5j})x_5, \;\; j = 1, 2, \ldots, n \\
b \;&\geq\; \sum_{k=1}^{5} c_k(\tau_k - x_k) \\
x_k \;&\geq\; \ell_k, \;\; k = 1, 2, \ldots, 5.
\end{aligned}
$$

This linear program has $n+5$ decision variables $y_1, y_2, \ldots, y_n$, $x_1, x_2, \ldots, x_5$ and $3n + 6$ constraints.

Uniform convergence

What we hope is that $\theta(\widehat{\mathbf{x}}_n^\star) \to \theta(\mathbf{x}^\star)$, but pointwise convergence is not enough since $\widehat{\mathbf{x}}_n^\star$ is not a fixed value of $\mathbf{x}$ but rather the solution to the SAA problem over *all* $\mathbf{x} \in \mathsf{C}$.

Uniform convergence means that, for any $\epsilon > 0$, with probability 1 there exists an n' such that

$$\sup_{\mathbf{x} \in \mathsf{C}} \left| \widehat{\theta}(\mathbf{x}; n; \mathbf{U}) - \theta(\mathbf{x}) \right| \leq \epsilon$$

for all $n \geq n'$. This is the essential condition.

Stochastic Constraints

Extended SO formulation:

$$\begin{aligned} &\min\ \theta(\mathbf{x}) \\ &\mathbf{x} \in \mathsf{C} \\ &c_\ell(\mathbf{x}) \geq q_\ell,\ \ell = 1, 2, \ldots, v \end{aligned}$$

where $\theta(\mathbf{x}), c_1(\mathbf{x}), c_2(\mathbf{x}), \ldots, c_v(\mathbf{x})$ must all be estimated.

Assume that for any scenario $\mathbf{x}$ we can observe simulation outputs $Y(\mathbf{x}), C^{(1)}(\mathbf{x}), C^{(2)}(\mathbf{x}), \ldots, C^{(v)}(\mathbf{x})$ such that

$$\begin{aligned} \mathrm{E}\left(Y(\mathbf{x})\right) &= \theta(\mathbf{x}) \\ \mathrm{E}\left(C^{(\ell)}(\mathbf{x})\right) &= c_\ell(\mathbf{x}),\ \ell = 1, 2, \ldots, v \end{aligned}$$

Examples

Call center: $\mathbf{x}$ is staffing schedule

$\theta(\mathbf{x})$ staffing cost

$c(\mathbf{x})$ is the fraction of calls answered within 2 minutes

$q = 0.92$

Manufacturing: $\mathbf{x}$ determines a plant layout and machine capacities

$\theta(\mathbf{x})$ work in process

$c(\mathbf{x})$ is throughput

$q = 5000$ parts/day

Feasibility checking

Suppose we just want to check if $\mathbf{x}$ is feasible using $\bar{C}(\mathbf{x})$.

$$\begin{aligned}\Pr\{\bar{C}(\mathbf{x}) \geq q\} &= \Pr\left\{\frac{\sqrt{n}\left(\bar{C}(\mathbf{x}) - c(x)\right)}{\sigma(\mathbf{x})} \geq \frac{\sqrt{n}\left(q - c(\mathbf{x})\right)}{\sigma(\mathbf{x})}\right\} \\ &\stackrel{n\to\infty}{\longrightarrow} \Pr\{\mathrm{N}(0,1) \geq \lambda\}\end{aligned}$$

There are three cases for λ:

1. If $\mathbf{x}$ is infeasible then $q - c(\mathbf{x}) > 0$ so $\lambda = \infty$.
2. If $\mathbf{x}$ is strictly feasible then $q - c(\mathbf{x}) < 0$ so $\lambda = -\infty$.
3. If $\mathbf{x}$ is tight then $q - c(\mathbf{x}) = 0$ so λ is 0. *This means that no matter how much simulation we do, we cannot determine with certainty if a constraint is tight.*

Compromise

As opposed to feasible-infeasible, we accept some sloppiness:

If $\mathrm{E}[C(\mathbf{x})] \geq q + \varepsilon$ it is **desirable** to declare $\mathbf{x}$ is feasible: $\mathbf{x} \in D$

If $q - \varepsilon \leq \mathrm{E}[C(\mathbf{x})] < q + \varepsilon$ it is **acceptable** to declare $\mathbf{x}$ is feasible or infeasible: $\mathbf{x} \in A$

If $\mathrm{E}[C(\mathbf{x})] < q - \varepsilon$ it is **unacceptable** to declare $\mathbf{x}$ is feasible: $\mathbf{x} \in U$

We would like $\Pr\{D \subset \mathcal{F} \subset (D \cup A)\} \geq 1 - \alpha$.

