Analyzing Simulation Output: Batch Means Intervals

DCS 307: Simulation
Winter 2023

29 March 2023

Great Ideas in Simulation

1. Lehmer RNGs (linear congruential generators)

$$x_{i+1} = ax_i \bmod m$$

2. Probability integral transformation:

$$F(X) \sim U(0,1)$$

3. Algorithms for variate generation: inversion, acceptance/rejection, composition, special properties
4. Handling autocorrelation in constructing confidence intervals
5. Next-event simulation: e.g., use arrival to create next arrival
6. Agent-based simulation

Great Ideas in Simulation

1. Lehmer RNGs (linear congruential generators)

$$x_{i+1} = ax_i \bmod m$$

2. Probability integral transformation:

$$F(X) \sim U(0,1)$$

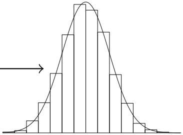
3. Algorithms for variate generation: inversion, acceptance/rejection, composition, special properties
4. Handling autocorrelation in constructing confidence intervals
5. Next-event simulation: e.g., use arrival to create next arrival
6. Agent-based simulation

Recap: Central Limit Theorem

- Choose *any* of the distributions encountered so far
- Generate a sample of $n > 1$ variates (mean μ, sd σ)
- Generate $j > 1$ n-point samples, compute $\bar{x}$ for each

$$\underbrace{x_1, x_2, \ldots, x_n}_{\bar{x}_1}, \underbrace{x_{n+1}, x_{n+2}, \ldots, x_{2n}}_{\bar{x}_2}, \underbrace{x_{2n+1}, x_{2n+2}, \ldots, x_{3n}}_{\bar{x}_3}, x_{3n+1}, \ldots$$

$\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots \longrightarrow$ `hist` $\longrightarrow$



- For large n, histogram density approximates *normal*$(\mu, \sigma/\sqrt{n})$

Recap (Cont.)

- CLT: mean of *iid* sequence of RVs tends to *normal*

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \rightarrow \quad \mathit{normal}\,(\mu, \sigma/\sqrt{n})$$

- The x_i correspond to simulation estimates, i.e., samples
  - e.g., average wait, average queue length, etc.

- We want the (unknown) μ from $\mathit{normal}\,(\mu, \sigma/\sqrt{n})$
  - $\bar{x}$ is an estimate of μ

- Standardized sample means tend to *normal*(0,1)
- Using t-statistic, approximates *Student*(n-1)

$$t = \frac{\bar{x}_j - \mu}{s/\sqrt{n-1}}$$

  0-13-142917-5

Recap (Cont.)

- CLT: mean of *iid* sequence of RVs tends to *normal*

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \rightarrow \quad \textit{normal}\,(\mu, \sigma/\sqrt{n})$$

- The x_i correspond to simulation estimates, i.e., samples
  - e.g., average wait, average queue length, etc.

- We want the (unknown) μ from *normal* $(\mu, \sigma/\sqrt{n})$
  - $\bar{x}$ is an estimate of μ

- Standardized sample means tend to *normal* (0,1)
- Using t-statistic, approximates *Student*(n-1)

$$t = \frac{\bar{x}_j - \mu}{s/\sqrt{n-1}}$$

Recap (Cont.)

- CLT: mean of *iid* sequence of RVs tends to *normal*

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \rightarrow \quad normal\,(\mu, \sigma/\sqrt{n})$$

- The x_i correspond to simulation estimates, i.e., samples
  - e.g., average wait, average queue length, etc.

- We want the (unknown) μ from *normal* $(\mu, \sigma/\sqrt{n})$
  - $\bar{x}$ is an estimate of μ

- Standardized sample means tend to *normal* (0,1)
- Using t-statistic, approximates *Student*(n-1)

$$t = \frac{\bar{x}_j - \mu}{s/\sqrt{n-1}}$$

Interval Estimation Algorithm

Algorithm

To calculate an *interval estimate* for the unknown mean μ of the *population* from which a random sample $x_1, x_2, x_3, \ldots, x_n$ was drawn:

