CHAPTER 9

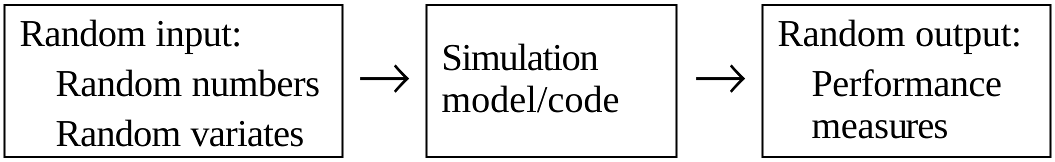
Output Data Analysis for a Single System

9.1 Introduction

Basic, most serious disadvantage of simulation:

With a stochastic simulation, don't get exact "answers" from run

Two different runs of same model ⇒ different numerical results



Random input:
Random numbers
Random variates

→

Simulation
model/code

→

Random output:
Performance
measures

Thus, the output performance measures:

Really observations from their probability distribution

Ask questions about this distribution:

Mean (expected value):	E(average WIP)
Variance:	Var(average WIP)
Probabilities:	P(average WIP > 250)
Quantiles:	What value of x is such that P(avg. WIP > x) ≤ 0.02?

Interpreting simulation output: *statistical analysis* of output data

Failure to recognize, deal with randomness in simulation output can lead to serious errors, misinterpretation, bad decisions

Also, must take care to use appropriate statistical methods, since simulation output data are usually nonstationary, autocorrelated, and non-normal, contrary to assumptions behind classical IID statistical methods

Enhanced computer power and speed is making it much easier to carry out appropriate simulation studies and analyze the output properly

The Statistical Nature of Simulation Output

Let Y_1, Y_2, ... be an output process from a *single* simulation run

Y_i = delay in queue of *i*th arriving customer

Y_i = production in *i*th hour in a factory

Y_i's are random variables that are generally neither independent nor identically distributed (nor normally distributed), so classical IID normal-theory statistical methods don't apply *directly* to the Y_i's

Let $y_{11}, y_{12}, \ldots, y_{1m}$ be a *realization* of the random variables $Y_1, Y_2, \ldots, Y_m$ resulting from making a single simulation run of length *m* observations, using a particular stream of underlying U(0, 1) random numbers.

If we use a separate stream of random numbers for another simulation run of this same length, we get a realization $y_{21}, y_{22}, \ldots, y_{2m}$ that is independent of, but identically distributed to, $y_{11}, y_{12}, \ldots, y_{1m}$

Make *n* such independent runs, each using "fresh" random numbers, to get

$$\begin{array}{cccc} y_{11}, & y_{12}, & \ldots, & y_{1m} \\ y_{21}, & y_{22}, & \ldots, & y_{2m} \\ \vdots & & & \\ y_{n1}, & y_{n2}, & \ldots, & y_{nm} \end{array}$$

Within a row: not IID

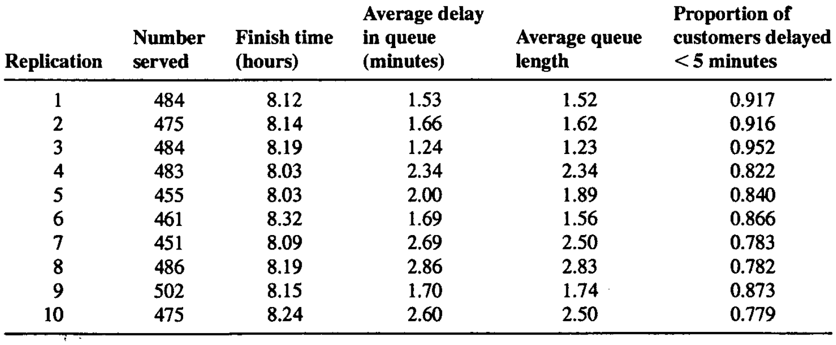
Across the *i*th column: IID realizations of the r.v. Y_i (but still not necessarly normally distributed)

Can compute a summary measure within a run, and then the summary measures across the runs are IID (but still not necessarily normally distributed)

Bank with 5 tellers, one FIFO queue, open 9am-5pm, flush out before stopping

Interarrivals ~ expo (mean = 1 min.), service times ~ expo (mean = 4 min.)

Summary measures from 10 runs (replications):



Replication	Number served	Finish time (hours)	Average delay in queue (minutes)	Average queue length	Proportion of customers delayed < 5 minutes
1	484	8.12	1.53	1.52	0.917
2	475	8.14	1.66	1.62	0.916
3	484	8.19	1.24	1.23	0.952
4	483	8.03	2.34	2.34	0.822
5	455	8.03	2.00	1.89	0.840
6	461	8.32	1.69	1.56	0.866
7	451	8.09	2.69	2.50	0.783
8	486	8.19	2.86	2.83	0.782
9	502	8.15	1.70	1.74	0.873
10	475	8.24	2.60	2.50	0.779

Clearly, there's variation across runs; need appropriate statistical analysis

Types of Output Performance Measures

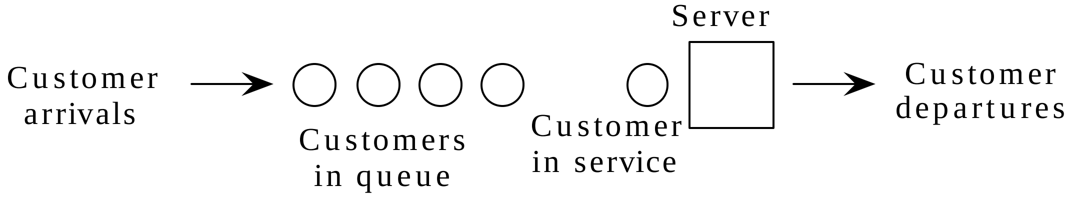
What do you want to know about the system?

- Average time in system
- Worst (longest) time in system
- Average, worst time in queue(s)
- Average, worst, best number of “good” pieces produced per day
- Variability (standard deviation, range) of number of parts produced per day
- Average, maximum number of parts in the system (WIP)
- Average, maximum length of queue(s)
- Proportion of time a machine is down, up and busy, up and idle

Ask the same questions of the model/code

Think ahead: Asking for additional output performance measures can change how the simulation code is written, or even how the system is modeled

Simple queueing model:



Want: Average number of customers in queue
Proportion of time server is busy

Maybe: Average time customers spend in queue

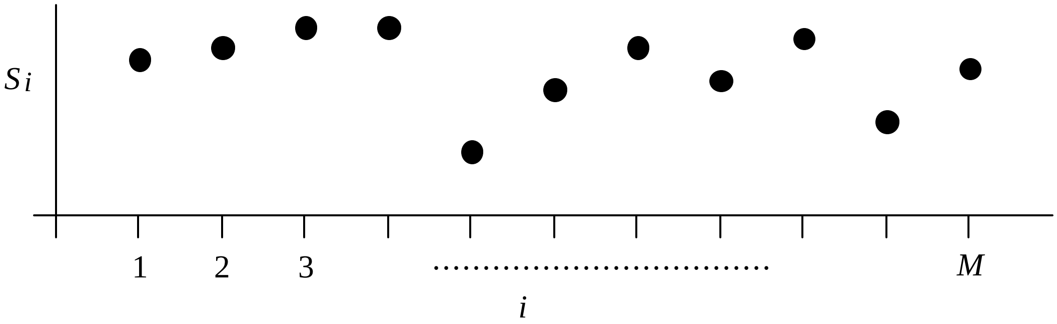
Question: How does wanting the average time in queue affect how the model is coded, data structures, etc.?

