Simulation Notes 67

Eric Desoiza

Chapter 6

Multivariate Normal

Let $Z_1, ..., Z_m$ be iid normal random variables with mean 0 and variance 1.

The vector $X_1, ..., X_n$ is said to have a multivaraite normal distribution. That is if each is a constant plus a linear combination of the same set of random variables. Since the sum is normal, X_i is thus normal

The means and covariances are as follows

$$E[X_i] = \mu_i$$

$$\text{Cov}(X_i, X_j) = \sum_{k=1}^{m} a_{ik}a_{jk}$$

We can compress a lot of this into matrix notation

$$\boldsymbol{X'} = \boldsymbol{AZ'} + \boldsymbol{\mu'}$$

Where A is the n x m matrix where X is the multivariate vector and Z is the row vector of independent standard normals and μ is the vector of means. If **C** is the matrix that depicts $\text{Cov}(X_i, X_j)$ then we can write the covariance as follows

$$\boldsymbol{C} = \boldsymbol{AA'}$$

An important property is that the joint distribution of **X** is completely determined by the Expected value and covariances. The joint distribution is determined by the knowledge of the mean vector μ and the covariance vector **C**

Generating a Multivariate Normal Random Vector

We need to find the **C** where

$$\boldsymbol{C} = \boldsymbol{AA'}$$

and generate the independent standard normals and set

$$\boldsymbol{X'} = \boldsymbol{AZ'} + \boldsymbol{\mu'}$$

To find such a matrix we can use Choleski decomposition, which states for any n x n symmetric and positive definite matrix, there is an n x n lower triable A such that $\boldsymbol{M = AA'}$.

$$\begin{bmatrix} a_{11} & 0 \\ a_{21} & a_{22} \end{bmatrix} * \begin{bmatrix} a_{11} & a_{21} \\ 0 & a_{22} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & c \\ c & c_2^2 \end{bmatrix}$$

Which yields the following system of equations

$$a_{11}^2 = \sigma_1^2, \quad a_{11}a_{21} = c, \quad a_{21}^2 + a_{22}^2 = \sigma_2^2$$

We can then evaluate ρ as $\frac{c}{\sigma_1\sigma_2}$ and hence

$$a_{11} = \sigma_1, \quad a_{21} = c/\sigma_1 = \rho\sigma_2, \quad a_{22} = \sqrt{\sigma_2^2 - \rho^2\sigma_2^2} = \sigma_2\sqrt{1-\rho^2}$$

or

$$\boldsymbol{A} = \begin{bmatrix} \sigma_1 & 0 \\ \rho\sigma_2 & \sigma_2\sqrt{1-\rho^2} \end{bmatrix}$$

Where we can now generate X_1 and X_2 by generating Z_1 and Z_2 and filling our $\boldsymbol{X' = AZ' + \mu'}$ where $X_1 = \sigma_1 Z_1 + \mu_1$ and $X_2 = \rho\sigma_2 Z_1 + \sigma_2\sqrt{1-\rho^2}Z_2 + \mu + 2$

Big Page of getting the joint density function on page 101 :(

It is easier to do the Choleski decomposition when we look at elements of the matrices by going down successive columns.

Copulas

a joint dsitribution function that represents both marginal distribtuions being uniformly distribtued on (0,1) is called a copula, that is the joint distribution C(x,y) is a copula if C(0,0) = 0 and C(x, 1) = x, C(1, y) = y for $0 \leq x, y \leq 1$

Suppose we have joint H(x,y) with P(X ≤ x) = F(x) and P(Y ≤ y) = G(y) and we have some knowledge of their dependency so we want to pick a proper Joint CDF. Because X has a distribution F and Y has a distribution G it follows that F(X) and G(Y) are both uniform on (0,1). and thus they are a copula. This means to find a good H(x,y) we must find C(F(x), G(y))

We must first decide the marginal distributions of F and G and then select the proper copula that can model the presumed dependency. The resulting copula should be close to that of our assumed ρ.

A popular copula is that of a Gaussian. If X and Y are standard normal with a joint bivariate normal distribution with ρ then $\Phi(X)$ and $\Phi(Y)$ is called a Gaussian Copula

If s(x) and t(x) are increasing functions then the copula generated by the random vector s(X), t(Y) is equal to the copula generated by X,Y

$$C_{s(X),t(Y)}(x,y) = C_{X,Y}(x,y)$$

A tail distribution generated copula that indicates a positive correlation between X and Y and which gives a positive probability that X = Y is the *Marshall-Olkin copula.*

We can also use copulas to model n-dimensional probability distributions. The C is said to be a copula if all n marginal distributions are uniform on (0,1)

Generating Variables from Copula Models

Suppose we want to generate a random vector X with marginal distributions F and copula C, and that we can invert F. Because the joint distribution of F is C we can generate X by first generating a random vector having distribution C and then inverting the generated values to obtain the desired vector X.