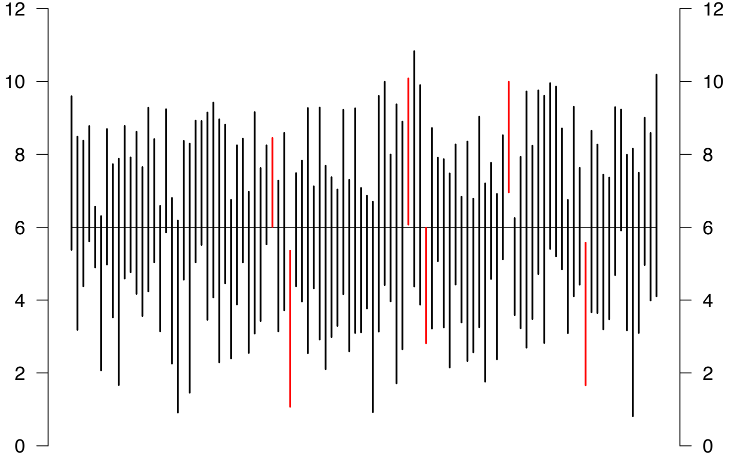
- Pick a level of confidence $1 - \alpha$
  - typically $\alpha = 0.05$ (95% confidence)
- Calculate the sample mean $\bar{x}$ and standard deviation s
- Calculate the critical value $t^* = \texttt{qt}(1 - \alpha/2, n - 1)$
- Calculate the interval endpoints

$$\bar{x} \pm \frac{t^* s}{\sqrt{n-1}}$$

If n is sufficiently large, then you are $(1 - \alpha) \times 100\%$ confident that the mean μ lies within the interval. The midpoint of the interval is $\bar{x}$.

Example

- 100 samples of size $n = 9$ drawn from *normal*$(6, 3)$ population
- For each sample, construct an exact 95% confidence interval
- 94 intervals cover $\mu = 6$
- 3 missed high (red), 3 missed low (red)



- Recall that `ssq()` service times are *exponential*$(\lambda = 10/9)$
- Use `t.test()` approach to produce 95% confidence intervals

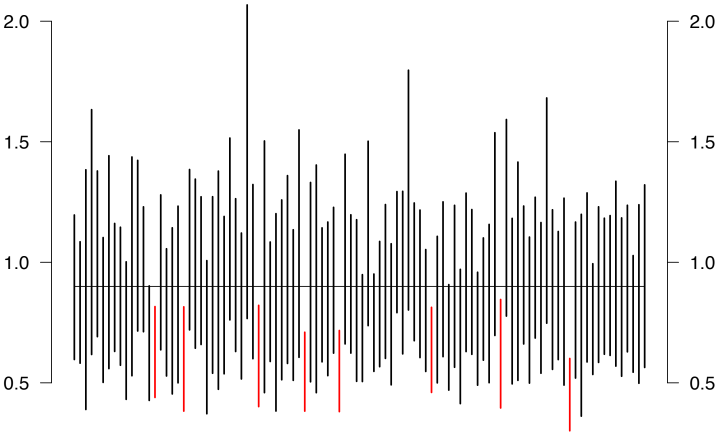
$$\bar{x} \pm t_{\alpha/2,n-1} \frac{s}{\sqrt{n-1}}$$

```
set.seed(12345)
seeds <- sample(10^6:(10^7 - 1), size = 100, replace = FALSE)

ci_lo <- numeric(100)  # vector for 100 CI low endpoints
ci_hi <- numeric(100)  # vector for 100 CI high endpoints

numJobs <- 30
for (i in 1:100) {
    output <- ssq(numJobs, seeds[i], showOutput = FALSE,
                       saveServiceTimes = TRUE)
    ci <- t.test(output$serviceTimes, conf.level = 0.95)
    ci_lo[i] <- ci$conf.int[1]
    ci_hi[i] <- ci$conf.int[2]
}
```

Confidence Intervals for Discrete-Event Simulation



- 92 intervals cover $\mu = 0.9$
- 8 intervals missed low (red)
- CIs are <u>approximate</u> when population distribution not *normal*

- The distribution of wait times for ssq is unknown
- Try `t.test()` approach to produce 95% confidence intervals

