Module 4. Monte Carlo Simulation Methods and Multivariate Probability Distributions.

Overview:
In this module we teach how to simulate random variables, and using them in Monte Carlo simulations to confirm or derive probabilistic properties of these random variables. You will learn how to generate data from multivariate distributions and describe their pdf. You will learn more probability calculations for multiple random variables: the means and variances of linear combinations of random variables, and the Central Limit Theorem. Monte Carlo simulations are used to check Central Limit Theorem and other probability properties taught in this module. The Monte Carlo methods are widely used in computational biology and applied statistics, which we will learn more in later modules.

This module consists of several lessons. In Lesson 1 we will look at Monte Carlo simulations for univariate random variables; In Lesson 2, multivariate distributions. Then in Lesson 3 we will look again at linear combinations of random variables in terms of their properties. Lesson 4 will focus on running what is known as Monte Carlo simulations for confirming or deriving formulas. Finally, Lesson 5 will introduce the Central Limit Theorem. As in previous modules, these lessons include examples, script for how to perform the calculations in R, and opportunities for practice.

Learning Objectives

1. **Monte Carlo simulations** to check various probability properties of random variables.
2. **Generating random variables** and defining the pdf for the **multivariate distribution** with independent marginals.
3. Generating random variables and defining the pdf for the **multivariate normal distribution** and the **multinomial distribution**.
4. Calculate **mean and variances of linear combination** of random variables.
5. Use **Central Limit Theorem** to calculate probabilities about the sample mean.

(Extra Readings. The materials in this module were partly covered in the textbook: Seefeld & Linder's book pages 105-107, 121-130, 217-221; Krijnen Pages 31-43.)

Lesson 1: Monte Carlo Simulation with Random Samples from a Univariate Distribution.

Objectives

By the end of this lesson you will have had the opportunity to:

- Use Monte Carlo simulations to estimate the event probability, the mean and the variance of an univariate distribution

Overview

This lesson introduces the Monte Carlo method to estimate various probabilistic quantities. We use a few examples to illustrate the Monte Carlo estimations in R. You will have the opportunity to practice these examples as well.

Monte Carlo Simulations to Estimate Probabilistic Quantities

At the end of last module, we taught how to generate random samples for a random variable X with a given probability distribution. We used these random samples for Monte Carlo estimation of event probability P(A). The reason is that, for a large sample of size n, P(A) will be close to the proportion of the sample belonging to the event A. This is in fact the definition of P(A) -- the long-term frequency of A.

Similar rational indicates that other probability quantities such as E(X) and Var(X) can also be estimated by Monte Carlo simulations. For big sample size n, the sample mean of a random sample $X_1, X_2, \ldots, X_n$ (from the same distribution of X) will be very close to the true mean E(X). This is called the Rule of Large Numbers in probability theory. Similarly, the sample variance will be close to the true variance Var(X) for large sample size.

Example: Monte Carlo Estimations for Normal Distribution

We consider a random variable X from the normal distribution $N(\mu = 2,\ \sigma = 3)$. The probability definition in last module tells us that the true mean E(X) is 2.

If you did not know that, but can generate a random sample $X_1, X_2, \ldots, X_n$ from $N(\mu = 2,\ \sigma = 3)$, then you can estimate the true mean by sample mean. For example, if I generate a random sample of size n=100,000, then the Monte Carlo estimate for the mean is

```
> mean(rnorm(n=100000, mean=2, sd=3))
[1] 1.99402
```

You can see that the Monte Carlo estimate for the mean 1.994 is very close to the true mean of 2.

The Monte Carlo estimate does change from different runs of the same simulation. Following we do 4 Monte Carlo simulations for the norm mean.

```
> for (i in 1:4) print(mean(rnorm(n=100000, mean=2, sd=3)))
[1] 2.008566
[1] 2.006411
[1] 2.020758
[1] 2.015036
```

We can see the estimated means are different for different runs, but all close to the true mean of 2.

Similarly, we can also estimate the variance Var(X) by Monte Carlo simulation

```
> var(rnorm(n=100000, mean=2, sd=3))
[1] 9.000293
```

Again the Monte Carlo estimate is close to the true Var(X)=9 (square of sd=3).

The accuracy of the Monte Carlo estimate depends on the sample size n. So for more accurate estimate, larger n should be used.

```
> x<-rnorm(n=100, mean=2, sd=3)
> c(mean(x), var(x))
[1] 2.306790 9.220502
```

When we use only n=100, the Monte Carlo estimates differ from the true values on the first decimal space. This is in contrast to the difference in second decimal space or smaller, when n=100,000.

Example: Monte Carlo Estimations for Gamma Distribution

Let us use Monte Carlo simulations to estimate some probability quantities for a random variable X from the Gamma distribution with shape parameter $\alpha = 3$ and scale parameter $\beta = 5$.

Let us find (a) P(1<X<10); (b) mean E(X); (c) standard deviation sd(X); and (d) variance Var(X).

These quantities can be calculated exactly from the formulas in the last module. But let us first find their Monte Carlo estimates, which is straightforward. We use a sample size n=100,000, and generate a random sample $X_1, X_2, \ldots, X_n$ from this distribution. Then we can calculate these quantities on this sample as the Monte Carlo estimates.

```
x<-rgamma(n=100000, shape=3, scale=5) #generate the random sample
#print results on this random sample. "\n" to continue printing in a new line.
cat("(a) P(1<X<10)=", mean( 1<x & x<10), "\n (b) E(X)=", mean(x), "\n (c) sd(X)=", sd(x), "\n (d) Var(X)=", var(x), "\n")
```

Run these commands, and we get the Monte Carlo estimates.

```
(a) P(1<X<10)= 0.32054
 (b) E(X)= 14.99879
 (c) sd(X)= 8.642188
 (d) Var(X)= 74.68742
```

We can compare these Monte Carlo estimates with their true values.

```
> c(integrate(function(x) dgamma(x, shape=3, scale=5), lower=1, upper=10)$value, 3*5, sqrt(3*5^2), 3*5^2)
[1]    0.3221751 15.0000000    8.6602540 75.0000000
```

We can see that the Monte Carlo estimates differ from the true theoretical values in their third digits. The first quantity is calculated from pdf. The last three quantities are calculated using the formulas listed in the Table 3.4.1 (Lesson 4 of Module 3).

Monte Carlo Methods

We have shown examples of simple Monte Carlo simulations to estimate some probability quantities. In these simple examples, we do know the true values from the analytic formulas. The Monte Carlo estimates are straightforward, and do provide good estimates for big sample size n.

In practice, the Monte Carlo estimates are generally used for more complicated models where the analytic answers are hard to get. In those cases, the Monte Carlo estimates are often still easy to do. We do more Monte Carlo simulations in the next few lessons.

