LINE Solver Cheatsheet (R)

Node Types

`Cache`	Caching station
`ClassSwitch`	Class switching
`Delay`	Infinite server
`Fork`	Forks jobs
`Join`	Joins jobs
`Logger`	Trace logging
`Place`	Petri net place
`Queue`	Queueing station
`Router`	Routing node
`Sink`	Job departures (open)
`Source`	Job arrivals (open)
`Transition`	Petri net transition

R nodes are R6 objects: `Queue$new(model, "Q", SchedStrategy$FCFS)`. They register themselves with the model, so there is nothing to assign back.

Job Classes

```
ClosedClass$new(model, "name", N, refStat)
OpenClass$new(model, "name")
SelfLoopingClass$new(model, "name", N, ref)
```

`DisabledClass` has no R spelling; omit the class instead.

Distributions

`APH(`α`, T)`	Acyclic PH
`Cox2(`μ`, `ϕ`)`	Two-phase Coxian
`Coxian(`μ`, `ϕ`)`	Coxian
`Det(v)`	Deterministic
`Disabled`	None
`Erlang(`λ`, k)`	Erlang-k
`Exp(`λ`)`	Exponential
`Gamma(`α`, `β`)`	Gamma
`HyperExp(p, `λ_1`, `λ_2`)`	Hyper-exp
`Immediate`	Zero service time
`Lognormal(`μ`, `σ`)`	Log-normal
`Normal(`μ`, `σ`)`	Normal
`Pareto(`α`, k)`	Pareto
`PH(`α`, T)`	Phase-type
`Uniform(a, b)`	Uniform
`Weibull(`α`, `β`)`	Weibull

Discrete

`Bernoulli(p)`	Bernoulli
`Binomial(p, n)`	Binomial
`DiscreteUniform(a, b)`	Discr. uniform
`Geometric(p)`	Geometric
`Poisson(`λ`)`	Poisson
`Zipf(s, n)`	Popularity

Correlated / trace-driven

`BMAP(D0..Dk)`	Batch MAP
`NHPP(bp,`λ`,cyclic)`	Non-homog. Poisson
`DMAP(D0, D1)`	Discrete MAP
`EmpiricalCDF(F)`	Empirical CDF
`MAP(D0, D1)`	MAP
`MarkedMAP`	Marked MAP
`ME(`α`, T)`	Matrix-exp.
`CME(mean, order)`	Concentr. matrix-exp.
`MMPP2(`λ_1`, `λ_2`, `σ_1`, `σ_2`)`	MMPP(2)
`RAP(H0, H1)`	Rational AP
`Replayer(trace)`	Trace replay

Marked / time-inhomogeneous

`MAPt(bp,D0,D1,cyclic)`	Time-varying MAP
`PHt(bp,`α`,S,cyclic)`	Time-varying PH
`MPH(`α`,S,{`s_k`})`	Marked PH
`MMAPt(bp,D0,D1k,cyclic)`	Marked time-varying MAP
`MPHt(bp,`α`,S,{`s_k`},cyclic)`	Marked time-varying PH
`BMMAPt(bp,D0,D1kb,cyclic)`	Batch marked time-varying MAP

A mark selects the CLASS of the arriving job (`setMarkedArrival`); a `BMMAPt` epoch releases a whole batch of that class. As a service process the mark is discarded, and a `BMAP` or `BMMAPt` service is a bulk server. Declared by `LDES`; `MAPt`/`PHt` also by `FLD` method `kp`.

R distributions are plain functions, not classes: `Exp(0.5)`, `Erlang(2, 3)`. There is no `fitMean`; a mean m is `Exp(1/m)`.

Scheduling Strategies

`DPS`	Discriminatory PS
`EDD`	Earliest Due Date
`EDF`	Earliest Deadline First
`EXT`	External (source)
`FB`	Feedback (LAS)
`FCFS`	First-Come First-Served
`FCFSPI`	FCFS Preempt. Resume-idle
`FCFSPR`	FCFS Preemptive Resume
`FSP`	Fair Sojourn Protocol
`GPS`	Generalized PS
`HOL`	Head-of-Line Priority
`INF`	Infinite Servers
`LCFS`	Last-Come First-Served
`LCFSPI`	LCFS Preempt. Resume-idle
`LCFSPR`	LCFS Preemptive Resume
`LEPT`	Longest Expected PT
`LJF`	Longest Job First
`LPS`	Limited PS
`LRPT`	Longest Remaining PT
`OI`	Order Independent
`PAS`	Pass and Swap
`POLLING`	Polling (switchover)
`PS`	Processor Sharing
`PSJF`	Preemptive SJF
`SEPT`	Shortest Expected PT
`SETF`	Shortest Elapsed Time First
`SIRO`	Random
`SJF`	Shortest Job First
`SRPT`	Shortest Remaining PT

Solvers

`AUTO`	Auto-select best
`BA`	Bound analysis (closed)
`CTMC`	CTMC (exact, small)
`ENV`	Random environments
`FLD`	Fluid/mean-field ODEs
`LDES`	Discrete Event Sim
`LN`	Layered networks
`MAM`	Matrix Analytic Methods
`MVA`	Mean Value Analysis
`NC`	Normalizing Constant Methods
`SSA`	Stochastic simulation

Wrappers

`JMT`	JMT simulator
`LQNS`	LQNS interface; `qns.*` methods run qnsolver on a Network

Solver Options

```
MVA$new(model, "exact")
JMT$new(model, d = 123,
samples = 10000)
SSA$new(model, samples = 5000,
method = "para")
LDES$new(model, d = 23000)
```

Output Metrics

`ArvR`	Arrival rate
`QLen`	Mean queue length
`ResidT`	Residence time (total across visits)
`RespT`	Response time (per visit)
`Tput`	Throughput
`Util`	Server utilization
`QLenVar`	Queue-length variance
`QLenSCV`	Queue-length squared coeff. of variation
`QLenSkew`	Queue-length skewness
`RespTVar`	Response-time variance (FCFS only)
`RespTSCV`	Response-time SCV (FCFS only)
`RespTSkew`	Response-time skewness (FCFS only)

Moments (product form)

Moment tables have no R method; reach the engine's own report with `solver$analysis("var", raw = TRUE)`.

