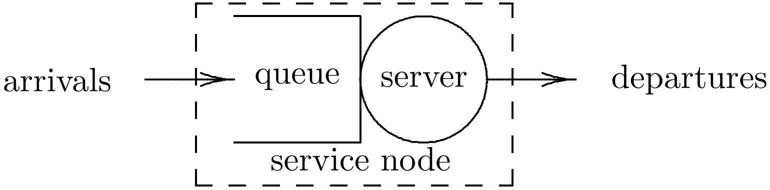
ssq in R

DCS 307:Simulation
Winter 2023

Queueing Model



- Assumptions:
  - FIFO queue (ssq); others: LIFO, SIRO, priority
  - server takes no break
  - non-preemptive: once service begins, must be completed
  - conservative: if queue is not empty, no server can be idle
  - infinite queue length possible

Queue Terminology

- An $A/B/c$ service node (Kendall's notation):
  - A: distribution of interarrival times
  - B: distribution of service times
  - c: number of servers
- Standard notation for interarrival, service:

M: *exponential*	D: deterministic (not stochastic)
E: *Erlang*	G: general (other distributions)

- Examples:
  - $M/M/1$: *exponential* interarrival and service times, one server
  - $M/G/4$: *exponential* interarrivals, four (identical) servers, each with same general service time distribution
  - `ssq`: $M/M/1$ service node by default

Job-Averaged Statistics

i	r_i	a_i	$n(a_i + \epsilon)$	w_i	b_i	s_i	c_i
1	15	15	1	0	15	43	58
2	32	47	2	11	58	36	94
3	24	71	2	23	94	34	128
4	40	111	2	17	128	30	158
5	12	123	3	35	158	38	196
6	29	152	3	44	196	30	226
7	80	232	1	0	232	31	263
8	13	245	2	18	263	29	292

- *Average service time*: $\bar{s} = 33.875$ seconds per job

  Service rate: $1/\bar{s} \approx 0.030$ jobs per second

  Arrival rate: $1/\bar{r} \approx 0.033$ jobs per second

  Service rate: $1/\bar{s} \approx 0.030$ jobs per second

Traffic Intensity

i	r_i	a_i	$n(a_i + \epsilon)$	w_i	b_i	s_i	c_i
1	15	15	1	0	15	43	58
2	32	47	2	11	58	36	94
3	24	71	2	23	94	34	128
4	40	111	2	17	128	30	158
5	12	123	3	35	158	38	196
6	29	152	3	44	196	30	226
7	80	232	1	0	232	31	263
8	13	245	2	18	263	29	292

- *Average service time*: $\bar{s} = 33.875$ seconds per job

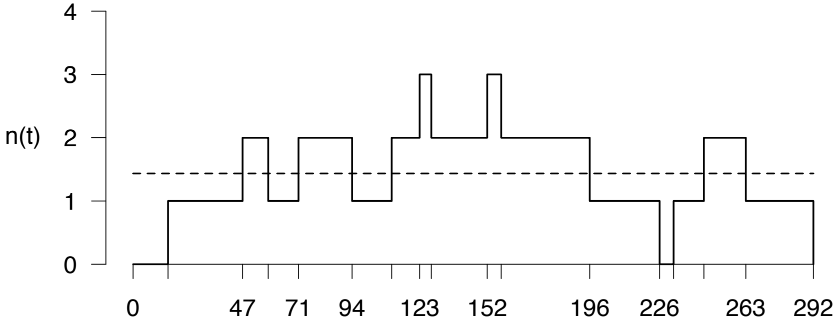
  Service rate: $1/\bar{s} \approx 0.030$ jobs per second

  Arrival rate: $1/\bar{r} \approx 0.033$ jobs per second

  Service rate: $1/\bar{s} \approx 0.030$ jobs per second

Time-Averaged Statistics

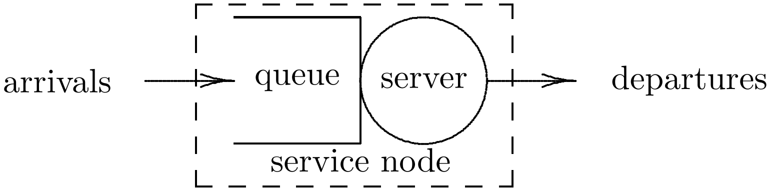
Time-averaged statistics: area under a curve (integration)



- $n(t)$: number in the system $\quad \overline{n} = 1.435$
- $q(t)$: number in the queue $\quad \overline{q} = 0.507$
- $x(t)$: number in service $\quad \overline{x} = 0.928$

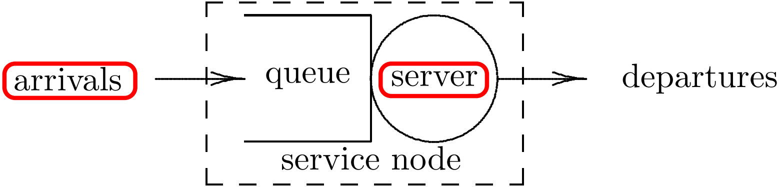
Stochastic Components

- Given the algorithms, what are the *stochastic* components?