Lesson Summary

In this lesson, we did Monte Carlo simulations from univariate distributions. You should know how to use Monte Carlo simulations to estimate various probability quantities in R.

In the next lesson, we cover multivariate distributions, and how to generate random samples for some multivariate distributions.

Lesson 2. Generating from multivariate distributions; multivariate normal distribution, and multinomial distribution.

Objectives

By the end of this lesson you will have had the opportunity to:

- Calculate joint pdf for the multivariate distribution with independent marginals
- Generate random variables from the multivariate distribution with independent marginals
- Generate random variables from the multivariate normal distribution and the multinomial distribution

Overview

In this lesson, we learn about pdf of multivariate distributions with independent marginals. We then learn about generating random samples from these multivariate distributions with independent marginals in R. Next, we will learn about getting pdfs and generating random samples from two multivariate distributions: the bivariate normal distribution and the multinominal distribution.

Multivariate Distribution with Independent Marginals.

We know that a univariate random variable X's distribution is described by its pdf f(x). The probability distribution about a d-dimensional random vector $(X_1, ..., X_d)$ is a multivariate distribution represented by the d-dimensional pdf $f(x_1, ..., x_d)$. Each element X_j in $(X_1, ..., X_d)$ is a random variable, and we call its pdf $f_j(x)$ as the j-th marginal pdf.

We first consider the simple case of random vector $(X_1, ..., X_d)$ whose elements are independent random variables. If each random variable in $(X_1, ..., X_d)$ are *independent* of the others (that is, its value is not affected by the values of the other random variables), then the joint pdf $f(x_1, ..., x_d) = f_1(x_1) f_1(x_2) \cdots f_1(x_d)$ is the product of the marginal pdf's.

A common basic case used in probability and statistics is the independently identically distributed (i.i.d.) random variables. That is, $X_1, ..., X_d$ are independent and all follow the same univariate distribution with univariate pdf $f_1(x)$. Hence its joint pdf is $f(x_1, ..., x_d) = f_1(x_1) f_1(x_2) \cdots f_1(x_d)$.

For example, d=3 random variables from the exponential distribution with parameter λ will have joint pdf $f(x_1, x_2, x_3) = (\lambda e^{-\lambda x_1})(\lambda e^{-\lambda x_2})(\lambda e^{-\lambda x_3})$. That is, the probability density (called likelihood in statistics theory of later modules) of $X_1 = 4$, $X_2 = 7$ and $X_3 = 2$ will be $f(4, 7, 2) = (\lambda e^{-4\lambda})(\lambda e^{-7\lambda})(\lambda e^{-2\lambda})$.

For a second example, d=4 random variables from the normal distribution $N(mean = \mu, sd = \sigma)$ will have joint pdf

$f(x_1, x_2, x_3, x_4) = (\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_1-\mu)^2}{2\sigma^2}})(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_2-\mu)^2}{2\sigma^2}})(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_3-\mu)^2}{2\sigma^2}})(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_4-\mu)^2}{2\sigma^2}})$. So the joint density at $X_1 = 1.4$, $X_2 = 3.2$, $X_3 = 0.7$ and $X_4 = 6.9$ will be

$f(1.4, 3.2, 0.7, 6.9) = (\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(1.4-\mu)^2}{2\sigma^2}})(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(3.2-\mu)^2}{2\sigma^2}})(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(0.7-\mu)^2}{2\sigma^2}})(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(6.9-\mu)^2}{2\sigma^2}})$.

So the representation of the multivariate distribution density function in the i.i.d. case is very simple. We can also calculate these densities in R using pre-programmed univariate pdf.

Calculate Joint pdf for i.i.d. Random Variables in R.

When the univariate pdf is pre-programmed in R (see the common distributions listed in Table 3.4.1 of Lesson 4 in Module 3), the joint pdf in i.i.d. case can be calculated using formula $f(x_1, ..., x_d) = f_1(x_1) f_1(x_2) \cdots f_1(x_d)$.

Example 1. (X_1, X_2, X_3) is a random vector whose elements are three independent random variables from the exponential distribution with parameter $\lambda = 1.5$. Then the joint probability density at $X_1 = 4$, $X_2 = 7$ and $X_3 = 2$ will be $f(4, 7, 2) = (1.5e^{-4(1.5)})(1.5e^{-7(1.5)})(1.5e^{-2(1.5)})$. We can calculate this using simple R commands

```
x<-c(4,7,2) #store the values of Xs in a vector
#product of univariate densities evaluated at each X element
prod(dexp(x, rate=1.5))
```

Example 2. (X_1, X_2, X_3, X_4) are four random variables from the normal distribution $N(\mu = 2, \sigma = 4)$. Then we can calculate the joint density at $X_1 = 1.4$, $X_2 = 3.2$, $X_3 = 0.7$ and $X_4 = 6.9$ by R commands

```
x<-c(1.4, 3.2, 0.7, 6.9) #store the values of Xs in a vector
prod(dnorm(x, mean=2, sd=4)) #joint pdf
```

These expression of the joint pdf (likelihood) can be used for finding maximum likelihood estimators in later modules.

Generating i.i.d. Random Variables in R.

To generate i.i.d. random variables from the joint multivariate pdf $f(x_1, ..., x_d) = f_1(x_1) f_1(x_2) \cdots f_1(x_d)$ is easy. Due to independence, we simply generate the d variables separately from the marginal distribution $f_1(x)$. This is in fact the same as sampling d times from the distribution $f_1(x)$.

Notice that, when we generate d samples from a univariate distribution, the sample is **d realizations of one univariate random variable**. However, we can also treat it as **one realization of a d-dimensional random** vector from the multivariate distribution with independent marginals.

For example, rpois(n=4, lambda=1) produces a 4-dimensional vector. This was considered as four realizations from the $Poisson(\lambda = 1)$ distribution in Module 3. Alternatively, this vector is also one realization of four independent variables (X_1, X_2, X_3, X_4) each from the $Poisson(\lambda = 1)$ distribution. Therefore, if we want a random sample of size 3 for the 4-dimensional random vector (X_1, X_2, X_3, X_4), we simply do the generation three times and save each realization in one row.
rbind(rpois(n=4, lambda=1), rpois(n=4, lambda=1), rpois(n=4, lambda=1))
This R command generates three realizations of (X_1, X_2, X_3, X_4), and saved the results in a 3 by 4 matrix (each row stores one realization).