As simulation progresses through time, there are typically two kinds of *processes* that are observed:

Discrete-time process: There is a natural "first" observation, "second" observation, etc.—but can only observe them when they "happen"

S_i = time in system of ith part produced, $i \in \{1, 2, ...\}$

Suppose there are M parts produced during the simulation



Typical discrete-time output performance measures:

Average time in system: $\bar{S}(M) = \dfrac{\sum_{i=1}^{M} S_i}{M}$

Maximum time in system: $S^*(M) = \max_{i=1,2,\ldots,M} S_i$

Proportion of parts that were in system more than 60 minutes:

$$P_{60}(M) = \frac{\sum_{i=1}^{M} I_{(60,\infty)}(S_i)}{M}, \text{ where } I_{(60,\infty)}(S_i) = \begin{cases} 1 & \text{if } S_i > 60 \\ 0 & \text{if } S_i \le 60 \end{cases}$$

Other examples of discrete-time processes:

D_i = delay of ith customer in queue

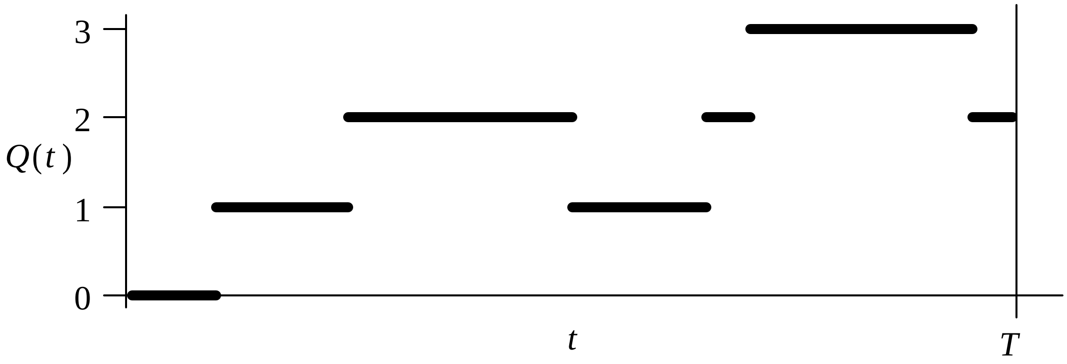
Y_i = throughput (production) during ith hour

B_i = 1 if caller i gets a busy signal, 0 otherwise

Continuous-time process: Can jump into system at any point in time (real, continuous time) and take a “snapshot” of something — there is no natural “first” or “second” observation

$Q(t)$ = number of parts in a particular queue at time $t \in [0, \infty)$

Run simulation for T units of simulated time



Typical continuous-time output performance measures:

Time-average length of queue: $\overline{Q}(T) = \dfrac{\int_0^T Q(t)\,dt}{T}$

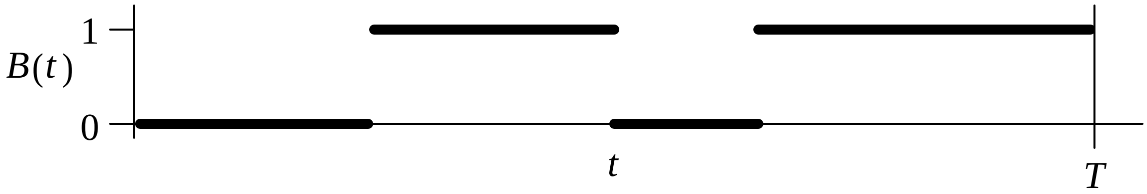
Maximum length of queue: $Q^*(T) = \max\limits_{0 \le t \le T} Q(t)$

Proportion of time that there were more than two in the queue:

$$P_2(T) = \frac{\int_0^T I_{(2,\infty)}(Q(t))\,dt}{T}$$

Another important kind of continuous-time statistic: *utilizations*

$$\text{Let } B(t) = \begin{cases} 1 & \text{if server is busy at time } t \\ 0 & \text{if server is idle at time } t \end{cases}$$



Server utilization (proportion of time busy): $U(T) = \dfrac{\int_0^T B(t)\,dt}{T}$

Other examples of continuous-time processes:

N(t) = number of parts in shop at time *t* (WIP)

$D(t)$ = number of machines down at time t

Typically, we want to observe several (maybe lots of) different performance measures from the same system/model

Usually low additional cost/hassle to do so, can always ignore later

But *not* getting a particular output measure could imply rerunning

Difficulty in statistical analysis of output with several performance measures:

May want to make several simultaneous estimates, statements

Be careful how this is done, what is said

Multiple-comparisons problem in statistics literature (more in Sec. 9.7)

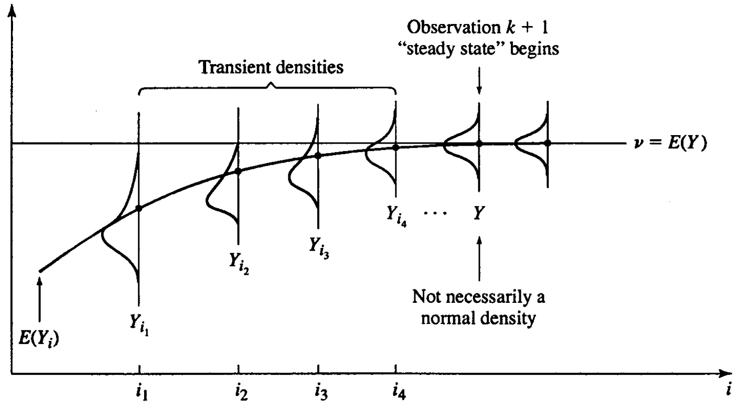
9.2 Transient and Steady-State Behavior of a Stochastic Process

Output process (discrete-time) Y_1, Y_2, ...

Let $F_i(y \mid I) = P(Y_i \le y \mid I)$ be the *transient* (cumulative) *distribution* of the process at (discrete) time i

In general, F_i depends on both the time i and the initial condition I

Corresponding transient density functions:



If there is a CDF $F(y)$ such that $F_i(y \mid I) \to F(y)$ as $i \to \infty$ for all y and for all initial conditions I, then $F(y)$ is the steady-state distribution of the output process

$F(y)$ may or may not exist

$F(y)$ must be independent of the initial conditions — same for all I

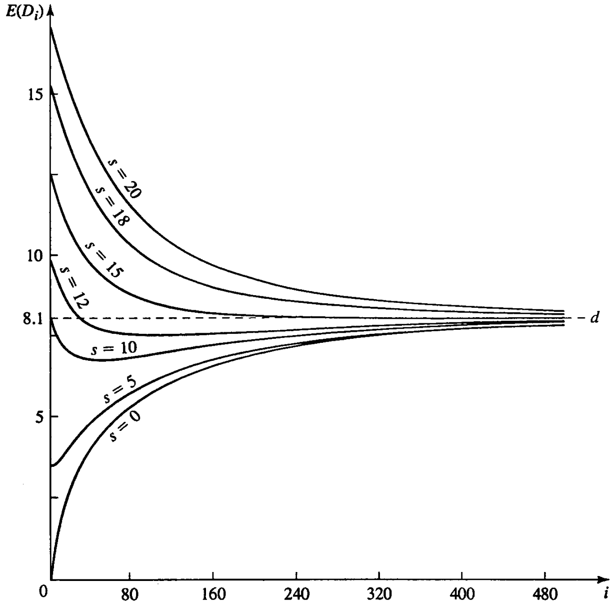
Roughly speaking, if there is a time index k such that for $i > k$ $F_i(y \mid I) \approx F(y)$ in some sense, then we say that the process is "in steady state" after time k