Stochastic Components

- Given the algorithms, what are the *stochastic* components?



Model Development At Three Levels

1. *Conceptual*: big picture, what questions to ask (done!)
2. *Specification*: equations, (psuedocode) algorithms
3. *Computational*: implement in code

Computational Model Using `ssq`

Program `ssq()` implemented in R package `simEd`:

- To use, type `library(simEd)` on R/RStudio startup
- Interarrival times: *exponential*(1.0) `vexp(1, rate = 1)`
- Service times: *exponential*(0.9) `vexp(1, rate = 10/9)`

- Investigate *transient* behavior
  - Fix # of processed jobs, replicate using same initial state
  - Each replication uses different initial seed

- Investigate *steady-state* behavior
  - Will the statistics converge independent of initial seed?
  - How many jobs until steady-state?

Running ssq via R

```
> ssq(maxArrivals=1000, seed=1234567)

$customerArrivals
[1] 1000

$customerDepartures
[1] 1000

$simulationEndTime
[1] 996.19928

$averageWait
[1] 6.2324

$averageSojourn
[1] 7.1231

$avgNumInSystem
[1] 7.1503

$avgNumInQueue
[1] 6.2562

$utilization
[1] 0.89415
```

```
> ssq(maxArrivals=1000, seed=8675309)

$customerArrivals
[1] 1000

$customerDepartures
[1] 1000

$simulationEndTime
[1] 1022.81829

$averageWait
[1] 4.7171

$averageSojourn
[1] 5.6327

$avgNumInSystem
[1] 5.507

$avgNumInQueue
[1] 4.6118

$utilization
[1] 0.89518
```

?ssq gives R help

Arrival & Service Process in `ssq`

Program `ssq()` implemented in R package `simEd`:

- To use, type `library(simEd)` on R/RStudio startup
- Interarrival times: *exponential*(1.0) `vexp(1, rate = 1)`
- Service times: *exponential*(0.9) `vexp(1, rate = 10/9)`

What are these `vexp` function calls in `simEd`?

- `v*` functions are a collection of *random variate* generators (algorithmically generated realizations of random variables)
  - `vexp`, `vnorm`, `vunif`, ...
- compare to default R variate generators:
  - `rexp`, `rnorm`, `runif`, ...

What is the difference? Let's investigate using R...

Using `ssq` to Reproduce Your Table

`ssq`: can use custome inter-arrival and/or service functions

- can be built using built-in R generators (e.g., `rexp`)

```
getService <- function() {
    return(rgamma(1, shape = 1.0,  scale = 0.9))
}
```

  or

```
getService <- function(a = 1.0, b = 0.9) {
    rgamma(1, shape = a,  scale = b)
}
```

- can also be built using hard-coded trace data

Use `?ssq` and let's try it out ...

Grabbing Data from ssq

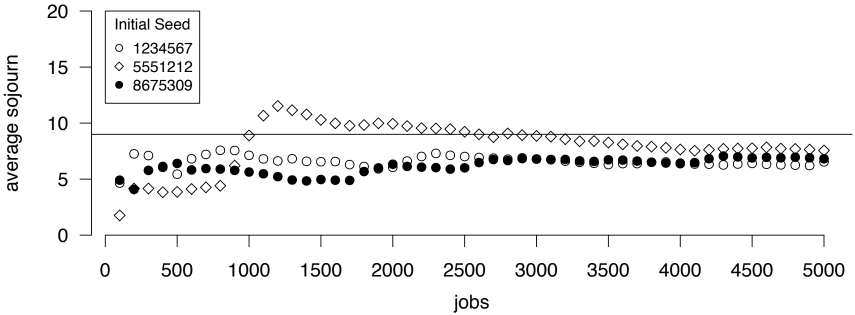
- ssq's output, like all R lists, allows individual access via the dollar sign

```
> ssq(maxArrivals=1000, seed=8675309)
$customerArrivals
[1] 1000
$customerDepartures
[1] 1000
```

Let's try it out...

Convergence To Steady-State

- The accumulated $\bar{o}$ printed every 100 jobs



- Steady state average sojourn time is $\dfrac{1}{\mu - \lambda} = \dfrac{1}{10/9 - 1} = 9$

- Convergence is slow, erratic, and dependent on initial seed

In-class Exercises (in R)

1. Write R functions for three different gamma service processes:

```
getSvc1 = function() { rgamma(1, shape = 1.0,  scale = 0.9) }
getSvc2 = function() { rgamma(1, shape = 1.05, scale = 0.9) }
getSvc3 = function() { rgamma(1, shape = 1.1,  scale = 0.9) }
```

2. Experiment using the original and new service processes:

```
ssq(maxArrivals = 10000, seed = 1234567)
ssq(maxArrivals = 10000, seed = 1234567, serviceFcn = getSvc1)
ssq(maxArrivals = 10000, seed = 1234567, serviceFcn = getSvc2)
ssq(maxArrivals = 10000, seed = 1234567, serviceFcn = getSvc3)
```

What effects do the new service processes have on the output statistics? How do the arrival and service rates compare? What happens as you increase the number of arrivals?