Since each elements in the random vector (X_1, X_2, X_3, X_4) are i.i.d., the 3 by 4 matrix above simply contains 12 i.i.d. elements. Therefore, a faster alternative way is to generate 12 observations from the $Poisson(\lambda = 1)$ distribution, and then arrange them into the 3 by 4 matrix.
matrix(rpois(n=3*4, lambda=1), nrow=3)

As another example, we generate 50 realizations of three i.i.d. random variables X_1, X_2 and X_3 from the t-distribution with 7 degree of freedom. This is done with the R command matrix(rt(n=50*3, df=7), nrow=50).
Here we get a 50 by 3 matrix. Each row stores one realization of the three random variables X_1, X_2 and X_3.

Generating from Multivariate Distributions with Independent Marginals.
When the random variables $X_1, ..., X_d$ are independent, but not necessarily identically distributed, we also can generate the independent random variables separately from their marginal distributions.

For example, for the three independent random variables $X_1 \sim Poisson(\lambda = 1)$, $X_2 \sim Poisson(\lambda = 2)$, $X_3 \sim Poisson(\lambda = 3)$, we can generate the vector (X_1, X_2, X_3) using R command

c(rpois(n=1, lambda=1), rpois(n=1, lambda=2), rpois(n=1, lambda=3))

That is, we generate them separately and then put them together into a 3-dimensional vector.

If we wish to get 20 samples of the three independent random variables X_1, X_2 and X_3, then we need 20 such 3-dimensional vectors. We can put these 20 vectors in to a 20 by 3 matrix (each vector in a row). In R, we achieve this by

cbind(rpois(n=20, lambda=1), rpois(n=20, lambda=2), rpois(n=20, lambda=3))

cbind(rpois(20,1), rpois(20,2), rpois(20,3))

Here, we get 20 samples each of X_1, X_2 and X_3, and applies the column combination. Hence each row is in fact the combination of one sample each of X_1, X_2 and X_3.

Note

Generally the joint pdf $f(x_1,\ldots,x_d)$ is not a simple product of the marginal pdf's, $\neq f_1(x_1)f_2(x_2)\cdots f_d(x_d)$.

Hence we cannot always generate the multivariate distributions simply by sampling from their univariate marginal distributions.

The sampling from a general multivariate distributions is harder. We only discuss two example multivariate distributions in this section: bivariate normal distribution and the multinomial distribution.

Bivariate normal distribution (a continuous bivariate distribution).

Bivariate normal distribution is the distribution of a 2-dimensional random vector (X,Y), where X and Y each follow the univariate normal distribution $N(\mu_1,\sigma_1)$ and $N(\mu_2,\sigma_2)$ respectively.

That is, $E(X)=\mu_1, Var(X)=E[(X-EX)^2]=\sigma_1^2$ and
$E(Y)=\mu_2, Var(Y)=E[(Y-EY)^2]=\sigma_2^2$.

If X and Y are independent, then we can generate them separately as before.

However, generally X and Y are dependent. We measure their dependence by the covariance $Cov(X,Y)=E[(X-EX)(Y-EY)]=\sigma_{12}$. The zero covariance corresponds to the case that X and Y are independent.

Generally, the bivariate normal distribution is specified with the mean vector (μ_1,μ_2) and the covariance matrix $\begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{12} & \sigma_2^2\end{pmatrix}$.

We define the correlation as $\rho=\frac{\sigma_{12}}{\sigma_1\sigma_2}$. When $\rho=1$, X=Y always; when $\rho=-1$, X=-Y always; when $\rho=0$, X and Y are independent.

The pdf for the bivariate normal distribution is given by
$f(x,y)=\frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}e^{\frac{-1}{2(1-\rho^2)}[(\frac{x-\mu_1}{\sigma_1})^2-2\rho(\frac{x-\mu_1}{\sigma_1})(\frac{x-\mu_2}{\sigma_2})+(\frac{x-\mu_2}{\sigma_2})^2]}$.
You do not have to memorize the formula for the bivariate normal pdf. Using the package "mvtnorm", we can get the pdf using the function "dmvnorm()" and generate from the bivariate normal distribution using the function "rmvnorm()".

Example 4.2.1

To calculate the pdf $f(3,4)$ for bivariate normal distribution $N((\mu_1,\mu_2)=(1,2),\begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{12} & \sigma_2^2\end{pmatrix}=\begin{pmatrix}1 & 0.9\\ 0.9 & 1\end{pmatrix})$ at $(x,y)=(3,4)$, use R command

```
require(mvtnorm) #load library "mvtnorm"
dmvnorm(c(3,4), mean=c(1,2), sigma=matrix(c(1,0.9,0.9,1), nrow=2) )
[1] 0.04447738
```

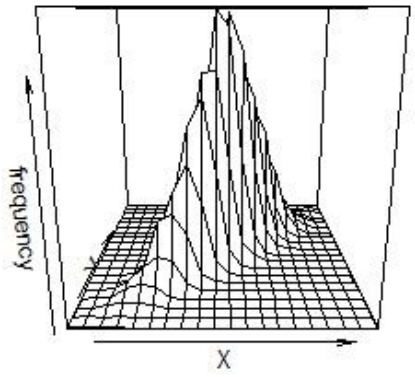
Hence f(3,4)=0.04447738

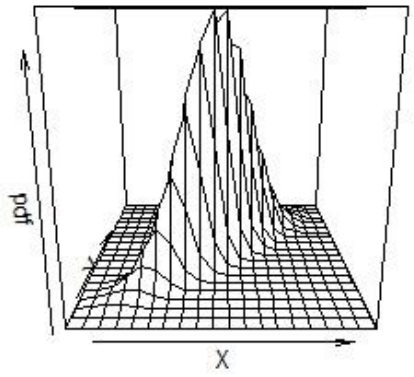
To generate three realizations (a 3 by 2 matrix, each realization in one row) from this distribution, use R command

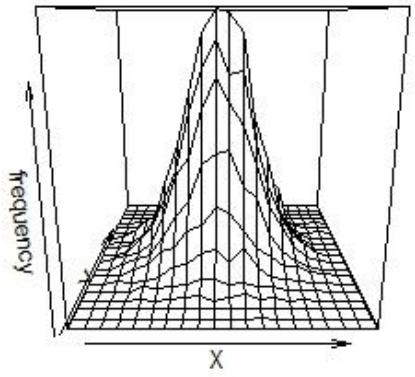
```
rmvnorm(3, mean=c(1,2), sigma=matrix(c(1,0.9,0.9,1), nrow=2) )
          [,1]     [,2]
[1,]  1.2122712 2.214745
[2,]  1.7119394 3.512107
[3,] -0.2733299 1.234825
```

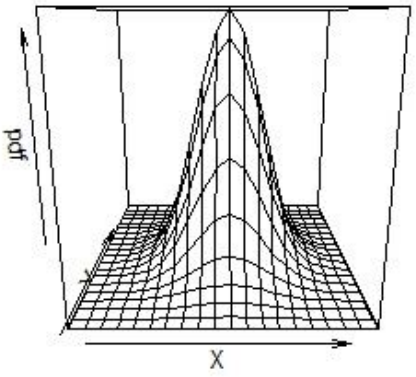
Following plot shows the three-dimensional histograms of 10,000 samples from the $N((\mu_1,\mu_2)=(1,2),\begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{12} & \sigma_2^2\end{pmatrix}=\begin{pmatrix}1 & 0.9\\ 0.9 & 1\end{pmatrix})$ and $N((\mu_1,\mu_2)=(1,2),\begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{12} & \sigma_2^2\end{pmatrix}=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix})$ respectively on the left. On the right, the three-dimensional perspective plots of the corresponding pdf are provided.