Set of solutions declared feasible

Constraint-guided search

A natural approach is to add the constraints to the objective as a penalty.

Let $i = 0, 1, 2, \ldots$ be the iteration index, and let $n_i(\mathbf{x})$ be the total number of replications obtained from scenario $\mathbf{x}$ through iteration i.

Define the estimated value of scenario $\mathbf{x}$ through iteration i to be

$$\widehat{\theta}(\mathbf{x}) = \bar{Y}(\mathbf{x}) + \beta_i \max\left\{0, q - \bar{C}(\mathbf{x})\right\}$$

where the averages include all observations.

Do we want $\beta_i \to \infty$?

Probability of a large penalty

Consider the probability of a penalty larger than, say, $\delta > 0$:

$$
\begin{aligned}
&\Pr\left\{\beta_i \max\left\{0, q - \bar{C}(\mathbf{x})\right\} > \delta\right\} \\
&\quad \geq \quad \Pr\left\{\beta_i \left(q - \bar{C}(\mathbf{x})\right) > \delta\right\} \\
&\quad = \quad \Pr\left\{\bar{C}(\mathbf{x}) - q < -\frac{\delta}{\beta_i}\right\} \\
&\quad = \quad \Pr\left\{\frac{\sqrt{n_i(\mathbf{x})}\left(\bar{C}(\mathbf{x}) - c(\mathbf{x})\right)}{\sigma} < \frac{\sqrt{n_i(\mathbf{x})}}{\sigma}\left(q - c(\mathbf{x}) - \frac{\delta}{\beta_i}\right)\right\} \\
&\quad \to \quad \Pr\left\{\mathrm{N}(0,1) < \frac{\sqrt{n_i(\mathbf{x})}}{\sigma}\left(q - c(\mathbf{x}) - \frac{\delta}{\beta_i}\right)\right\}
\end{aligned}
$$

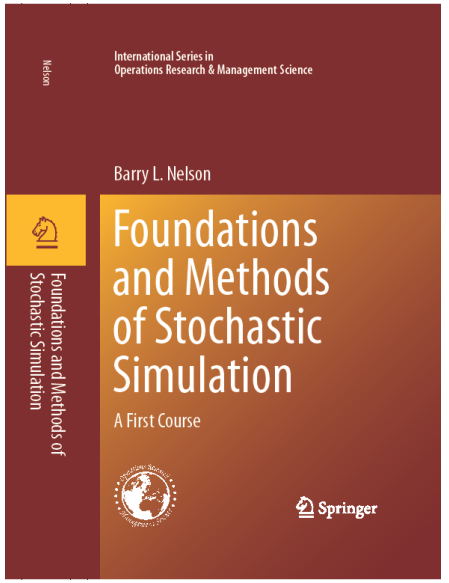
Analysis

1. If $\mathbf{x}$ is infeasible then we want this probability to get large quickly. Since $q - c(\mathbf{x}) > 0$ and $\delta > 0$, this means we would like β_i to become large quickly (big penalty), so $\beta_i \to \infty$ is fine.

2. If $\mathbf{x}$ is feasible but not tight, then we want this probability to get small quickly (no penalty). Since $q - c(\mathbf{x}) < 0$ and $\delta > 0$, we would like β_i to remain small and ideally go to 0.

3. If $\mathbf{x}$ is tight then we also want this probability to get small quickly (no penalty). Since $q - c(\mathbf{x}) = 0$ we *need* β_i to go to 0 to avoid a penalty.

Therefore, penalties should adapt to the observed (in)feasibility.



Chapter 9:
Simulation for Research

©Barry L. Nelson

Northwestern University

March 2013

Simulation as a research tool

- Simulation is often used as a research tool, even when the research has nothing to do with simulation.
  - Generating random test problems for optimization algorithms.
  - Testing the robustness of policies derived from simple stochastic models on complex systems.
  - Evaluating the accuracy of an approximation (e.g., queueing).
  - Establishing the properties of a new simulation-based estimator.
- Such studies are often done poorly, jeopardizing the publishability of otherwise good ideas.