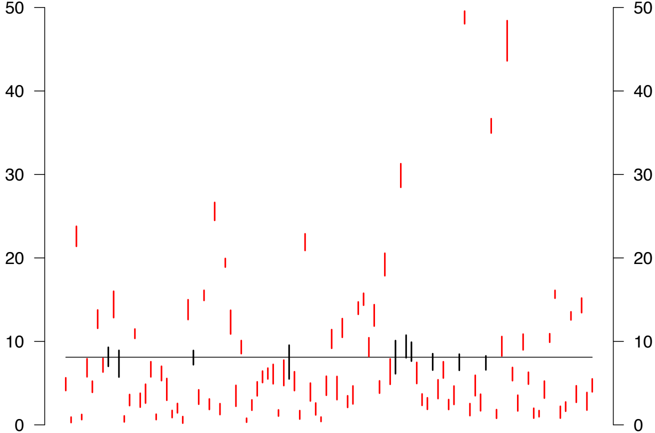
```
ci_lo <- rep(NA, 100)  # vector for 100 CI low endpoints
ci_hi <- rep(NA, 100)  # vector for 100 CI high endpoints

numJobs <- 30
warmup  <- 1000
for (i in 1:100) {
  output <- ssq(numJobs + warmup, seeds[i], showOutput = FALSE,
                saveWaitTimes = TRUE)
  waits <- output$waitTimes[-(1:warmup)]

  ci <- t.test(waits, conf.level = 0.95)
  ci_lo[i] <- ci$conf.int[1]
  ci_hi[i] <- ci$conf.int[2]
}
```

   0-13-142917-5

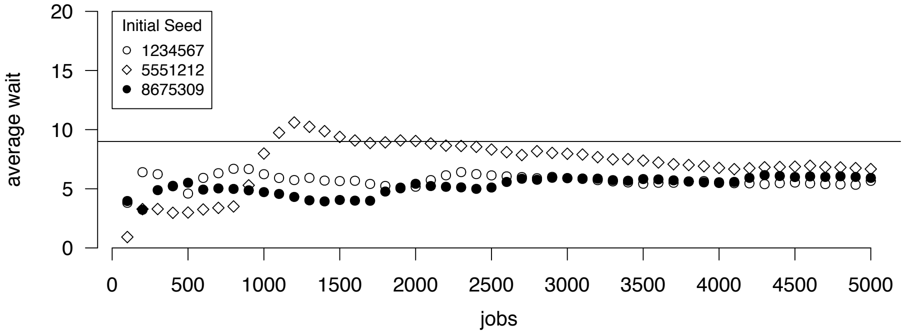
Confidence Intervals for Discrete-Event Simulation



- For an $M/M/1$ queue with arrival rate λ and service rate ν, the theoretical steady-state wait is $1/(\nu - \lambda) - 1/\nu = 8.1$
- Despite claim of 95% confidence, only 10 of 100 CIs bracket
- 28 miss high, 62 miss low

Effect of Initial Conditions: "Warm Up"

- `ssq`: accumulated $\bar{w}$ printed every 100 jobs

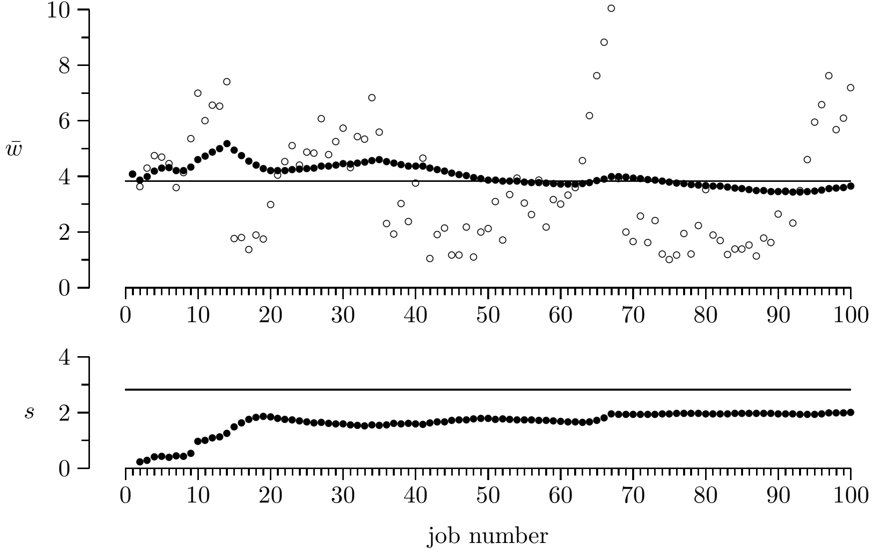


- Again, the theoretical steady-state wait is 8.1

Queueing Wait Times

- How does a long wait for the first person in line affect. . .
    - second in line? third in line? . . . far back in line?