Even though the distribution of the Y_i's after time k is not appreciably changing, observations on the Y_i's could still have large variance and thus "bounce around" a lot — they're just not systematically trending any more

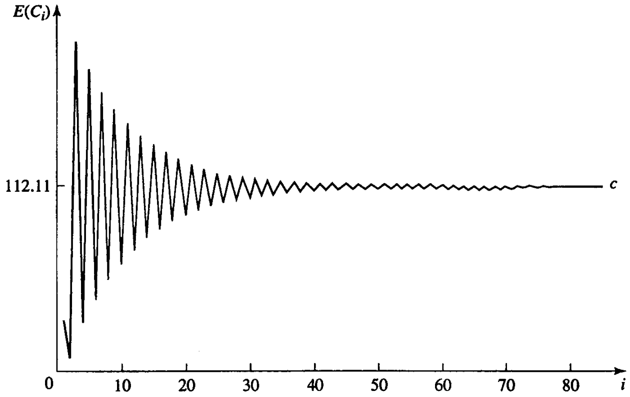
Even in steady state, the Y_i's are generally not independent, and could be heavily (auto)correlated

Steady-state distribution does not depend on initial conditions, but the nature and rate of convergence of the transient distributions can depend heavily on the initial conditions

M/*M*/1 queue, *E*(delay in queue), different number of customers *s* present initially:



Inventory system, *E*(cost in month *i*):



9.3 Types of Simulations with Regard to Output Analysis

Terminating: Parameters to be estimated are defined relative to specific initial and stopping conditions that are part of the model

There is a "natural" and realistic way to model both the initial and stopping conditions

Output performance measures generally depend on both the initial and stopping conditions

Nonterminating: There is no natural and realistic event that terminates the model

Interested in "long-run" behavior characteristic of "normal" operation

If the performance measure of interest is a characteristic of a steady-state distribution of the process, it is a *steady-state* parameter of the model

Theoretically, does not depend on initial conditions

Practically, must ensure that run is long enough so that initial-condition effects have dissipated

Not all nonterminating systems are steady-state: there could be a periodic "cycle" in the long run, giving rise to *steady-state cycle* parameters

Focus on terminating vs. steady-state simulations:

Which is appropriate?

- Depends on goals of the study
- Statistical analysis for terminating simulations is a *lot* easier

Is steady-state relevant at all? Maybe:

- 24 hours/day, "lights-out" manufacturing
- Global telecommunications
- Design conservatively for peak load of infinite duration

Some examples:

Physical model	Terminating estimand	Steady-state estimand
Single-server queue	Expected average delay in queue of first 25 customers, given empty-and-idle initial conditions	Long-run expected delay in queue of a customer
Manufacturing system	Expected daily production, given some number of workpieces in process initially	Expected long-run daily production
Reliability system	Expected lifetime, or probability that it lasts at least a year, given all components initially new, working	Probably not sensible
Battlefield model	Probability that attacking force loses half its strength before defending force loses half its strength	Probably not sensible

9.4 Statistical Analysis for Terminating Simulations

Make n IID replications of a terminating simulation

Same initial conditions for each replication

Same terminating event for each replication

Separate random numbers for each replication

Let X_j be a summary measure of interest from the jth replication
e.g., X_j = the average delay in queue of all customers in the jth replication

Then $X_1, X_2, \ldots, X_n$ are IID random variables, can apply classical statistical analysis to them

Rely on central limit theorem to justify normality assumption even though it's generally not true

So basic strategy is replication of the whole simulation some number n of times

One simulation run is a *sample of size one* (not worth much statistically)

What About Classical Statistics?

Classical statistical methods don’t work directly within a simulation run, due to *autocorrelation* usually present

Example: Delays in a queue of individual jobs: $D_1, D_2, D_3, \ldots, D_m$

Want to estimate m = E(average delay of the m jobs)

Sample mean $\overline{D}(m) = \sum_{i=1}^{m} D_i \Big/ m$ is an unbiased estimator for m

Need to estimate Var($\overline{D}(m)$) for confidence intervals on m, test hypotheses like H_0: $m = m_0$

But “sample variance” $\sum_{i=1}^{m} \left(D_i - \overline{D}(m)\right)^2 \Big/ \left[m(m-1)\right]$ may be severely biased for Var($\overline{D}(m)$)

Reason:

Corr(D_i, D_{i+l}) ≠ 0, in general

Unbiasedness of variance estimators follows from independence of data, which is *not true* within a simulation

Usual situation:

Positive autocorrelation: Corr(D_i, D_{i+l}) > 0

Causes $E\left\{\sum_{i=1}^{m} \left(D_i - \overline{D}(m)\right)^2 \Big/ \left[m(m-1)\right]\right\}$ < Var($\overline{D}(m)$) — maybe *far* too small

Intuition:

D_{i+1} is pretty much the same as D_i

D_i’s are more stable than if they were independent

Their sample variance is understated

Thus, must take care to estimate variances properly: understating variances

Have too much faith in our point estimates

Believe our simulation results too much

9.4.1 Estimating Means

Want: Estimate of some parameter m of the process

Often (not always): $m = E$(something)

$m = E$(average delay in queue of m customers)

$m = E$(time-average WIP)

$m = E$(machine utilization)

$m = P$(average delay > 5 minutes) $= E[\mathrm{I}_{(5,\infty)}$(average delay)]

Point estimate: $\hat{m} = 12.3$

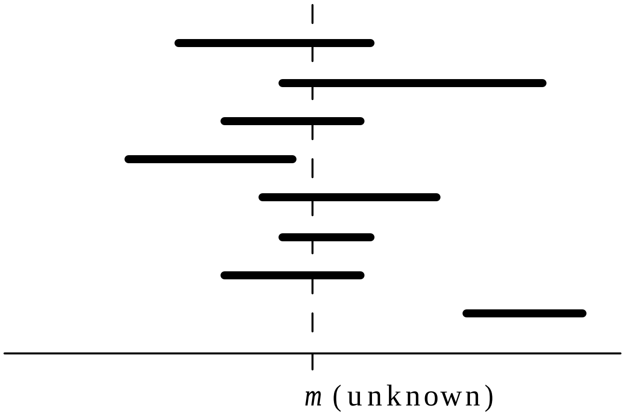
How close is $\hat{m}$ to the true unknown value of m?

Customary, useful method for assessing precision of estimator: *confidence interval* for m

Pick *confidence level* $1 - a$ (typically 0.90, 0.95, etc.)