Practitioner vs. Researcher

The *practitioner's experiment* solves a real problem, has a practical limit on time and effort, and leads to a decision.

Q: How much production capacity should we add?

The practitioner never knows for sure whether they got the "right answer."

The *researcher's experiment* is driven by a precisely formulated research question.

Q: Which optimization heuristic is better?

The researcher may repeatedly solve many problem instances, including ones for which they already know the answer.

Research experiments vs. illustrations

A *research experiment* should allow you to make statements about cases, scenarios or problems that you *did not* try based on ones you did try.

Ex: Optimization problems over a range of numbers of decision variables, numbers of constraints, tightness of the RHS, etc.

An *illustration* is a specific case, scenario or problem that helps us understand how a method is implemented and the results interpreted.

Ex: $M(t)/M/\infty$, $M/G/1$, *SAN and Asian option.*

Research Principle 1. *Completely random cases, scenarios or problems are not necessarily relevant ones. Instead, generate test cases, scenarios or problems that have features that are representative of the space of interest.*

Example: Hill and Reilly (2000) used randomly generated test problems to compare the speed and quality of solution of a heuristic and a convergent algorithm for solving two-dimensional knapsack problems:

$$\max \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \; i = 1, 2$$

$$x_j \in \{0, 1\}, \; j = 1, 2, \ldots, n$$

where $c_i > 0$ and $a_{ij} \geq 0$.

Standard test problems

Fix the number of items at n and set b_i to be a specified fraction of $\sum_{j=1}^{n} a_{ij}$.

Need to generate test values of c_i and a_{ij}.

Select uniform distributions for c_j, a_{1j} and a_{2j}, then randomly and independently generate the necessary values.

These "completely random test problems" are certainly random, and seem fair, right?

Why not the "standard approach?"

These problems are neither realistic nor interesting.

- Positive correlation between a_{ij} and c_j implies that if the jth item is valuable then it tends to be costly, which makes the problem realistic and hard.
- Negative correlation between a_{1j} and a_{2j} means an item that is inexpensive relative to b_1 tends to be expensive relative to budget b_2; this also makes the problem realistic and hard.
- Turns out correlation structure affects the heuristic and convergent algorithms differently, which might be important in selecting which one to use.

We would see few of these realistic and interesting problems if generated "completely randomly."

Including correlation in the design

Hill and Reilly systematically controled the strengths of the correlations in a designed experiment.

This allowed them to make stronger conclusions than they could by simply averaging over the space of "completely random" test problems.

They also made sure the heuristic and convergent algorithm saw *exactly the same test problems.* This illustrates...

Research Principle 2. *If you are comparing things, use common random numbers.*

Random & relevant test problems

Test problems are often needed in our research.

Random $\neq$ Relevant

1. Think carefully about the space of real problems over which you want to draw inference.
2. Create test problems that randomly cover this relevant space.
3. Apply each method to the *same* test problems.
4. Look at a lot of test problems, especially if the space is high dimensional.