Analysis Methods

```
solver$avg_table()
solver$avg_sys_table()
solver$avg_node_table()
solver$avg_chain_table()
solver$get_avg()$QLen   # matrices
tget(tbl, "QLen", "Q", "C1")
```

Table Methods

`avg_table`	per station
`avg_chain_table`	per chain
`avg_node_table`	per node
`avg_sys_table`	system
`get_avg`	as matrices
`analysis`	any -a name
`method_used`	what actually ran
`solver_meta`	method, arith, iter

Routing

```
model$link(serial_routing(
n1, n2, n3))
P <- model$init_routing_matrix()
P$set(cls, n1, n2, 0.7)
P$set(cls, n1, n3, 0.3)
model$link(P)
n1$set_routing_strategy(cls,
RoutingStrategy$RROBIN)
```

Multi-Server & Finite Capacity

```
queue$set_number_of_servers(k)
queue$set_capacity(bufferSize)
queue$set_class_capacity(cls, n)
queue$set_drop_rule(cls, DropRule$Drop)
```

Class Switching

```
cs <- ClassSwitch$new(model, "CS")
P <- matrix(c(0.3, 0.7, 0.5, 0.5),
nrow = 2, byrow = TRUE)
cs$set_class_switch_matrix(P)
```

Fork-Join Networks

```
fork <- Fork$new(model, "Fork")
join <- Join$new(model, "Join", fork)
```

Layered Networks

```
# no layered builder:  wrap an existing .lqnx
txt <- paste(readLines(path), collapse = "\n")
LN$new(LayeredNetwork$new(txt))$avg_table()
# or capi_run(c("-f", path, "-i", "lqnx", "-s", "ln"))
```

Caching

```
cache <- Cache$new(model, "Cache", nItems, cap,
ReplacementStrategy$LRU)
cache$set_read(cls, Zipf(1.4, nItems))
MVA$new(model)$get_cache_avg_table() # hit/miss
MVA$new(model)$get_item_avg_table() # per-item
```

ReplacementStrategy

`CLIMB`	Climb
`FIFO`	First-in first-out
`HLRU`	h-LRU
`LRU`	Least recently used
`QLRU`	q-LRU
`RR`	Random replacement
`SFIFO`	Static FIFO

Model Import/Export

```
save_model(model, "model.json")
model <- load_model("model.json")
validate_model(model)
# no viewer; export and open in JMT
```

Routing Strategies

`JSQ`	Join shortest queue
`KCHOICES`	Power of k choices
`PROB`	Probabilistic routing
`RAND`	Random selection
`RROBIN`	Round-robin
`WRROBIN`	Weighted round-robin

Stochastic Petri Nets

```
p <- Place$new(model, "P1")
t <- Transition$new(model, "T1")
m <- t$add_mode("Mode1")
t$set_number_of_servers(m, Inf)
t$set_distribution(m, Exp(4))
t$set_enabling_conditions(m, jobclass, p, 1)
t$set_inhibiting_conditions(m, jobclass, p, 3)
t$set_firing_outcome(m, jobclass, sink, 1)
f <- Place$new(model, "Buf"); f$set_fluid()
t$set_fluid_rate(m, f, 1.5)
```

Markov Chains (solved directly)

`MarkovProcess` and `MarkovChain` have no R spelling. The engine's own routines are reachable by name: `api_invoke("ctmc_solve", list(Q = Q))`.

Random Environments

```
# no Environment builder:  solve a saved document
capi_run(c("-f", path, "-s", "env",
"-a", "avg", "-o", "json"))
```

Finite Capacity Regions

```
model$add_region("FCR", list(queue1, queue2),
global_max_jobs = 10,
class_max_jobs = c(Class1 = 5),
drop_rule = c(Class1 = DropRule$drop))
```

Load-Dependent Service

```
queue$set_load_dependence(
pmin(seq_len(N), c))
# class dependence takes no R callback
```

Impatience & Retrials

```
queue$set_patience(cls, Exp(0.1))
queue$set_balking(cls, "QUEUE_LENGTH", thresholds)
queue$set_retrial(cls, Exp(2), maxAttempts)
```

State Probabilities

```
solver$analysis("prob", raw = TRUE)
solver$analysis("marg", raw = TRUE)
solver$analysis("normconst", raw = TRUE)
```

Transient & Distributions

```
solver$analysis("tran", raw = TRUE)
solver$analysis("cdf", raw = TRUE)
solver$analysis("reward", raw = TRUE)
# rewards are declared in the document
```

Symbolic Analysis

```
# needs a SymEngine-enabled build
line_version()$features$symengine
solver$analysis("sens", raw = TRUE)
```