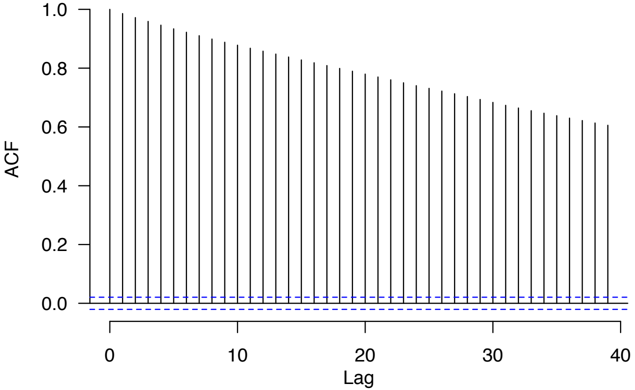
Example: Effect of Autocorrelation



(Note: different parameter values here than in other examples)

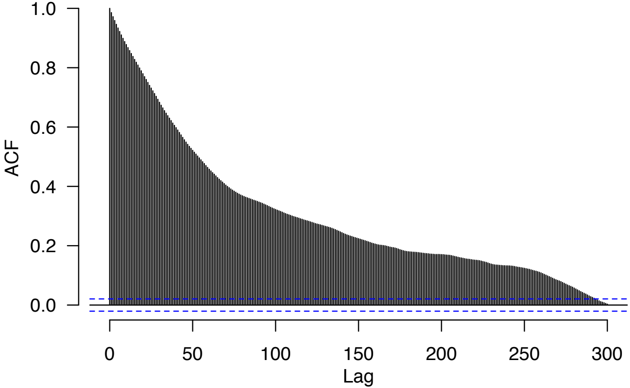
Autocorrelation of ssq Wait Times

- In ssq, strong positive autocorrelation b/w wait times




```
output <- ssq(10000, 1234567, showOutput = F, saveWaitTimes = T)
waits <- output$waitTimes[-(1:1000)]  # remove "warm-up"
acf(waits)
```

- Strong positive autocorrelation even for large lags




```
output <- ssq(10000, 1234567, showOutput = F, saveWaitTimes = T)
waits <- output$waitTimes[-(1:1000)]  # remove "warm-up"
acf(waits, lag.max = 300)
```

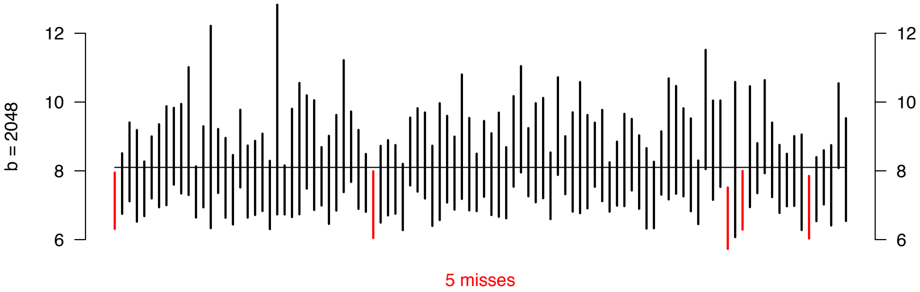
Confidence Intervals in Presence of Autocorrelation

- Method of Batch Means:
  (a) Make *one* long simulation run, deleting warmup
  (b) Partition into n batches each of batch size b
  (c) Compute a sample mean for each batch
  (d) Construct an interval estimate using means of batches

$$\underbrace{x_1, x_2, \ldots, x_b}_{\bar{x}_1}, \underbrace{x_{b+1}, x_{b+2}, \ldots, x_{2b}}_{\bar{x}_2}, \ \ldots, \ \underbrace{x_{(n-1)b+1}, x_{(n-1)b+2}, \ldots, x_{nb}}_{\bar{x}_n}$$