We can see that the histograms (on the left) are close to the corresponding theoretical pdf (on the right). For highly correlated bivariate normal distribution $\rho=0.9$, the data concentrates around the ridge of X=Y. For the independent bivariate normal distribution $\rho=0$, the shape of the pdf is like a hill -- product of the two bell curves on x-axis and y-axis.


frequency
X


pdf
X


frequency
X


pdf
X

R code and further resource

The above plots are produced with following R-script.

```
require(mvtnorm) #load library "mvtnorm"
require(gplots) #load library "gplots"

par(mfrow=c(2,2)) #put four plots in 2 rows by 2 columns
#generate random data from a bivariate normal distribution
my.data<-rmvnorm(10000,mean=c(1,2),sigma=matrix(c(1,0.9,0.9,1),nro
w=2))
#get data for 2d-histogram, no plot yet "show=F". Save to my.hist for later plotting
my.hist<-hist2d(my.data,nbins=20,show=F)
#use data in my.hist for 3d display of the histogram
persp(my.hist$x,my.hist$y,my.hist$counts,xlab="X",ylab="Y",zlab="f
requency")
#draw the density function (a 2d surface) in 3d display
persp(my.hist$x,my.hist$y,
matrix(dmvnorm(cbind(rep(my.hist$x,rep(20,20)),my.hist$y),mean=c(1
,2),sigma=matrix(c(1,0.9,0.9,1),nrow=2)),nrow=20) ,xlab="X",ylab="
Y",zlab="pdf")
#Redo above task for another bivariate normal distribution (different
covariance)
my.data<-rmvnorm(10000,mean=c(1,2),sigma=matrix(c(1,0,0,1),nrow=2)
)
my.hist<-hist2d(my.data,nbins=20,show=F)
persp(my.hist$x,my.hist$y,my.hist$counts,xlab="X",ylab="Y",zlab="f
requency")
persp(my.hist$x,my.hist$y,
matrix(dmvnorm(cbind(rep(my.hist$x,rep(20,20)),my.hist$y),mean=c(1
,2),sigma=matrix(c(1,0,0,1),nrow=2)),nrow=20) ,xlab="X",ylab="Y",z
lab="pdf")
```

Resource: the following website link has an interactive demo showing the shape of pdf for the bivariate normal distribution with various parameter values.

http://demonstrations.wolfram.com/TheBivariateNormalAndConditionalDistributions/

Multinomial distribution (a discrete multivariate distribution).

Recall that binomial distribution counts the number of "success" out of n independent trials each with two possible outcomes: "success" and "failure".

Generally if there are r possible outcomes in each trial, labeled as "1", "2", …, "r". Then the number of trials in each category out of n independent trials follow the multinomial distribution.

Let X_1, X_2, …, X_r denote respectively the number of trials with 1st, 2nd, …, r-th outcomes out of n independently repeated trials. Then $(X_1,\ldots,X_r)$ follows the multinomial distribution with size=n and probabilities=$(p_1,\ldots,p_r)$. Here $(p_1,\ldots,p_r)$ denote the probabilities for each possible outcome in one trial, so that $p_1+\ldots+p_r=1$.
The pdf for the multinomial distribution is given by
$f(x_1,\ldots,x_r)=\binom{n}{x_1x_2\ldots x_r}p_1^{x_1}p_2^{x_2}\ldots p_r^{x_r}$. (See pages 121-122 of Seefeld).

In R, the pdf of multinomial distribution is "dmultinom" function, and random samples of multinomial distribution are generated by the "rmultinom" function in the "stats" package.

dmultinom(x, prob) calculates the pdf at $x = (x_1, ..., x_r)$ when $prob = (p_1, ..., p_r)$.

rmultinom(n, size, prob) generates n samples from the multinomial distribution, with each sample saved in a column. So this generates a r by n matrix.

Please note the difference between rmvnorm (which stores each sample in a row), and rmultinom (which stores each sample in a column). There is no universal standard on how to store the multivariate data. The programmers choose to do row-wise or column-wise storage by their own preference. When using their code, you need to know if the data is stored by row or by column, and carry out your operations accordingly.

Example 4.2.2

Assume the four types of bases A,G,C,U occur with proportions 0.4,0.3,0.2 and 0.1 in a RNA sequence. Then the number of the nucleotides of each type among 10 randomly selected positions on this sequence follows a multinomial distribution with size=10 and probabilities=(0.4,0.3,0.2,0.1).

To find the probability that these 10 nucleotides consists of 3A,3G,2C,2U, we use the R command

```
> dmultinom(c(3,3,2,2),size=10,prob=c(0.4,0.3,0.2,0.1)) #density of multinomial
[1] 0.01741824
```

To generate 2 random samples from this multinomial distribution, we use the R command

```
> rmultinom(2,size=10,prob=c(0.4,0.3,0.2,0.1)) #random data from multinomial
     [,1] [,2]
[1,]    3    2
[2,]    5    5
[3,]    1    1
[4,]    1    2
```

So the first sample consists of 3A,5G,1C,1U. The second sample consists of 2A,5G,1C,2U. (Notice here we stored each sample in a column, not in a row).

To see the mean proportions of each type in 1000 random samples of 10 randomly selected positions, we first generate 1000 random samples from this multinomial distribution. Then we calculate the mean proportions from these 1000 samples in R as:

```
> gen.data<- rmultinom(1000, size=10,prob=c(0.4,0.3,0.2,0.1))
> apply(gen.data,1,mean)/10  #mean (over 1000 samples) AGCU counts divide 10
[1] 0.3990 0.3044 0.1991 0.0975
```

We can see that the empirical proportions from 1000 samples are very

close the theoretical proportions.

R syntax: the apply(matrix, 1, function) applies the function on the 1st dimension of the matrix (apply to each row).

