Inventory Model Simulation

- **Assumptions:**
  a) customers arrive as Poisson(\(\lambda\)) for purchase of a product;
  b) unit price \(r\), with amount demanded \(D \sim G\), some distribution \(G\);
  c) when inventory \(x < s\), additional product is ordered to increase to some level \(S > s\), with time lag \(L\) for delivery;
  d) shop has cost function \(c(y)\) for \(y\) units, new orders arrive at times with some distribution \(G\);
  e) inventory holding cost is \(h\) per unit per unit time;
  f) if new customer demand \(w > y\) supply, excess demand is lost;
  g) goal of simulation is to determine profit at time \(T\).

- **Variables:**
  - **Time** \(t\);
  - **System State** \((x, y)\) (inventory \(x\), orders \(y\));
  - **Counters** total ordering cost \(C\), holding cost \(H\), revenue \(R\);
  - **Events** \(t_0\) for next arrival, \(t_1\) next delivery.

- **Typical Output** is average profit \((R - C - H)/T\);
  several runs provide \(E[(R - C - H)/T]\), which could be computed with different \((s, S)\)s to determine optimal ordering strategy.
INVENTORY MODEL CONTINUED

- Inventory Model Simulation Algorithm:
  given constants $r, h, s, S, L$, and $c(y), G(x)$;
  **Initialize**: $x = t = H = R = C = 0$,
  $t_0 = -\ln(\text{Uni}(0, 1))/\lambda$, $t_1 = L$, $y = S$.
  **While** $t \leq T$, update the system state using two cases:

  - **If** $t_0 < t_1$ (new customer before next order arrives)
    reset $H = H + (t_0 - t)hx$, $t = t_0$;
    generate $D \sim G$, set $w = \min(D, x)$;
    reset $R = R + wr$, $x = x - w$;
    if $x < s$ and $y = 0$, reset $y = S - x$, and $t_1 = t + L$;
    reset $t_0 = t - \ln(\text{Uni}(0, 1))/\lambda$.

  - **Else** (new order arrives before next customer)
    reset $H = H + (t_1 - t)hx$, $C = C + c(y)$, $t = t_1$;
    reset $x = x + y$, $y = 0$, $t_1 = \infty$.

  - **EndIf**

**EndWhile**

**Output** average profit $(R - C - H)/T$
INSURANCE RISK MODEL

Insurance Risk Model Simulation

- Assumptions:
  a) policyholders generate claims as Poisson(\(\lambda\)) process;
  b) amount of each claim \(C \sim F\), for some distribution \(F\);
  c) new customers enroll according to Poisson(\(\nu\)) process;
  d) policyholders remain enrolled for \(\text{Exp}(\mu)\) time;
  e) customers pay for policies at rate \(c\);
  f) initial capital \(a_0\) and \(n_0\) policyholders;
  g) goal of simulation is to check if capital \(> 0\) for all \(t \leq T\).

- Variables: **Time** \(t\);
  **System State** \((n, a)\) (policyholders \(n\), capital \(a\));
  **Events** new and lost policyholders, claims;
  **Event List** \(t_E\), the time for next event.

- Analysis: inter-event times are all independent Exponential; the minimum of independent Exponentials is Exponential; given \(n\), the next event time is \(\text{Exp}(\nu + n\mu + n\lambda)\); (new policy, lost policy, new claims) have probabilities \((\nu, n\mu, n\lambda)/(\nu + n\mu + n\lambda)\).

- Output \(I = 1\) if capital \(> 0\), \(\forall t \leq T\), otherwise \(I = 0\).
INSURANCE RISK MODEL CONTINUED

• Insurance Risk Model Simulation Algorithm:
  
  Initialize: \( t = 0, \ n = n_0, \ a = a_0, \ I = 1, \)
  \[ t_E = -\ln(\text{Uni}(0, 1))/(\nu + n(\mu + \lambda)) \].

  While \( t_E \leq T \) and \( I = 1 \), update the system using:
  reset \( a = a + nc(t_E - t) \), \( t = t_E \);
  generate \( J \) with pmf \( (\nu, n\mu, n\lambda)/(\nu + n\mu + n\lambda) \);
  if \( J = 1 \), reset \( n = n + 1 \) (new policy)
  elseif \( J = 2 \), reset \( n = n - 1 \) (lost policy)
  else (new claim)
    generate \( C \sim F \);
    if \( C > a \), set \( I = 0 \); otherwise reset \( a = a - C \);
  endif
  \[ t_E = t - \ln(\text{Uni}(0, 1))/(\nu + n(\mu + \lambda)) \].