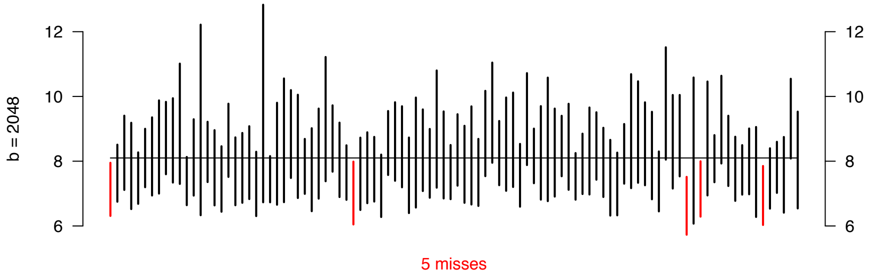
- Each batch: like a "transient simulation" w/o initial state bias
- State @ beginning of each batch is state @ end of previous

Batch Means for ssq Wait Times



- 100 different 95% confidence intervals using 100 diff. seeds
- Number of batches $n = 32$, batch size $b = 2048$
- First 1024 jobs discarded to mitigate start-up bias
- Batch means reduces the impact of autocorrelated data

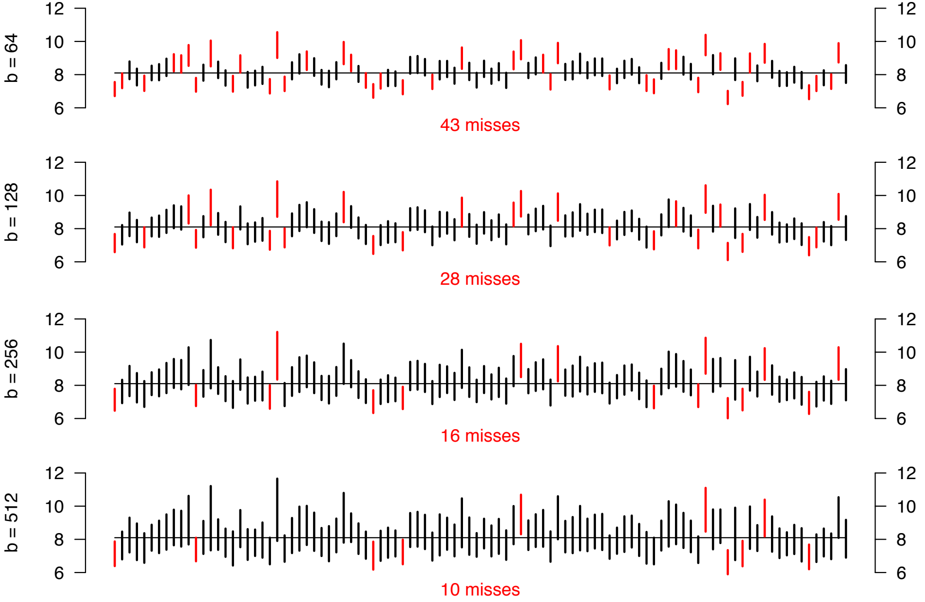
Batch Means for ssq Wait Times




```
numJobs <- 65536; warmup <- 1024
b <- 2048; n <-  numJobs / b

output <- ssq(numJobs + warmup, 1234567, showOutput = FALSE, saveWaitTimes = TRUE)
waits  <- output$waitTimes[-(1:warmup)]  # remove warmup
batchMeans <- numeric(n)
for (j in 1:n) {
    batch <- waits[(1:b) + (b * (j - 1))]
    batchMeans[j] <- mean(batch)
}
ci <- t.test(batchMeans, conf.level = 0.95)
```

Batch Size & Confidence: ssq with $b = 64, 128, 256, 512$



Effect of Batch Parameters

- Consider any one confidence interval from previous figures
- Choice of (b, n) has *no* impact on the *point* estimate
- Only the width of the 95% confidence interval is affected