Summary

In this lesson, we started with generating data from independent marginals in R. We then introduced bivariate normal distribution and multinomial distribution. We also obtain their pdf and generate random data from those distributions using R syntax. We also taught the R commands to display bivariate data and bivariate function (pdf) in 3D perspective plots. In the next lesson, we will reexamine examples with linear combinations of random variables.

4.3 Properties of Linear combination of the random variables.

For random variables $X_1,...,X_n$, $E(a_1X_1+...+a_nX_n)=a_1EX_1+...+a_nEX_n$.

Equivalently using the Sigma notation: $E(\sum_{k=1}^{n}a_kX_k)=\sum_{k=1}^{n}a_kEX_k$.

For <u>independent</u> random variables $X_1,...,X_n$,

$Var(a_1X_1+...+a_nX_n)=a_1^2Var(X_1)+...+a_n^2Var(X_n)$. Or using the Sigma notation:

$Var(\sum_{k=1}^{n}a_kX_k)=\sum_{k=1}^{n}a_k^2Var(X_k)$

Notice that the mean formula $E(\sum_{k=1}^{n}a_kX_k)=\sum_{k=1}^{n}a_kEX_k$ is always true, while the variance formula $Var(\sum_{k=1}^{n}a_kX_k)=\sum_{k=1}^{n}a_k^2Var(X_k)$ is only true for independent random variables.

Example 4.3.1.

Let $Y = 2X_1 - X_2 - 3X_3$ for three *independent* random variables X_1, X_2 and X_3.

The means are $E(X_1) = 1$, $E(X_2) = 2$ and $E(X_3) = 3$.

The standard deviations are $sd(X_1) = 4$, $sd(X_2) = 5$ and $sd(X_3) = 6$.

Then what are E(Y) and Var(Y)?

Solution:

Due to independence, we can just apply the formula for linear combinations.

$E(Y) = 2EX_1 - EX_2 - 3EX_3 = 2(1) - (2) - 3(3) = -9$.

$Var(Y) = 4\,\mathrm{var}(X_1) + \mathrm{var}(X_2) + 9\,\mathrm{var}(X_3) = 4(4^2) + 5^2 + 9(6^2) = 413$.

$sd(Y) = \sqrt{Var(Y)} = \sqrt{413}$. Notice that the formula is for variance. So we need to convert standard deviation to variance for calculation.

Example 4.3.2.

Let $Y = 2X_1 - X_2 - 3X_3$ for three *dependent* random variables X_1, X_2 and X_3.

The means are $E(X_1) = 1$, $E(X_2) = 2$ and $E(X_3) = 3$.

The standard deviations are $sd(X_1) = 4$, $sd(X_2) = 5$ and $sd(X_3) = 6$.

Then what are E(Y) and Var(Y)?

Solution:

The formula for the mean of linear combination is always true. Therefore,

$E(Y) = 2EX_1 - EX_2 - 3EX_3 = 2(1) - (2) - 3(3) = -9$.

However, the formula for the variance of linear combination only holds for independent random variables. Here the random varialbes are dependent. Therefore, Var(Y) is UNKNOWN. Its value dependents on the covariance between the variables, which are not given.

Example 4.3.3.

We have three independent random variables $X_1 \sim N(\mu=1, \sigma=2)$, $X_2 \sim \exp(\lambda=3)$, $X_3 \sim$ t-distribution with m=4 degrees of freedom. Let $Y = 2X_1 + 3X_2 - 5X_3$.

What are E(Y) and Var(Y)?

Solutions:

$E(X_1)=1, \quad Var(X_1)=2^2=4.$

$E(X_2)=\frac{1}{3}, \quad Var(X_2)=\frac{1}{3^2}=\frac{1}{9}.$

$E(X_3)=0, \quad Var(X_3)=\frac{4}{4-2}=2.$

Therefore, $EY = 2EX_1 + 3EX_2 - 5EX_3 = 2(1)+3(\frac{1}{3})-5(0)=3$, and

$Var(Y) = 4Var(X_1)+9Var(X_2)+25Var(X_3) = 4(4)+9(\frac{1}{9})+25(2)=67.$

(http://www.livescribe.com/cgi-bin/WebObjects/LDApp.woa/wa/MLSOverviewPage?sid=SxZ5l4g95DR6)

Example 4.3.3 (continued)

Can we check our answer E(Y)=3 and Var(Y)=67 with Monte Carlo method?

We can generate n=10,000 random samples of Y and calculate the sample mean and sample variances to see if the mean=3 and variance=67 are correct.

```
> x1<-rnorm(10000, mean=1, sd=2)
> x2<-rexp(10000, rate=3)
> x3<-rt(10000, df=4)
> y<-2*x1+3*x2-5*x3
> mean(y)
[1] 2.996921
> var(y)
[1] 65.30099
```

The empirical mean and variance from the Monte Carlo simulation are close to our answers.

More on Monte Carlo simulations:

When we do one simulation that generates a size n random samples from a distribution, we can calculate a statistic S_n from the random samples. However, this S_n value changes from one simulation to another, because the random samples are different from each simulation run. We can repeat the simulation nsim times (each time is a size n random samples), and calculate the mean and variances of the nsim S_n values. Then a 95% confidence interval for the true value of S is calculated as $mean(S_n) \pm 1.96 * \sqrt{\text{var}(S_n)/nsim}$. We will cover the confidence interval concept in a later module. For now, this gives a range of possible values of S from the Monte Carlo method.

Example 4.3.3. (continued)

$Y = 2X_1 + 3X_2 - 5X_3$, with

$X_1 \sim N(\mu = 1, \sigma = 2)$, $X_2 \sim \exp(\lambda = 3)$, $X_3 \sim$ t-distribution (df=4).

When we use Monte Carlo simulation to estimate the mean and variance of Y, we can also get correspondingly 95% confidence intervals.

We will store each simulation run in one column, so the results are a 10,000 by 1,000 matrix.

```
> x1<-matrix(rnorm(10000*1000, mean=1, sd=2),nrow=1000) #X1~normal
> x2<- matrix(rexp(10000*1000, rate=3), nrow=1000) #X2~exp(3)
> x3<- matrix(rt(10000*1000,df=4), nrow=1000) #X3~t(4)
> y<-2*x1+3*x2-5*x3 #calculate Y from X1, X2, X3
> mean.y<-apply(y,1,mean) #sample means in each of the 1000 runs
> var.y<- apply(y,1,var) #sample variances in each of the 1000 runs
> mean(mean.y) + c(-1,1)*1.96*sqrt(var(mean.y)/1000) #95%CI for mean
 [1] 2.993553 3.003757
> mean(var.y) + c(-1,1)*1.96*sqrt(var(var.y)/1000)  #95%CI for
variance
[1] 66.84934 67.54416
```

We can see that our answers from the theoretical formula E(Y)=3 and Var(Y)=67 both fall into the respective confidence interval (2.994,3.004) and (66.85,67.54). So this provides a confirmation on our theoretical calculation.