  EndWhile

• Repeated runs provide
  a) \( E[I] \), the probability of capital > 0;
  b) other data which could be collected, e.g. average \( n \)'s, \( a \)'s.
Example: car insurance company with $T$ in months,
\[ F \sim 4000 + 1000 \times \text{Normal}(0, 1), \quad c = 100, \quad a_0 = \$100000, \]
\[ n_0 = 1000 \text{ customers}, \quad \lambda = 10, \quad \nu = .005, \quad \mu = .01, \]
so initially \[ (\nu, n\mu, n\lambda) / (\nu + n\mu + n\lambda) = (.4 .4 .2). \]
Matlab
\[
[I \ a \ t \ n]=\text{insrsk}(60,1000,100000,.005,10,.01,100);
disp([I \ a \ t \ n])
\]
\[ 1 \ 4.7792e+06 \ 59.978 \ 1001 \]
for \( i = 1 \) : \( 100 \)
\[
[F(i) \ a(i)]=\text{insrsk}(60,1000,100000,.005,10,.01,100);
end, \text{disp}([\text{mean}(F) \ \text{mean}(a)])
\]
\[ 1 \ 4.8895e+06 \]
function \([I \ a \ t \ n] = \text{insrsk}(T,ni,ai,la,nu,\mu,c)\)
% Insurance Risk Model, with
% initial customers \( ni \), initial capital \( ai \),
% claim rate \( la \), new customer signup rate \( nu \),
% customer loss rate \( \mu \), payment rate \( c \).
%
t = 0; \ n = ni; \ a = ai; \ I = 1;
te = -\log(rand)/(\ nu + n*(\mu+la) );
while te <= T & I == 1
\[ a = a + n*c*( \ te - t ); \ t = te; \ U = \text{rand}; \]
if \( U < \ nu / (nu+n*(\mu+la)) \), \( n = n + 1 \);
elseif \( U > (n*la+nu)/(nu+n*(\mu+la)) \), \( n = n - 1 \);
else, \( C = F; \ a = a - C; \) if \( a < 0 \), \( I = 0 \); end
end, \( te = t - \log(rand)/(\ nu + n*(\mu+la) ) \);
end
function \( C = F \), \( C = 4000 + 1000*\text{randn}; \) % Normal
MACHINE REPAIR MODEL

Machine Repair Model Simulation

• Assumptions:
  a) system needs \( n \) working machines;
  b) machines fail independently after time \( X \sim F \), some \( F \);
  c) broken machine immediately replaced and sent for repair;
  d) one-person repair facility repairs machines in sequence;
  e) repaired machines become spares \( s \);
  d) repair times \( R \sim G \), for some \( G \);
  g) system crashes if machine fails with no spares available;
  h) assume initially \( n + s \) total machines;
  i) goal of simulation is determine expected failure time \( E[T] \).

• Variables:  
  Time \( t \);
  System State \( r\) # of machines in repair;
  Events \( t_1 \leq t_2 \leq \cdots \leq t_n \) failure times,
  and \( t^* \) time for next completed repair.

• Typical Output is crash time \( T \); several runs provide \( E[T] \).
MACHINE REPAIR MODEL CONTINUED

- Machine Repair Model Simulation Algorithm:
  given constants $n$, $s$, and $F(x)$, $G(x)$;

  **Initialize**: $t = r = 0$, $t^* = \infty$,
  generate $X_i \sim F$, $i = 1, 2, \ldots, n$ and sort to get $t_i$'s;

  **While** $r < s + 1$, update the system state using two cases:
  - **If** $t_1 < t^*$ (a new failure)
    reset $t = t_1$, $r = r + 1$;
    if $r < s + 1$ generate $X \sim F$,
    and sort $t_2, t_3, \ldots, t_n, t + X$;
    if $r = 1$, generate $R \sim G$, and reset $t^* = t + R$.
  - **Else** (a completed repair)
    reset $t = t^*$, $r = r - 1$;
    if $r > 0$, generate $R \sim G$, and reset $t^* = t + R$;
    otherwise set $t^* = \infty$.

  - **EndIf**

  **EndWhile**

  **Output** crash time $T = t$.

- Example: $n = 5$, $s = 4$, $F = 1 - e^{-2x}$, $G = 1 - e^{-5x}$. 