Use simulation output to construct an interval $[A, B]$ that covers m with probability $1 - a$

Interpretation: $100(1 - a)$% of the time, the interval formed in this way will cover m



Wrong interpretation: "I'm 95% sure that m is between 9.4 and 11.1"

Common approach to statistical analysis of simulation output:

Can’t do “classical” (IID, unbiased) statistics *within* a simulation run

Try to modify setup, design, to get back to classical statistics

In terminating simulations, this is conceptually easy:

Make n independent *replications* of the *whole simulation*

Let X_j be the performance measure from the jth replication

X_j = average of the delays in queue

X_j = time-average WIP

X_j = utilization of a bottleneck machine

Then X_1, X_2, ..., X_n are IID and unbiased for $m = E(X_j)$

Apply classical statistics to X_j’s, *not* to observations *within* a run

Approximate 100(1 – a)% confidence interval for m:

$$\overline{X}(n) = \frac{\sum_{j=1}^{n} X_j}{n}$$ is an unbiased estimator of m

$$S^2(n) = \frac{\sum (X_j - \overline{X}(n))^2}{n-1}$$ is an unbiased estimator of Var(X_j)

$$\overline{X}(n) \pm t_{n-1,1-a/2} \frac{S(n)}{\sqrt{n}}$$ covers m with approximate probability 1 – a

($t_{n-1,1-a/2}$ = point below which is area (probability) 1–a/2 in Student’s t distribution with n – 1 d.f.)

Most important point:

The “basic ingredients” to the statistical analysis are the performance measures from the different, independent replications

One *whole* simulation run = a “sample” of size *one* (not worth much)

Example: n = 10 replications of single-server queue

X_j = average delay in queue from jth replication

X_j's: 2.02, 0.73, 3.20, 6.23, 1.76, 0.47, 3.89, 5.45, 1.44, 1.23

Want 90% confidence interval, i.e., a = 0.10

$\overline{X}(10) = 2.64$, $S^2(10) = 3.96$, $t_{9,\, 0.95} = 1.833$

Approximate 90% confidence interval is 2.64 ± 1.15, or [1.49, 3.79]

Other examples in text:

Inventory model

Estimation of proportions

Why “approximate” 90% confidence interval?

Assumes X_j’s are normally distributed — never true, but does it matter?

Central-limit theorem:

As n (number of replications) grows, coverage probability → $1 - a$

Robustness study with this model:

Know $m = 2.12$ from queueing theory

Pick an n (like $n = 10$)

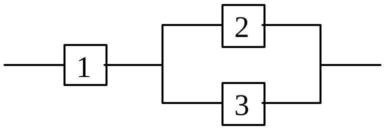
Make 500 “90%” confidence intervals (total no. runs = 10 × 500)

Count % of these 500 intervals covering $m = 2.12$:

n	Estimated coverage (want 90%)
5	88%
10	86%
20	89%
40	92%

Bad news: actual coverages can depend (a lot) on the model

Reliability model:



Components fail independently

Times T_i to failure in components ~ Weibull($a = 0.5$, $b = 1.0$)

Time to system failure = $T = \min(T_1, \max(T_2, T_3))$

$m = E$(time to system failure) = 0.78

n	Estimated coverage (want 90%)
5	71%
10	75%
20	80%
40	84%

Conventional wisdom: If X_j’s are averages of something (discrete- or continuous-time averages), their distribution tends to be not too asymmetric, and this confidence-interval method usually has reasonably good coverage

Obtaining a Specified Precision

If the number n of replication is chosen too small, the confidence intervals might too wide to be useful

M/*M*/1 example:

90% confidence interval from n = 10 replications: 2.64 ± 1.15, or [1.49, 3.79]

Half width (1.15) is 44% of point estimate (2.64)

Equivalently: 2.64 ± 44%, not very *precise*

Half-width expression: $d(a, n) = t_{n-1,1-a/2} \dfrac{S(n)}{\sqrt{n}}$

To decrease: a ↑: undesirable, since a = probability of missing

$S(n)$ ↓: estimates $\sqrt{\mathrm{Var}(X_j)}$, which is fixed (maybe ...)

n ↑: *more replications*

Sequential sampling: Increase n until $d(a, n)$ is "small enough"

Two versions of what “small enough” might mean (more details in text):

Absolute precision:

Specify $b > 0$, want n big enough so that $d(a, n) < b$

Requires at least some knowledge of context to set meaningful b

Relative precision:

Specify g ($0 < g < 1$), want n big enough so that $d(a, n)/\overline{X}(n) < g$

Need not know much about context to set meaningful g

Notes:

Above description leaves out a few technical details; see text

“Fixes” robustness issue: As b or $g \to 0$, coverage probability $\to 1 - a$

Can be dangerous for small b or g: Required n increases *quadratically* as b or g decrease

May be difficult to automate with a simulation language, depending on modeling constructs available, what automated statistical capabilities present, and what access the user has to internal software variables

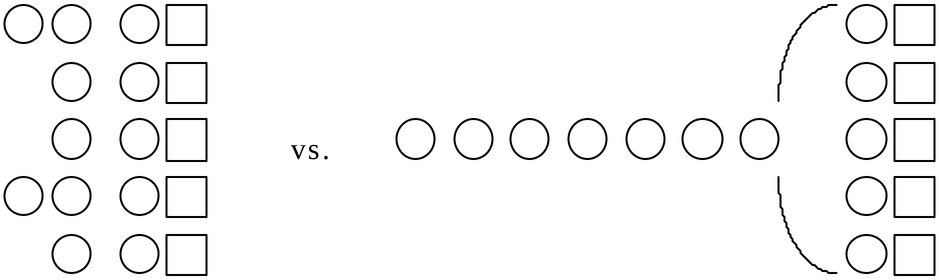
9.4.2 Estimating Other Measures of Performance

Sometimes can miss important system/model characteristics if we look *only* at averages

Other measures: Proportions, variances, quantiles

Proportions

Compare two operating policies for queueing system with five servers



	Estimates	
Performance measure	Five queues	Single queue
Average delay in queue	5.57 minutes	5.57 minutes
Average number in queue(s)	5.52	5.52
Number of delays ≥ 20 minutes	33	6

Variances (or Standard Deviations)

Interested in *process variability*

X_j = daily production of good items

Want to estimate $\sqrt{\text{Var}(X_j)}$

Make n replications, compute $S(n)$ as before

Confidence interval on $\sqrt{\text{Var}(X_j)}$: use chi-square distribution

Quantiles

Inventory system, X_j = maximum inventory level during the horizon

Want to determine storage capacity that is sufficient with probability 0.98

Want to find x such that $P(X_j \leq x) = 0.98$

One approach (more details in text):

Make n replications, observe $X_1, X_2, ..., X_n$

Sort the X_j's into increasing order

Estimate x to be a value below which are 98% of the X_j's

9.4.3 Choosing Initial Conditions

For terminating simulations, the initial conditions can affect the output performance measure, so the simulations should be initialized appropriately

Example: Want to estimate expected average delay in queue of bank customers who arrive and complete their delay between noon and 1:00pm

Bank is likely to be crowded already at noon, so starting empty and idle at noon will probably bias the results low

Two possible remedies:

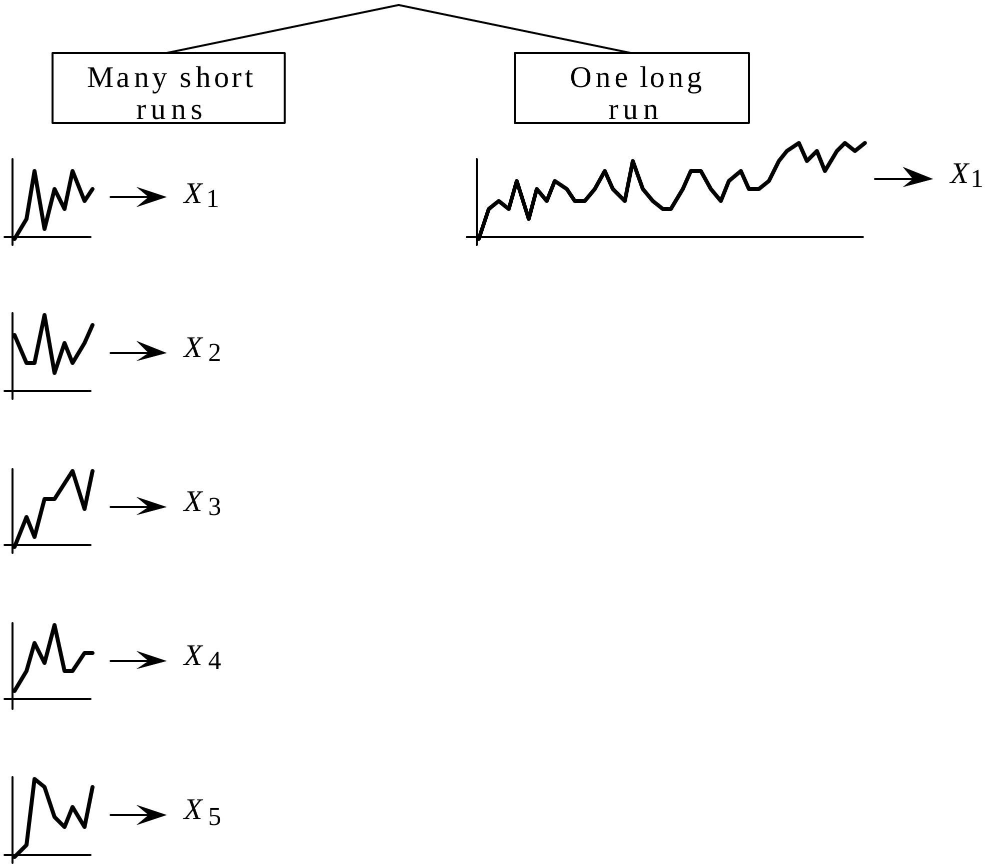
> If bank actually opens at 9:00am, start the simulation empty and idle, let it run for 3 simulated hours, clear the statistical accumulators, and observe statistics for the next simulated hour
>
> Take data in the field on number of customers present at noon, fit a (discrete) distribution to it, and draw from this distribution to initialize the simulation at time 0 = noon. Draw independently from this distribution to initialize multiple replications.
>
> > Note: This could be difficult in simulation software, depending on the modeling constructs available

9.5 Statistical Analysis for Steady-State Parameters

Much more difficult problem than analysis for terminating simulations
Want to estimate (e.g. for discrete- and continuous-time processes)

$$n = \begin{cases} \lim\limits_{i\to\infty} E(D_i) & D_i = \text{delay in queue of } i\text{th customer} \\ \lim\limits_{t\to\infty} E(Q(t)) & Q(t) = \text{number in queue at time } t \end{cases}$$

Basic question for designing runs:



	Many short runs	One long run
Good	Simple (same as terminating) Get IID data	*Less* point-estimator bias No restarts
Bad	Point-estimator bias (initial transient)	“Sample” of size 1 Hard to get variance estimate

9.5.1 The Problem of the Initial Transient

If steady-state is the goal, initial conditions will generally bias the results of the simulation for some initial period of time

Most common technique is to *warm up* the model, also called *initial-data deletion*

Identify index l (for discrete-time processes) or time t_0 (for continuous-time processes) beyond which the output appears not to be drifting any more

- Clear statistical accumulators at that time
- Start over with data collection, and “count” only data observed past that point

After warmup period, observations will still have variance, so will bounce around — they are just not “trending” any more

Facility for doing this in most simulation software (but the user must specify the warmup period)

Challenge — identifying a warmup period that is long enough, yet no so long as to be excessively wasteful of data — see text for details and examples

- Some statistical-analysis tests have been devised
- Most practical (and widely-used) method is to make plots, perhaps averaged across and within replications to dampen the noise, and “eyeball” a cutoff
- If there are multiple output processes, and if they disagree about what the appropriate warmup period is, a decision must be made whether to use different warmups for each process, or to use the same one for each process — which would have to be the maximum of the individual warmups, to be save, and so would be conservative for most of the output processes

A different approach: Try to find “smarter” initialization states or distributions that are “closer” to steady-state than something like “empty and idle”

- There has been some research on how to find such initial states/distributions
- “Priming” the model initially with entities may be tricky in simulation software

9.5.2 Replication/Deletion Approaches for Means

Assume that an appropriate warmup period has been determined

X_j = output measure on jth replication, collected only past warmup point

Proceed with statistical analysis exactly as in terminating case

Make independent replications, each warmed up

Compute mean, variance estimates across replications, confidence intervals

Advantages (compared to methods to be discussed below):

Simple — aside from warmup, the same as for terminating simulations

Get truly IID observations — important not only for variance estimation and confidence-interval construction, but also for more sophisticated statistical goals and techniques to be discussed in Chap. 10–12

Disadvantages:

No completely reliable method to identify an appropriate warmup

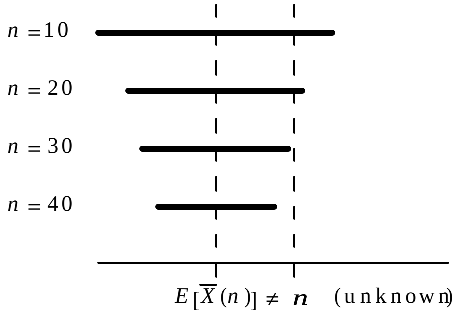
Too long ⇒ wasteful of data

Too short ⇒ point-estimator bias, which can have serious consequences, especially if used in concert with a sequential procedure:

$\overline{X}(n)$ is no longer an unbiased estimator of n

Confidence interval is centered in the –wrong” place— at $E[\,\overline{X}(n)\,] \neq n$

As n ↑, confidence interval shrinks down around the wrong point, causing coverage to *drop*



Work harder,
do worse (in coverage sense)

9.5.3 Other Approaches for Means

Make just one “replication,” presumably to ameliorate initial bias

Point estimator of n: average $\bar{Y}(m)$ of all the data Y_1, Y_2, ..., Y_m in *the* run

Problem: How to estimate Var($\bar{Y}(m)$), needed to get c.i.’s, etc.?