Example 4.3.4.

We have three copies of the same random variable $X \sim N(\mu=1, \sigma=2)$. That is, $X_1 = X_2 = X_3 = X$. Let $Y = X_1 + X_2 + X_3$.

Does $EY = EX_1 + EX_2 + EX_3 = 1+1+1 = 3$?

And does $Var(Y) = Var(X_1) + Var(X_2) + Var(X_3) = 2^2 + 2^2 + 2^2 = 12$?

Check these formulas by simulation.

Solutions: We can generate 1,000,000 random samples of Y as specified and calculate their mean and variance to check the formula.

```
> x<-rnorm(1000000,1,2)
> x1<-x2<-x3<-x
> y<-x1+x2+x3
> mean(y)
[1] 2.997389
> var(y)
[1] 36.0726
```

From the R results, EY=3 looks correct and Var(Y)=12 is wrong.

Notice that these three random variables are NOT independent. The mean formula is true for all random variables, but the variance formula is true only for independent variables. So the theoretical results agree with what we see from simulation.

In fact, since these three random variables are copies of the same variable X, Y=3X. So EY=3, EX=3(1)=3, Var(Y)=9Var(X)=9(4)=36. This agrees with what the simulations tell us.

4.4 Simulations for checking or deriving properties formulas.

We already see in the last section how to use Monte Carlo simulation to check our theoretical formulas. In practice, Monte Carlo simulations are also used often when we do not have explicit theoretical formulas.

We can do nsim simulation runs, providing nsim realizations of the quantity of interest S. Then the 95% confidence interval for the true quantity value S can be constructed $mean(S) \pm 1.96 * \sqrt{\text{var}(S)/nsim}$.

Example 4.4.1. If we have three independent random variables $X_1 \sim N(\mu=1, \sigma=2)$, $X_2 \sim \exp(\lambda=3)$, $X_3 \sim$ t-distribution with m=4 degrees of freedom.
Let Y=X1*X2+X3. What is the mean and variance of Y?

We cannot calculate these theoretically since we do not have formulas for mean and variance of the product of two random variables. (Such formulas do exist, but they are more complicated and we do not teach them here.)

However, the Monte Carlo simulation of Y is simple from the construction. We can approximate the mean and variances of Y from 10,000 random samples Y.

```
> x1<-rnorm(10000, mean=1, sd=2)
> x2<-rexp(10000, rate=3)
> x3<-rt(10000, df=4)
> y<-x1*x2+x3
> mean(y) #sample mean from one simulation run with n=10,000
[1] 0.3103503
> var(y) #sample variance from one simulation run with n=10,000
[1] 3.071038
```

Hence, approximately E(Y)≈0.3 and Var(Y)≈3.
We can further get interval estimates for E(Y) and Var(Y):

```
> x1<-matrix(rnorm(10000*1000,1,2),nrow=1000)
> x2<- matrix(rexp(10000*1000,3),nrow=1000)
> x3<- matrix(rt(10000*1000,4),nrow=1000)
```

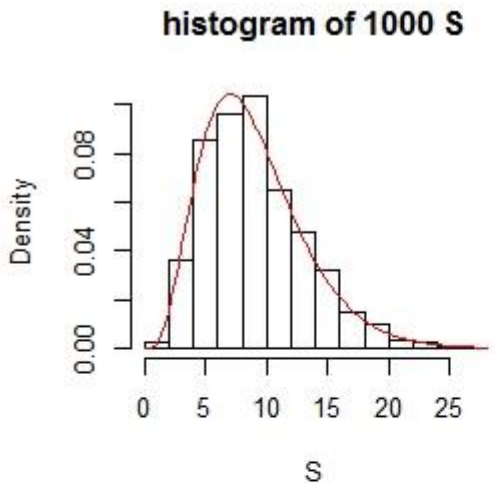
```
> y<-x1*x2+x3
> mean.y<-apply(y,1,mean)
> var.y<- apply(y,1,var)
> mean(mean.y) + c(-1,1)*1.96*sqrt(var(mean.y)/1000) #95%CI for mean
[1] 0.3318609 0.3340002
> mean(var.y) + c(-1,1)*1.96*sqrt(var(var.y)/1000) #95%CI for variance
[1] 2.993847 3.006208
```

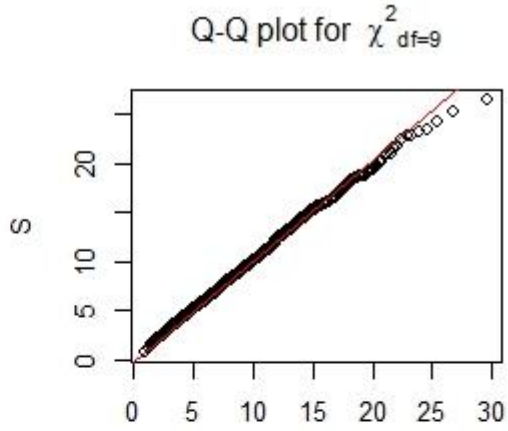
So from Monte Carlo simulation, we expect that E(Y) to fall between 0.332 and 0.334, and Var(Y) to fall between 2.99 and 3.01.

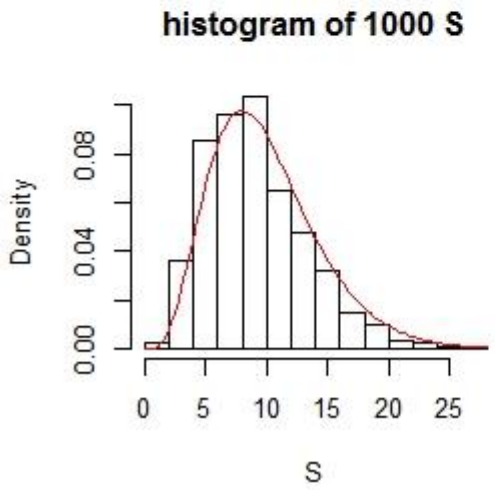
Example 4.4.2. Monte Carlo simulations can be used to derive or check the distribution of any statistic. If $X_1,...,X_{10}$ each follows the standard normal distribution $N(0,1)$, then a theorem in probability states that $S=\sum_{i=1}^{10}(X_i-\bar{X})^2$ follows a chi-square distribution with degrees of freedom=9. Here $\bar{X}=\frac{1}{10}\sum_{i=1}^{10}X_i$. We use a Monte Carlo simulation to check if this statement is true. We generate 1000 sets of $X_1,...,X_{10}$, therefore getting 1000 values of S. We then compare them to the theoretical chi-square distribution $\chi^2_{df=9}$. This can be done by comparing the histogram with the density curve. A better way is to check the quantile-quantile plot (qq-plot) by plotting quantiles of the observed data with the theoretical distribution's quantiles. If the data does come from this theoretical distribution, then the qq-plot would approximately follow a straight line.