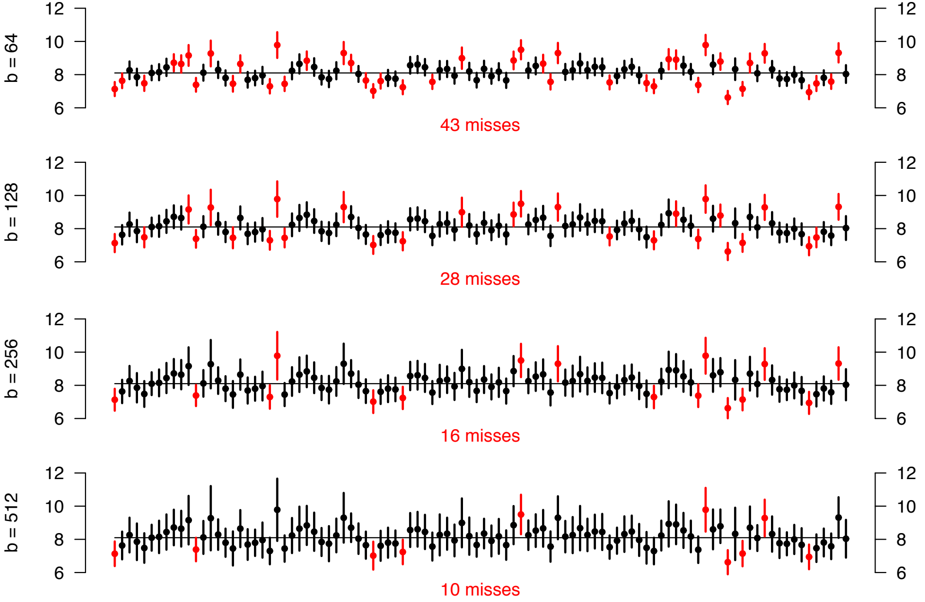
- Use `ssq`[1] to generate 65536 waits with $(\lambda, \nu) = (1, 10/9)$
- Batch means with different (b, n):

(b, n)	$(32, 2048)$	$(128, 512)$	$(512, 128)$	$(2048, 32)$
$\bar{w}$	8.68 ± 0.40	8.68 ± 0.73	8.68 ± 1.14	8.68 ± 1.38

- Note that $\bar{w} = 8.68$ is independent of (b, n)
- Width of the interval estimate is not

[1] Use `seed` = 1234567, `maxArrivals` = 65536 + 1024; discard first 1024 waits.

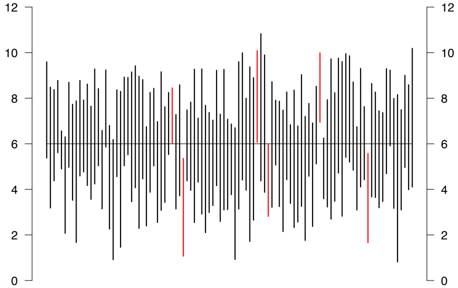
Batch Size & Confidence: ssq with $b = 64, 128, 256, 512$



Choosing Batch Size

- Schmeiser (1982) suggests number of batches k between 10 and 30
- Pegden et al. (1995) recommend a batch size b that is at least $10\times$ as large as the largest lag for which autocorrelation remains significant
- Banks et al. (2005) recommend b be increased until the lag-one autocorrelaction between batch means is less than 0.2

In-class Work: Meaning of Confidence



- Create figures like above, but using `craps` instead of *normal*:
  - Fig 1: 100 CIs with each CI via 9 samples from `craps(10)`
  - Fig 2: 100 CIs with each CI via 9 samples from `craps(100)`
- For each sample, construct a 95% CI using `t.test`
- Color intervals that miss using red (true prob: 244/495)

In-class Work: Getting You Started

- Create an empty plot with true theoretical value:

```
plot(NA, NA, xlim = c(1,100), ylim = c(ylo,yhi),
     bty = "n", xaxt = "n",
     xlab = "", ylab = "", las = 1)
exact <- 244/495
abline(h = exact)
```

- Use `segments` to draw each CI
- To create a 9-point vector of `craps` estimates:

```
values <- sapply(1:9,
   function(i) { craps(nreps, showProgress=FALSE) } )
```

In-Class Work

1. Repeat the previous batch-means experiments using sojourn times
2. For each of $b = 64, 128, 256, 512, 1024, 2048$:
   - Produce a figure of 100 different 95% confidence intervals
   - Use a different initial seed for producing each CI
   - Use red for intervals that miss, black for intervals that do not
   - In the figure, indicate the batch size and number of misses
   - Theoretical $\bar{o}$ for default `ssq` is $1/(\nu - \lambda) = 9.0$

(Note: each of these figures will require non-trivial execution time)

  0-13-142917-5