Chapter 7

Simulation via Discrete Events

Variables:

1) Time t: Referring to the amount of (simulated) time that has elapsed
2) Counter variables: Keep a count of the number of times that certain events have occurred by time t
3) System State (**SS**) variable: Describes the "state of the system" at the time t

In all the queueing models, we suppose that the customer arrive in accordance with a nonhomogeneous Poisson process (NHPP) with a bounded intensity function $\lambda(t), t > 0$. Let λ be such that $\lambda(t) \leq \lambda$.

Subroutine for Generating T_s

1) Let t = s
2) Generate U
3) Let t = t - $\frac{1}{\lambda} \log U$
4) Generate U
5) If $U \leq \lambda(t)/\lambda$, set $T_s = t$ and stop.
6) Go to Step 2

Single-Server Queueing System

Consider a service station in which customers arrive in accordance with a NHPP. There is a single server. The amount of time it takes to serivce a customer is a random variable (independent), having distribution G. In addition, there is a fixed time T after which no additional arrivals are allowed to enter the system, although the server completes servicing all those that are already in the sytstem at T.

To do a simulation on the quantities of the average time a customer spedns in the stysyem and that average time past T that the last customer departs.

a) **Time Variable**: t
b) **Counter Variables**: N_A: number of arrivals by t, N_D: the number of departures by t
c) **System State Variable**: n: number of customers in the system at time t

There are two types of events, arrival and departure. The event list is

$$\boldsymbol{EL} = t_A, t_D$$

where t_A is the time of the next arrival (after t) and t_D is the service completion time of the customer presently being served. If there is no customer presently being served then the departure is ∞

The outputs will be collected as A(i) and D(i) and T_p the time past T that the last customer departs

a) Set $t = N_A = N_D = 0$

b) Set **SS** $= 0$
c) Generate T_0 and set $t_A = T_0, t_D = \infty$

To update the system we move along the time axis until we encounter the next event. We must consider different cases to see how this is accomplished which depends on which member of the event list is smaller.

Case 1, Arrival is before departure and arrival is before the closing time

We reset the time to arrival time, increment number of arrivals by 1, increase number of customers by one. get the next time of arrival and jump to it. Collect that arrival data

Case 2, Departure is before arrival time, and departure time is before closing time

We set time to departure time, decrease number of customers by 1, and increase number of departures by 1. Collect that departure data

Case 3, The minimum of arrival time vs. departure time is less than closing time and the number of customers is more than 0

We set time to departure, decrease number of customers by 1 and increase number of departures by 1. Collect departure data

Case 4, The minimum arrival time vs. departure time is greater than closing time and number of customers is 0

Collect overtime closing time as the max if time minus closing time vs. zero

Queueing System with Two Servers in Series

A two-server system in which customer arrive in accordance of NHPP and each arrival must be served by server 1 and then goes over to 2 at completion. This is called sequential or tandem queuing. Upon arrival, will go to server 1 if free or join queue. Then repeat that with server 2.

a) **Time Variable**: t
b) **System State (SS) Variable**: (n_1, n_2) if there are n_1 customers at server 1, including both in queue and service and n_2 at server 2
c) **Counter Variables**: N_A is the number of arrivals, N_D is the number of departures.
d) **Output Variables**: $A_1(n)$ is the arrival time of customer n, n ≥ 1. $A_2(n)$ is the arrive time of customer n at server 2, n ≥ 1. $D(n)$ is the departure time of customer n
e) **Event List**: t_A, t_1, t_2 where t_A is the time of the next arrival and t_1 is the service completion time of server 1 and t_2 is the service completion time of server 2

We will initialize the process as follows

1) Set $t = N_A = N_D = 0$
2) $\boldsymbol{SS} = (0, 0)$
3) Generate T_0 and set $t_A = T_0, t_1 = t_2 = \infty$

Here are the following cases to account for

Case 1, time of the next arrival is the lowest value of the time of the customer currently being helped at both server 1 and server 2

We will set time to time of next arrive. Increase number of arrivals by 1, increase number of customers at server 1 by 1, set time of next arrival to the generated next arrival. if number of customers served at server 1 is 1 then we generate the time of completion at server 1 by adding a generated value to the current time. Lastly we collect the Arrival time of the customer at server 1 at the number of customers at the time