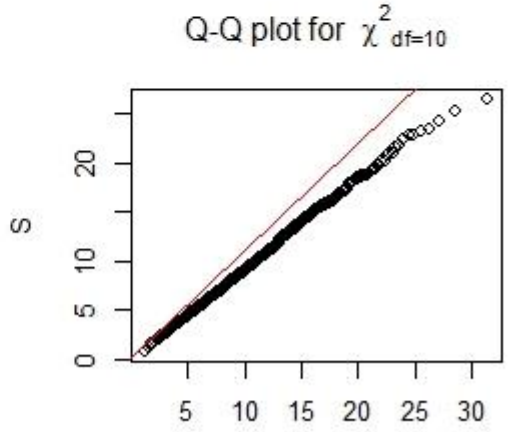
```
my.stat<-NULL #my.stat  -  vector to save results, empty now
for (i in 1:1000) {
     x.data<-rnorm(n=10)  #10 samples from N(0,1)
     my.stat<-c(my.stat, sum((x.data-mean(x.data))^2)) #add S to
my.stat
}
par(mfrow=c(2,2))
hist(my.stat, freq = FALSE, main="histogram of 1000 S",xlab="S")
#simulated distribution
curve(dchisq(x,df=9),col=2,add=T) #overlay chi-square(9) density
curve
qqplot(qchisq(ppoints(1000), df = 9), my.stat, main = expression("Q-Q
plot for" ~~ {chi^2}[df == 9]), xlab="",ylab="S") #qq-plot to check
fit to chi-square(9)
qqline(my.stat, distribution = function(p) qchisq(p, df = 9),
datax=TRUE, col=2) #add qq-line for chi-square(9)
hist(my.stat, freq = FALSE, main="histogram of 1000 S",xlab="S")
curve(dchisq(x,df=10),col=2,add=T) #overlay chi-square(10) density curve
qqplot(qchisq(ppoints(1000), df = 10), my.stat, main = expression("Q-Q
```

```
plot for" ~~ {chi^2}[df == 10]),xlab="",ylab="S") #qq-plot to check fit to chi-square(10)
qqline(my.stat, distribution = function(p) qchisq(p, df = 10), datax=TRUE, col=2) #add qq-line for chi-square(10)
```









In the plots, we compared the histogram with pdf of chi-square distributions $\chi^2_{df=9}$ and $\chi^2_{df=10}$. We can see the histogram is close to both pdf's. However, it is clearer in the qq-plots that $\chi^2_{df=9}$ is closer to the data distribution. Generally to check the distributions, we use the qq-plots instead of the histogram.

Also, the plots are not the same for each simulation. You may wish to run a few more times and observe the pattern.

4.5 Central Limit Theorem.

We suppose that $X_1,\ldots,X_n$ is a random sample of size n from a distribution with mean=μ and variance=σ^2. Then the sample mean $\bar{X}=\frac{1}{n}\sum_{i=1}^{n}X_i$ is a linear combination of $X_1,\ldots,X_n$.

From the properties of linear combination of random variables, $\bar{X}$ has mean $\frac{1}{n}\sum_{i=1}^{n}\mu = \mu$ and variance $(\frac{1}{n})^2\sum_{i=1}^{n}\sigma^2 = \frac{\sigma^2}{n}$.

Central Limit Theorem (CLT):

$X_1,\ldots,X_n$ is a random sample of size n from a distribution with mean=μ and variance=$\sigma^2<\infty$. Then for large sample size n, $\bar{X}$ is approximately normal. In other words, $\sqrt{n}(\bar{X}-\mu)$ converges in distribution to $N(mean=0,\ \ sd=\sigma)$.

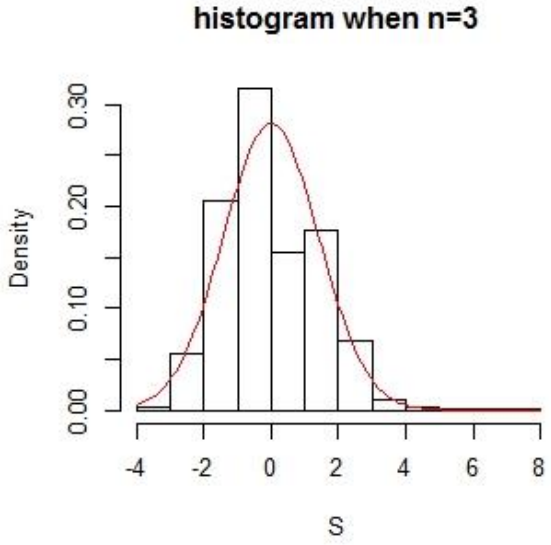
Example 4.5.1.

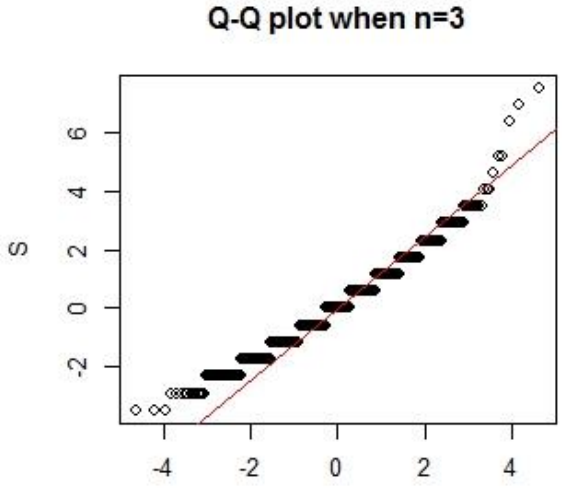
We can use the Monte Carlo simulations to check the statement of the CLT. Let the random sample $X_1,...,X_n$ comes from Poisson distribution with parameter $\lambda = 2$ so that mean=2 and variance=2.

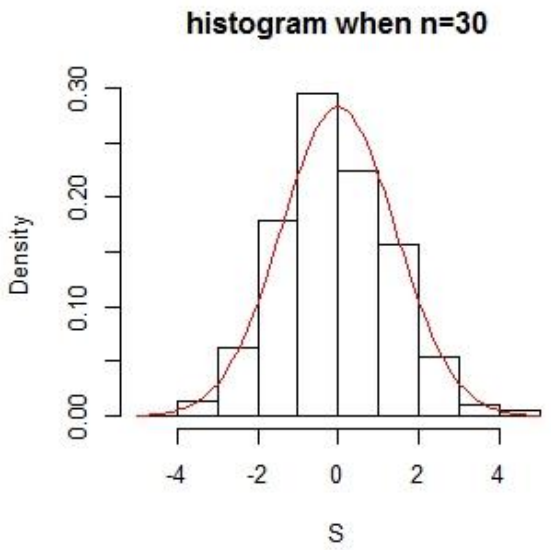
We compare the histogram of 1000 samples of $\sqrt{n}(\bar{X}-2)$ with the pdf of $N(mean=0, sd=\sqrt{2})$, and also do the qq-plot.