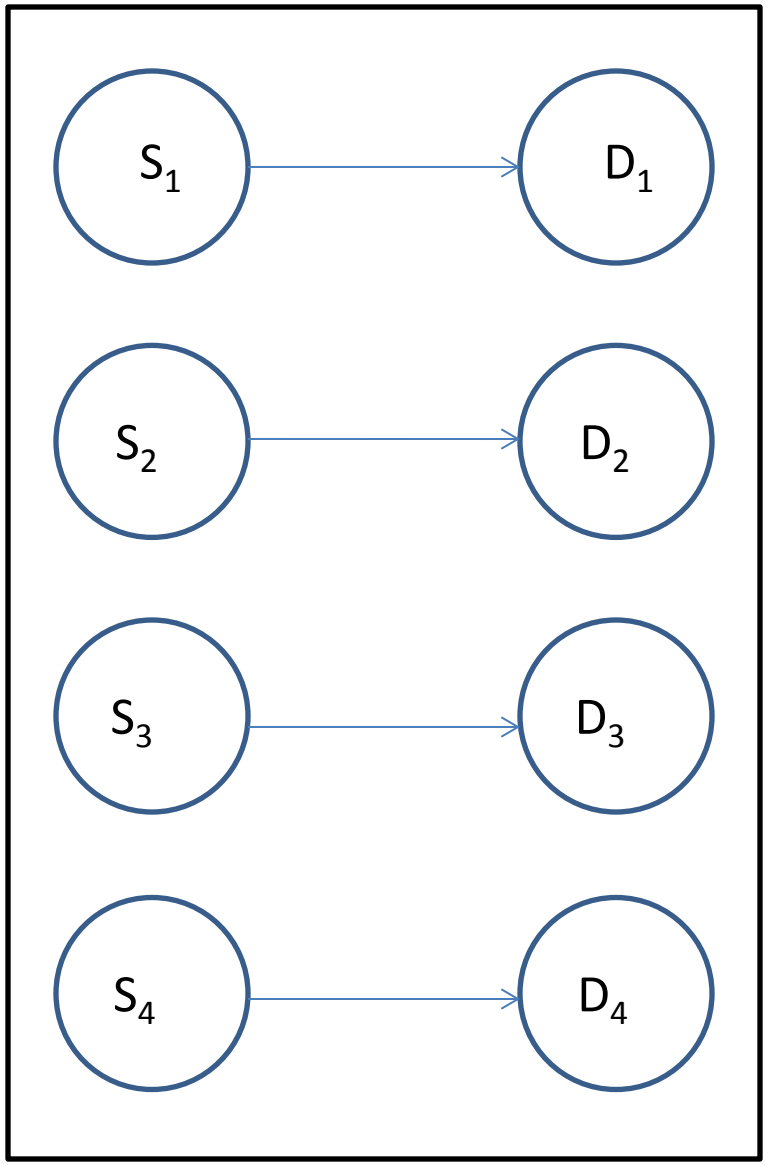
Research Principle 3. *Run a designed experiment that identifies and controls factors that might influence the outcome, both favorably and unfavorably, and that are actually encountered in practice.*

Iravani, Van Oyen and Sims (2005) wanted to create a simple measure of *structural flexibility (SF)*: the capability to satisfy multiple types of demand in the face of changing demand and resource capacity.

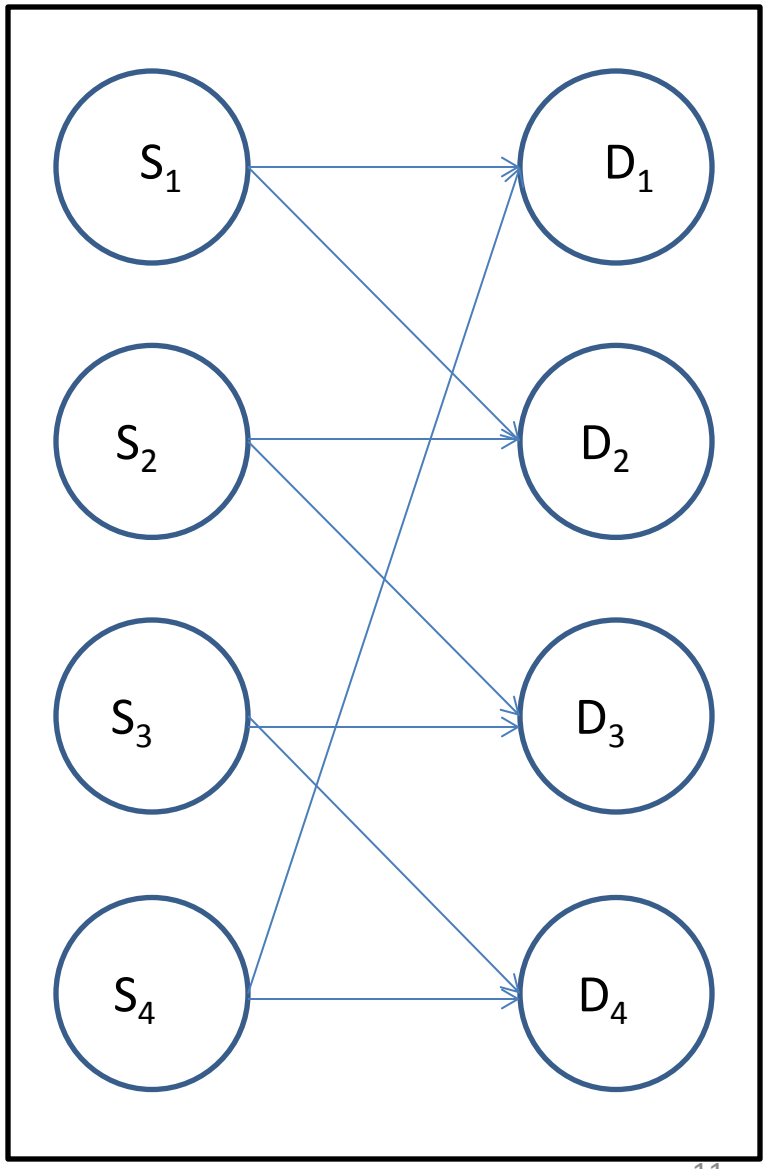
They represented a system as a graph from resources S_k to demand types D_j with an arc $S_k \rightarrow D_j$ if resource k can satisfy some demand of type j.