Case 2, time of completion at server 1 is less than time of next arrival and time of completion at server 1 is less than or equal to server 2

we set the time to the time of completion at server 1. we decrease the number being served by server 1 by 1 and increase the number being served by server 2 by 1. If the number of customers being served at server 1 is zero then we set the time of completion by server 1 to ∞, if not we generate the time to complete service at server 1 and add it to the time and assign that value to the completion time at server 1. We then save the completion time at server to under the number of arrivals minus the number being served at server 1 to the current time

Case 3, the completion time of server 2 is less than the time of arrival and the completion time of server 2 is less than server 1's

We set the current time to the completion time at server 1. We increase the number of departures by 1 and decrease the number being served by server 2 by 1. If number being served by server 2 is zero then we set the time of completion to ∞. If the number being served by server 2 is more than 0 then we generate a completion time and add it to the current time and assign it to the time of completion server server 2. We then save the departure data at the number of departed customers to the current time.

Queueing System with Two Parallel Servers

Consider a model in which customers arrive at a system with two servers. You will try to go to server 1 if they are free if not go to server 2 if they are free and if not you enter the queue if both server are busy.

Additionally it is important to note that because of the path not being linear the arrival times will not match up with the departure times given what number has arrived. The person who came first may still take longer than the person who came second given our generate completion time value

We need the following variables for this situation

a) **Time Variable**: t
b) **System State Variable (SS)**: (n, i_1, i_2) if there are n customers in the system, i_1 is with server 1 and i_2 is with server 2. Not that **SS** = (0) when the system is empty, and **SS** = (1,j,0) or (1,0,j) when the only customer is j and they are being served by server 1 or server 2, respectively
c) **Counter Variables**: N_A is the number of customer arrivals at t. C_j is the number of customers served by j, j = 1,2, by time t
d) **Output Variables**: A(n) is the arrival time of customer n. D(n) is the departure time of customer n
e) **Event List**: t_A, t_1, t_2. t_A is the time of the next arrival, t_1 and t_2 are the service completion time at their respective servers 1 and 2.

We will initialize as follows

1) $t = N_A = C_1 = C_2 = 0$
2) **SS** = (0)
3) Generate T_0, set $t_A = T_0, t_1 = t_2 = \infty$

Here are the following cases to account for

Case 1.1, the system state has customers with some somebody being serviced and the time of the next arrival is the minimum of the completion of the service completion times 1 and 2

We will set the time to the time of the next arrival. We will increase the number of arrivals by 1. We will set the time of the next arrival by the generated next arrival time. We will save the arrival time of the cumulative number of arrive customers to the current time

Case 1.2, the System stat is empty

We set the system state to (1, number of arrivals, 0). we generate the time of service completion for that first server

Case 1.3, the system state has one customer at the first server

We set the system state to (2, j, number of arrivals). we generate the time of completion for server 2

Case 1.4, the system state has one customer at the second server

We set the system state to (2, number of arrivals, j). we generate the time of completion for server 1

Case 1.5, the customer number is more than 1

we set the system state to (n + 1, server 1 completion time, server 2 completion time)

Case 2.1,the system state has a number of customers with completion times at both servers and the time of completion for server 1 is less than the arrival time of the next customer and the time of completion for server 1 is less than or equal to server 2's completion time

We set the current time to the time of completion for serrver 1 and increase the number of completed customers for server 1 by 1. we save the number of departed customers from server 1 at current time.

Case 2.2, if the number being served is 1

We set the system state to empty and we set the time of completion for server 1 to ∞

Case 2.3, if the number being served is 2

We set the system state to (1,0,time to complete at server 2). We set the time of completion for server 1 to be ∞

Case 2.3, there are more than 2 customers being served

let variable m be the maximum of service time for server 1 and 2. Set the system state to (n - 1, m + 1, i_2). Generate the completion time and set it to the completion time for server 1 plus the current time

Inventory Model

Consider a shop that sells a specific product with price r per unit. and the Customers demand this product in accordance of a Poisson Process with rate λ. The amount demanded by each customer has distribution G. Inventory must be maintained at a certain level and if it falls below a threshold the shop must order more.

We need the follow defining variables

a) **Time Variable** t
b) **System State Variable** (x,y) where x is the amount of inventory in hand and y is the amount on order
c) **Counter Variables** C is the total amount of ordering costs by t. H is the total amount fo inventory holding costs by t. R is the total amount of revenue earned by time t.
d) **Events** will consist of a customer or an order arrival. t_0 is the arrival time of the next customer. t_1 is the time of an order being delivered.

We will account for the following situations

Case 1 the time of next arrival is less than the time of an order being delivered.