Know one way *not* to do this: $\sum_{i=1}^{m}\left(Y_i - \bar{Y}(m)\right)^2 \Big/ \left[m(m-1)\right]$

Several methods to estimate Var($\bar{Y}(m)$):

Batch means

Time-series models

Spectral analysis

Standardized time series

A different one-long-run approach (different point estimator):

Regenerative method

Two alternative modes of operation:

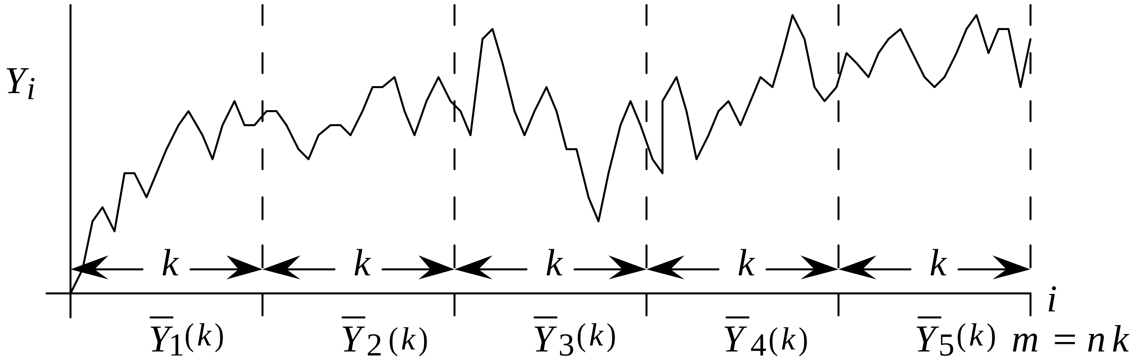
Fixed-sample size procedures — select run length m in advance, precision not controlled

Sequential procedures — prespecify precision, increase run length m as needed

Batch Means

Divide *the* run (of length m) into n contiguous “batches” of length k each ($m = nk$)

Let $\bar{Y}_j(k)$ be the (batch) mean of the jth batch of size k



Pretend: $\bar{Y}_j(k)$’s are IID, unbiased for n

Note: $\dfrac{\sum_{j=1}^{n} \bar{Y}_j(k)}{n} = \dfrac{\sum_{i=1}^{m} Y_i}{m} = \bar{Y}(m)$, i.e., mean of batch means is the “grand” mean

Form “sample variance” among the batch means, $S^2_{\bar{Y}}(n) = \dfrac{\sum_{j=1}^{n} \left(\bar{Y}_j(k) - \bar{Y}(m)\right)^2}{n-1}$

Approximate 100(1 – a)% confidence interval for n: $\bar{Y}(m) \pm t_{n-1,1-a/2} \dfrac{S_{\bar{Y}}(n)}{\sqrt{n}}$

Appeals of batch means:

- Simple (relatively)
- Often works fairly well (in terms of coverage)
- Automatically implemented in some simulation software

Issues with batch means (see text for more details and references):

- Choose batches big enough so that $\overline{Y}_j(k)$'s are approximately uncorrelated
- Otherwise, $S^2_{\overline{Y}}(n)$ can be biased (usually low) for Var($\overline{Y}_j(k)$), causing undercoverage
- How to choose batch size *k*? Equivalently, how many batches *n*?
  - Evidence: It may never pay to have more than n = 20 or 30 batches
  - Reason: Due to autocorrelation, splitting run into a larger number of smaller batches, while increasing degrees of freedom, degrades the quality (variability) of each individual batch

Generalizations of batch means

- Overlapping batch means
- Separated batches
- Weighting points within the batches

Sequential batch-means methods to control confidence-interval width

- Fix the number *n* of batches
- Increase overall run length *m*
- Increase batch size *k* proportionally to increase in *m*

Time-Series Models

Assume a relatively simple *statistical model* for the output process

Discrete-time output process $Y_1, Y_2, \ldots$ with steady-state mean n

As with batch means, use overall average $\bar{Y}(m)$ as point estimator of n

Examples of statistical models for output process:

Autoregressive of order 2 (AR(2)):

$Y_i = n + f_1(Y_{i-1} - n) + f_2(Y_{i-2} - n) + e_i$, e_i's IID N(0, s^2)

Autoregressive moving average of order (2, 1) (ARMA(2, 1)):

$Y_i = n + f_1(Y_{i-1} - n) + f_2(Y_{i-2} - n) + q_1 e_{i-1} + e_i$, e_i's IID N(0, s^2)

In general, for ARMA(p, q) model:

Use simulation output to *identify* (estimate) p and q — get $\hat{p}$ and $\hat{q}$ (several methods exist)

Estimate parameters via regression (least-squares) — ("fit the model"):

$$\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_{\hat{p}}$$
$$\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_{\hat{q}}$$
$$\hat{s}^2$$

Under this model, Var($\bar{Y}(m)$) = $g(f_1, f_2, \ldots, f_p, q_1, q_2, \ldots, q_q, s^2)$ (the function g is known but messy)

Estimate Var($\bar{Y}(m)$) by $g(\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_{\hat{p}}, \hat{q}_1, \hat{q}_2, \ldots, \hat{q}_{\hat{q}}, \hat{s}^2)$

Appeals of time-series models:

Physical intuition

Some theoretical support — most processes can be approximated by an ARMA(p, q) if we're willing to admit large p and q

Issues with time-series models:

Computation involved for the "fit"

Specifying degrees of freedom for variance estimator

Robustness questions — any better than batch means?

Spectral Analysis

Assume that output process is covariance-stationary:

Cov(Y_i, Y_{i+j}) = C_j, i.e., it depends only on j, and not on i

Fairly mild assumption, especially after some warmup

Then $\operatorname{Var}(\overline{Y}(m)) = \dfrac{C_0 + \sum_{j=1}^{m-1} (1 - j/m) C_j}{m}$

Estimate C_j, j = 1, 2, ..., m – 1 from output data:

$$\hat{C}_j = \frac{\sum_{i=1}^{m-j} \left(Y_i - \overline{Y}(m)\right)\left(Y_{i+j} - \overline{Y}(m)\right)}{m - j}$$

Plug $\hat{C}_j$'s into formula for Var($\overline{Y}(m)$) in place of corresponding C_j's

Appeals of spectral analysis:

Using an exact variance formula

Relationship to other methods (overlapping batch means)

Issues with spectral analysis:

For large m (as we'd expect), computationally burdensome to get all the $\hat{C}_j$'s — number of operations is about $m^2/2$

Remedy: Computational device — *Fast Fourier Transform*

For j near the end (m – 1), $\hat{C}_j$ will be based on only a few terms, so will itself be highly variable—in fact, for $j = m - 1$, $\hat{C}_{m-1} = \left(Y_1 - \overline{Y}(m)\right)\left(Y_m - \overline{Y}(m)\right)$, which is based on only one term in the sum

Remedy: Use different weights — *spectral window* — in sum for estimate of Var($\overline{Y}(m)$)

Regenerative Method

Different approach to gathering data — do *not* use $\bar{Y}(m)$ as point estimator for n

Assume output process is *regenerative*:

Can identify a sequence of random indices $1 \le B_1 < B_2 < \ldots$ such that:

Starting from each index B_j, the process from then on follows the same probability distribution as does the process from any other B_j

The process from index B_j on is independent of the process before B_j

This divides the process into IID *cycles* (or *tours*):

$$Y_{B_1}, Y_{B_1+1}, \ldots, Y_{B_2-1}$$
$$Y_{B_2}, Y_{B_2+1}, \ldots, Y_{B_3-1}$$
$$\vdots$$

Most familiar example:

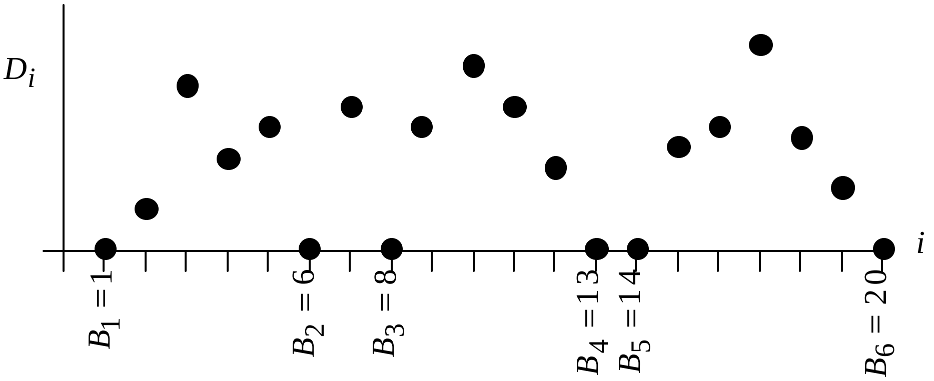
Single- or multiple-server queue

Any interarrival, service-time distributions

$Y_i = D_i$ = delay in queue of ith arriving customer

B_j = index of the jth customer who arrives to find the whole system empty of other customers and all servers idle

For single-server case, a customer begins a regeneration cycle if and only if delay in queue is zero (not true for multiple servers)



Comparable random variables defined on successive cycles are IID

$N_j = B_{j+1} - B_j$ = length of jth cycle

$Z_j = \sum_{i=B_j}^{B_{j+1}-1} Y_i$ = sum of the observations in the jth cycle

Can show (renewal reward theory): $\nu = E(Z_j)/E(N_j)$

Make a simulation run of n' cycles (*don't* stop in the middle of a cycle)

For $j = 1, 2, ..., n'$, let $\mathbf{U}_j = (Z_j, N_j)^T$; these are IID random *vectors*

Point estimator for ν: $\hat{\nu} = \frac{\bar{Z}(n')}{\bar{N}(n')}$, which is biased, but *strongly consistent*

Variance estimation, confidence interval:

Let $\mathbf{S} = \begin{bmatrix} s_{11} & s_{12} \\ s_{12} & s_{22} \end{bmatrix}$ be the covariance matrix of $\mathbf{U}_j$

Define $V_j = Z_j - n\ N_j$, which is unobservable, but has mean zero, variance $s_V^2 = s_{11} - 2n s_{12} + n^2 s_{22}$

Central-limit theorem applied to V_j's:

$$\frac{\overline{Z}(n') - n\ \overline{N}(n')}{\sqrt{s_V^2 / n'}} \xrightarrow{D} \text{N}(0, 1) \text{ as } n' \to \infty$$

Need to estimate s_V^2:

$$\text{Estimate } \mathbf{S} \text{ by } \begin{bmatrix} \hat{s}_{11}(n') & \hat{s}_{12}(n') \\ \hat{s}_{12}(n') & \hat{s}_{22}(n') \end{bmatrix} = \frac{\sum_{j=1}^{n'} \left[\mathbf{U}_j - \overline{\mathbf{U}}(n')\right]\left[\mathbf{U}_j - \overline{\mathbf{U}}(n')\right]^T}{n' - 1}$$

Let $s_V^2(n') = \hat{s}_{11}(n') - 2\hat{n}(n')\hat{s}_{12}(n') + \left[\hat{n}(n')\right]^2 \hat{s}_{22}(n')$, which is strongly consistent for s_V^2

Can replace s_V^2 in above CLT by $\hat{s}_V^2(n')$ (*continuous mapping thm.*):

$$\frac{\overline{Z}(n') - n\ \overline{N}(n')}{\sqrt{\hat{s}_V^2 / n'}} \xrightarrow{D} \text{N}(0, 1) \text{ as } n' \to \infty$$

Manipulate to get *asymptotically valid* 100(1 – a)% confidence interval for n:

$$\hat{n}(n') \pm \frac{z_{1-a/2}\sqrt{\hat{s}_V^2(n')/n'}}{\overline{N}(n')}$$

Appeals of regenerative method:

Firm mathematical foundation—asymptotic validity guarantees proper coverage probability $(1 - a)$ as $n' \to \infty$

Issues with regenerative method:

Underlying process must be regenerative — not universally true

Must identify regeneration points (proof), code their recognition into program, gather data differently

Asymptotic validity depends on having a lot of cycles — if cycles tend to be long (as they often do in complicated models) we may be able to observe only a few cycles, and asymptotic validity doesn't kick in

There are other ways to get a confidence interval (notably *jackknifing*; see text)

Sequential-sampling versions have been developed—keep simulating more *cycles* until the confidence interval is small enough

Standardized Time Series

Classical univariate statistics:

Take IID sample $X_1, X_2, ..., X_n$

Want to estimate $m = E(X_i)$

"Standardize" univariate data:

$$\frac{\overline{X}(n) - \text{unknown } m}{\text{Estimate of standard deviation of } \overline{X}(n)} \xrightarrow{D} \text{N}(0,1)$$

Basis for inference (confidence intervals, hypothesis tests, ...)

Observing a process (via simulation):

Observe the process $Y_1, Y_2, ..., Y_m$

Want to estimate $n = \lim_{i \to \infty} E(Y_i)$

"Standardize" process data:

$$\frac{Y_i - \text{unknown } n}{\text{Estimate of standard deviation of } Y_\text{i}} \xrightarrow{D} \textit{Brownian bridge} \text{ process}$$

(Brownian bridge process is fully understood, like N(0, 1)

Basis for inference (confidence intervals, hypothesis tests, ...)

Specifics:

Observe process $Y_1, Y_2, ..., Y_m$

Point estimator: $\bar{Y}(m)$, like batch means, time-series models, spectral analysis

Form *n* batches of size *k* each ($m = nk$), as for batch means

For large m, $\bar{Y}(m)$ is approximately normal with mean n and variance t^2/m where $t^2 = \lim_{m \to \infty} \mathrm{Var}(\bar{Y}(m))$

Let $$A = \frac{12}{k^3 - k}\sum_{j=1}^{n}\left[\sum_{s=1}^{k}\sum_{i=1}^{s}\left(\bar{Y}_j(k) - Y_{i+(j-1)k}\right)\right]^2$$

As $k \to \infty$, $A/t^2 \to c^2$ distribution with n d.f., and is independent of $\bar{Y}(m)$

So for large k, $$\frac{\left(\bar{Y}(m) - n\right)/\sqrt{t^2/m}}{\sqrt{\dfrac{A/t^2}{n}}} = \frac{\bar{Y}(m) - n}{\sqrt{A/(mn)}}$$

has an approximate *t* distribution with n d.f., so an asymptotically valid confidence interval for n is $\bar{Y}(m) \pm t_{n,1-a/2}\sqrt{A/(mn)}$

Appeals of standardized time series:

Firm mathematical foundation—asymptotically valid intervals

Assumptions are *much* weaker than for regenerative method

Relatively simple

Issues with standardized time series:

As with all asymptotic, methods, how long is long enough?