SF indices are functions of the number of paths through the network by which excess capacity for demand type i can be redirected to satisfy demand type j, and also the number of different resources that can satisfy demand of type j.

One factory to 1 car model



Two factories to each car model



Research question: Is a simple metric based only on structure really valid?

Approach: Choose pairs of system designs and compute their SF indices. Simulate the pairs to see if higher SF index predicts higher simulated productivity.

Key is designing an experiment that varies factors that are *not* inputs to the SF index, but do represent what occurs in the real world.

- Physical flow in the system (open or closed)
- Closeness of the system to capacity
- Uncertainty due to short-term variability and long-term shocks

Research Principle 4. *When there are no natural ranges of operability for your factors, link them so that they are large or small relative to each other.*

Example: Nelson, Swann, Goldsman and Song (2001) wanted to compare the computational efficiency of some ranking & selection (simulation optimization) procedures designed for up to $K = 500$ systems.

In addition to K, other factors that might affect efficiency include the true means $\mu_1, \ldots, \mu_K$, the true variances $\sigma_1^2, \ldots, \sigma_K^2$, the first-stage sample size n_0 and the indifference zone parameter δ.

How should these factors be set when there is no natural range of operability?

Linking factors

The key is to vary factors *relative to each other*.

- Anchor the means at $\mu_1 = \delta$ (bigger better).

  SK: $\mu_2 = \cdots = \mu_K = 0$

  MDM: $\mu_i = \mu_1 - (i-1)\delta/\tau$ with $\tau = 1, 2, 3$

- Anchor the variances to a common σ^2 or to the means.

  Common: $\sigma_2^2 = \cdots = \sigma_K^2 = \sigma^2$ and $\sigma_1^2 = \rho\sigma^2$ with $\rho = 1/2, 2$.

  Unequal: $\sigma_i = |\mu_i - \delta| + 1$ or $\sigma_i = 1/(|\mu_i - \delta| + 1)$.

- Set $\delta = d\sigma_1/\sqrt{n_0}$ with $d = 1/2, 1, 2$.

If we set $\sigma_1^2 = 1$, then there are only two remaining factors K and n_0 for which we can set reasonable ranges.

To be able to control all of these factors, Nelson et al. used normal and lognormal distributions to generate the simulation output data.

This might cause them to miss something, since real simulations have more complicated output processes.

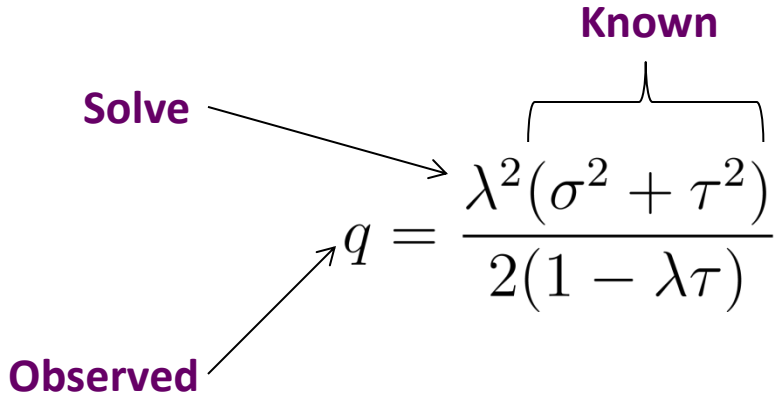
For this reason, we also want to include somewhat more realistic test cases, even though we have less control: Markovian queues, simple inventory models, AR(1), MA(1), SAN, Asian option, etc.