We will increase the total amount of holding costs by time of next arrival minus current time times the amount of inventory in hand and the holding cost h. We will set the current time. We generate a value from the demand distribution to attribute to the customer arriving next. We set w to be the minimum of that value and the amount of inventory and we give that to the customer and we subtract that amount from the inventory. We then set the amount of Revenue to increase by the amount of units and the price per unit. If the amount of units in inventory is less than the threshold and the amount on order is zero then we set y to the amount expected to have after an order subtracted by what is in hand. We then set the time of next arrival of the current time plus the generated order time. We now inversely sample the arrival of the next customer by taking a uniform random number (0,1) and apply it to $t_0 = t - \frac{1}{\lambda}\log(U)$

Case 2 the time of next customer arrival is less than or equal to the time of next order arrival

We increase the amount of holding cost by time of next order minus current time times amount of inventory times holding cost per unit. We set the current time to time of next order arrival. We increase the total ordering cost by the ordering cost distribution value at y many items. We then increase the amount of units in inventory by the amount ordered. We then set the amount ordered to zero and the time of next arrival as ∞

Insurance Risk Model

Suppose that the different policy holders of a casualty insurance company generate claims according to a poisson process with rate λ, and a claim amount distribution F. New customers sign at a poisson process with rate v and the exisiting policy holder remains with the company with a exponential distribution of μ. The holder pays at a fixed rate of c per unit time. we start with n_0 customers and initial capital of positive a_0.

We need the following variables

a) **Time Variable** t
b) **System State Variable** (n, a) where n is the number of policyholders and a is current capital
c) **Events** new policy holder event, lost policy holder, a claim. The event list consists of a single value which is equal to the time of next event occurrence.
d) **Event List** t_E if (n,a) is the SS at time t. then because the minimum is exponential the time at which the next event occurs with equal t + X, where X is an exponential random variable with rate $v + n\mu + n\lambda$.

$$\text{A new policyholder probability} = \frac{v}{v + n\mu + n\lambda}$$

$$\text{A lost policyholder probability} = \frac{n\mu}{v + n\mu + n\lambda}$$

$$\text{A claim probability} = \frac{n\lambda}{v + n\mu + n\lambda}$$

We will generate all of these times and then choose the minimum for the next event

e) **I** Where or not the firms capital is non-negative from the beginning to current time

We must initialize the following values

1) t = 0, a = a_0, n = $n = n_0$
2) Generate X and $t_E = X$

We will now account for the following situations

Case 1 the next event is past the end of the simulation time

We set I to 1 and end the run

Case 2 if the next event is less than the end of the simulation time

We increase the current capital by the number of policyholders times the cost per unit time times the different in time from now til the next event. We generate the event type from the above 3 probabilities and then if it is a new customer we increase number of customers by 1. If it is a lost customer we decrease it by 1. If it is a claim we generate the claim amount and if it is more than the companies capital we set I to 0 and end and if not we decrease the amount of capital by Y. we then generate the next time of event

Repair Problem

A system needs n working machines to be operational. To avoid any issues there are spares. When a machine breaks, a spare is brought in and the broken machine is sent off the repairs. there is a single repairman. Once it is fixed it is kept as a spare. repair times are independnt with a distirbution of G. and the likelihood of breaking is an independent distribution of F. the system crashes if there is not enough machines to work due to a lack of spares and a broken machine.

We need the following variables

a) **Time Variable** t
b) **System State Variable r** number of machines that are down at time t

For each unit we need to determine the minimum next time of failure and set that as the next event and t* to be the time for the currently being repaired machine to be repaired.

We will initialize the following variables

1) t = r = 0, t* = ∞
2) Generate X_n variables and order them
3) Set event list as $t_1, ..., t_n, t*$

We will account for the following cases

Case 1, the next break time is less than the repair time

Set the current time to the break time, increase the number being repaired by 1.

a) If the number being repaired is equal to the number of spares plus 1 then we need to collect the current time since the machines are done
b) If the number being repaired is less than the number of spares plus 1 then we will generate the next break time which will be for the spare being used now. We will sort the break times in ascending order
c) if the number being repaired is 1 then we generate the repair time and set the time of repair to the current time plus repair time.

Case 2, the time of the next repair is less than or equal to the next break

we set the time to the time of repair. we decrease the number being repaired by 1. If the number being repaired is more than one then we generate the time of the current one to be repaired. If r = 0 then we set the repair time of the next machine to be ∞

Exercising a Stock Option

Let S_n denote the price of a specified stock at the end of day n. We can model this as follows

$$S_n = S_0 \exp\{X_1 + ... + X_n\}, \quad n \geq 0$$

Verification of the Simulation Model