Above is called "area" approach — *A* represents *area* under the standardized Brownian bridge

Other approaches look instead at the *maximum* attained by the standardized Brownian bridge

Summary of Steady-State Estimation Procedures

Nothing will work well if computational budget is unduly limited

Batch means, spectral analysis, standardized time series display generally good performance (with respect to coverage probability, efficiency of data usage)

Simplicity might argue against spectral analysis

Sequential-sampling versions exist for most of the main methods, to control confidence-interval width

Performance measures other than means (variances, proportions, quantiles) have been investigated

It may be difficult to implement these methods in the context of some existing simulation software, though some software does allow for, and even builds in and automates, some of these methods

9.5.4 Estimating Other Measures of Performance

Means do not always provide the one and only appropriate measure of performance

Probabilities: For some set B, estimate $p = P(Y \in B)$, where Y is the steady-state random variable of the process $Y_1, Y_2, \ldots$

e.g., Y = delay of a message, B = (0, 5 minutes), so p is the probability that a message is delayed by less than 5 minutes

This is a special case of estimating means, if we define the indicator random variable $Z = \begin{Bmatrix} 1 & \text{if } Y \in B \\ 0 & \text{otherwise} \end{Bmatrix}$, then $p = P(Y \in B) = P(Z = 1) = 1 \times P(Z = 1) + 0 \times P(Z = 0) = E(Z)$

Thus, can use all the methods described above for means to estimate a proportion

Quantiles: The q-quantile y_q is the value such that $P(Y \le y_q) = q$

e.g., Y = delay of a message, $y_{0.75}$ is the value below which are 75% of the message delays

See text for details on methods for estimating quantiles based on order statistics, batch means, and regenerative methods

9.6 Statistical Analysis for Steady-State Cycle Parameters

To dichotomize simulations into terminating vs. steady-state is a bit of an oversimplification:

Manufacturing model, has 8-hour shifts

Simulating in detail what goes on within a shift

Performance might fluctuate widely within a shift

But what matters is the production (say) over the *whole* shift

Aggregate the data so that a “basic” observation Y_i is the production on the *i*th *shift*

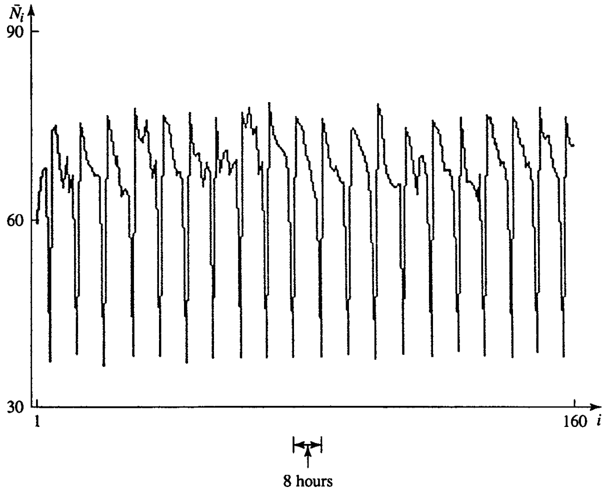
Steady-state *cycle* (e.g., cycle = shift) parameter: steady-state mean of process defined over a cycle as a “unit” of time

Use above steady-state methods, except applied to random variables defined over a cycle, rather than individually at their finest level of detail

Example:

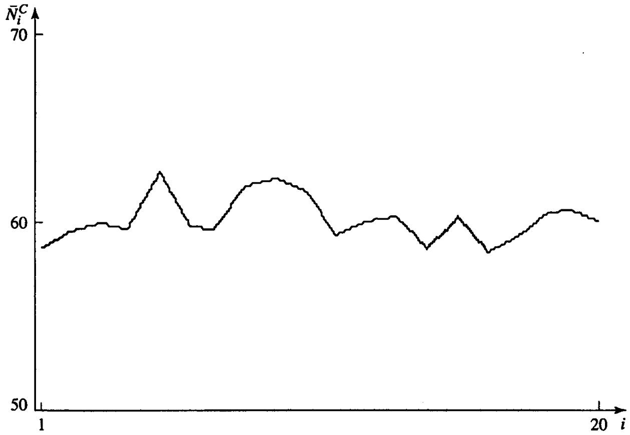
Production process, shift = 8 hours, lunch break 4 hours into each shift

N_i = production in hour i; would expect it to drop in 4th hour; plot averaged over 10 replications:



No steady-state

N_i^C = production in *shift i*; plot averaged over 10 replications:



Seems to have a steady-state; use earlier methods

9.7 Multiple Measures of Performance

Usually: Want to observe several performance measures from a large simulation

Average length of queue(s)

Maximum length of queue(s)

Utilization(s)

Throughput(s)

Difficulty:

Estimate each (expected) performance measure with a confidence interval

Any of the intervals could “miss” its expected performance measure

Must be careful about *overall* statements of coverage (i.e., that *all* intervals contain their expected performance measures *simultaneously*)

Sometimes called the problem of *multiple comparisons*

Have k output performance measures, want overall (*familywise*) probability of at least $1 - a$ that the confidence intervals for them *all* contain their target expected performance measures

For $s = 1, 2, ..., k$, suppose the confidence interval for performance measure s is at confidence level $1 - a_{\mathrm{S}}$

Then P(all intervals contain their respective performance measures) $\geq 1 - \sum_{s=1}^{k} a_s$

(*Bonferroni inequality*)

Thus, pick a_S’s so that $\sum_{s=1}^{k} a_s = a$

Could pick $a_S = a/k$ for all s, or pick a_S’s differently with smaller a_S’s for the more important performance measures

Obvious problem: For large k and reasonable overall a, the individual a_S’s could become tiny, making the corresponding confidence intervals enormous

Alternative Approach

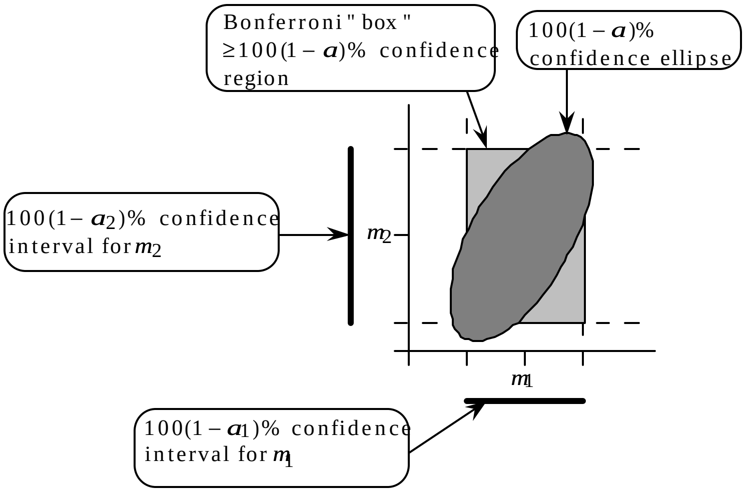
Use multivariate statistical methods to get a *joint* confidence *region* (not necessarily a rectangle) that covers expected-performance-measure *vector* with probability 1 – a

Example:

$k = 2$ performance measures m_1 and m_2

Bonferroni approach: Separate intervals for m_1 and m_2

Multivariate approach: Ellipse containing (m_1, m_2) with probability 1 – a



Bonferroni " box "
≥100(1 – a)% confidence region

100(1 – a)%
confidence ellipse

100(1 – a_2)% confidence
interval for m_2

100(1 – a_1)% confidence
interval for m_1

Specific multivariate methods:

- Multivariate batch means
- Multivariate spectral analysis
- Multivariate time-series methods

Appeal of multivariate methods:

- Smaller area (volume) with multivariate methods

Issues with multivariate methods:

- Complexity
- Practical interpretation

9.8 Time Plots of Important Variables

So far, have concentrated on performance measures over the course of the whole simulation run

- Averages
- Variances
- Extrema
- Proportions
- Quantiles

But these may mask important dynamic (within-run) behavior patterns

- Periodicities
- Explosions
- Learning

Ways to pick up such dynamic behavior

- Plot output processes (discrete or continuous)
- Animation