Research Principle 5. *Since it may not be possible to anticipate all important factors, include some realistic examples along with the controllable surrogate models.*

Research Principle 6. *If you reduce the standard error or confidence interval width enough, then the simulation estimate is effectively the same as the true value.*

Example: Whitt (1981) considered the situation where you can observe the congestion in a queue, and you know the service process, but you cannot observe the arrival process. Your goal is to predict what would happen to congestion if you replaced the current service process by a different one.

Simple example: $M/G/1$



Testing a useful approximation

Whitt wanted his method to work on queueing systems with very general, stationary arrival processes (meaning there could be dependence). Ex: $G/M/1$

He wanted to approximate the unknown arrival process G with a renewal process GI that yields a tractable queueing model. Ex: $GI/M/1$

A "good" GI means that the predicted results with a new service process should be right. Ex: $GI/M'/1 \approx G/M'/1$

Since he was using tractable queueing models as his approximations, he needed to use test cases that were *not* tractable (i.e., no known answer).

Using simulation to get "the truth"

Whitt considered his approximation good enough if

$$\frac{|q_{\text{approx}} - q_{\text{true}}|}{q_{\text{true}}} \leq 0.1$$

We do not know q_{true}, but simulation provides an estimate $\widehat{q}_{\text{true}} \pm H$.

The approximation is good enough if the relative error ratio holds *for every possible value* of q_{true} such that

$$\widehat{q}_{\text{true}} - H \leq q_{\text{true}} \leq \widehat{q}_{\text{true}} + H$$

We simulate until the $\pm H$ is small enough to know for sure.

Research Principle 7. *Measure and control the error in your research experiment using nested simulations.*

Example: A control-variate estimator $\widehat{\beta}_0$ is an alternative to the sample mean $\bar{Y}$ for estimating $\mu_Y = \mathrm{E}(Y)$ in a simulation.

As the number of replications goes to infinity, control-variate estimators are consistent and have smaller variance than $\bar{Y}$. In finite samples they may be biased and could have larger variance.

Nelson (1990) was interested in establishing properties of control-variate estimators, including how they perform in small samples.

Empirical evaluation

Do a designed experiment over relevant factors.

For each factor setting make m macroreplications—i.i.d. replications of the experiment—to estimate bias, variance and CI coverage. This is a nested experiment.

$$\begin{aligned}
\widehat{b} &= \frac{1}{m}\sum_{i=1}^{m}\widehat{\beta}_0^{(i)} - \mu_Y = \bar{\beta}_0 - \mu_Y \\
S^2_{\widehat{\beta}_0} &= \frac{1}{m-1}\sum_{i=1}^{m}\left(\widehat{\beta}_0^{(i)} - \bar{\beta}_0\right)^2 \\
\widehat{p} &= \frac{1}{m}\sum_{i=1}^{m} I\left\{\mu_Y \in \widehat{\beta}_0^{(i)} \pm t_{1-\alpha/2,n-2}\sqrt{\widehat{\Sigma}_{11}^{(i)}}\right\}.
\end{aligned}$$

m should not be chosen arbitrarily.

Number of macroreplications

Suppose that we set $1-\alpha = 0.95$, meaning 95% confidence intervals.

If the confidence interval has the desired coverage, then the standard error of $\widehat{p}$ is $\sqrt{(0.95)(0.05)/m}$.

To get approximately two decimal places of precision we need the number of macroreps m to satisfy

$$2\sqrt{\frac{(0.95)(0.05)}{m}} < 0.01$$

or $m \approx 1900$.

Macroreps can provide the "truth"

The control-variate variance estimator $\widehat{\Sigma}_{11}$ is expected to be biased, but we do not actually know what the true $\mathrm{Var}(\widehat{\beta}_0)$ is in small samples.

Note that $S^2_{\widehat{\beta}_0}$ is an unbiased estimator of $\mathrm{Var}(\widehat{\beta}_0)$, since it is based on i.i.d. observations of $\widehat{\beta}_0$.

An estimator of the bias of $\widehat{\Sigma}_{11}$ is therefore

$$\frac{1}{m}\sum_{i=1}^{m}\widehat{\Sigma}_{11}^{(i)} - S^2_{\widehat{\beta}_0}$$