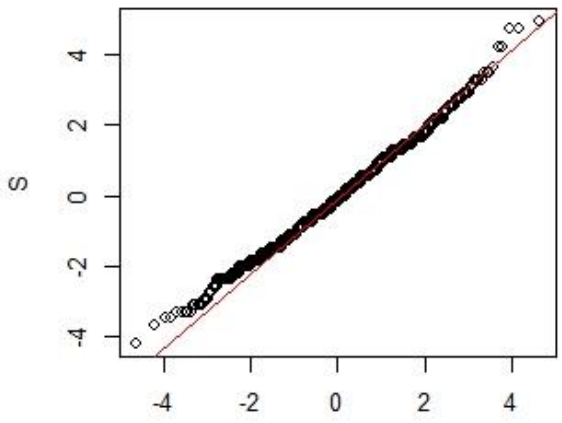
We do this at n=3 and n=30.

It is clear in the plot that when n=30, $\sqrt{n}(\bar{X}-2)$ is approximately $N(mean=0, sd=\sqrt{2})$. When n=3, it is not.









```
my.data.3<-matrix(rpois(3*1000,2),nrow=3)
my.stat.3<-sqrt(3)*(apply(my.data.3,2,mean)-2)
my.data.30<-matrix(rpois(30*1000,2),nrow=30)
my.stat.30<-sqrt(30)*(apply(my.data.30,2,mean)-2)
par(mfrow=c(2,2))
hist(my.stat.3, freq = FALSE, main="histogram when n=3",xlab="S")
curve(dnorm(x,mean=0,sd=sqrt(2)),col=2,add=T)
qqplot(qnorm(ppoints(1000), mean=0,sd=sqrt(2)), my.stat.3, main =
"Q-Q plot when n=3",xlab="",ylab="S")
qqline(my.stat.3, distribution = function(p)
qnorm(p,mean=0,sd=sqrt(2)), datax=TRUE,col=2)
hist(my.stat.30, freq = FALSE, main="histogram when n=30",xlab="S")
curve(dnorm(x,mean=0,sd=sqrt(2)),col=2,add=T)
qqplot(qnorm(ppoints(1000), mean=0,sd=sqrt(2)), my.stat.30, main =
"Q-Q plot when n=30",xlab="",ylab="S")
qqline(my.stat.30, distribution = function(p)
qnorm(p,mean=0,sd=sqrt(2)), datax=TRUE,col=2)
```

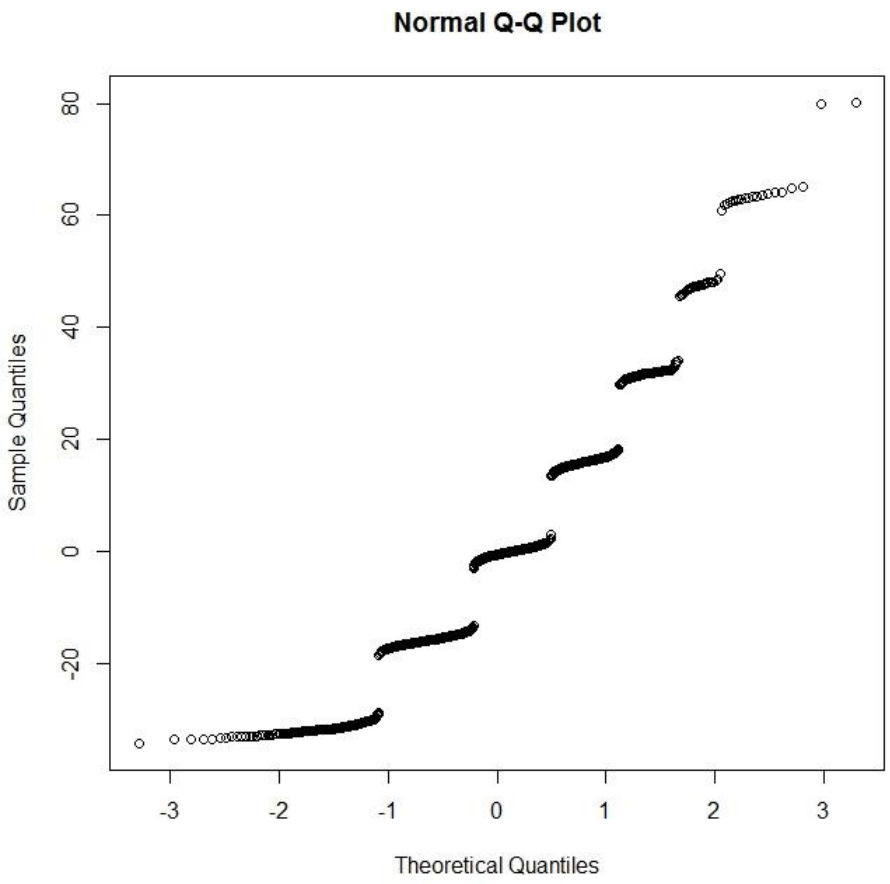
There is another app illustrating the CLT at
https://adamding.shinyapps.io/CLTadamding/
It is built using the Shiny package in R. You can try sample means from different distributions with varying sample size from 1 to 100. Drag the slider bar to vary the sample size n. You can see that as n increases, the distribution of sample mean will get closer to normal distribution, no matter which initial distribution you choose.

Example 4.5.2.

The random sample condition in CLT is necessary. We consider the following counter-example. Let n be an even number. Assume that, instead of being a random sample, the first half of $X_1,\ldots,X_n$ is just n/2 replicates of one random variable: $X_1 = X_2 = \ldots = X_{n/2}$ from Poisson distribution with parameter $\lambda = 2$. And the second half $X_{n/2+1},\ldots,X_n$ is a random sample of size n/2 from Poisson distribution with parameter $\lambda = 2$.

From the properties of linear combination of random variables, $\bar{X}$ has mean $\frac{1}{n}\sum_{i=1}^{n}\mu = \mu$ and variance $(\frac{1}{n})^2(\frac{n}{2})^2\sigma^2 + (\frac{1}{n})^2\sum_{i=\frac{n}{2}+1}^{n}\sigma^2 = (\frac{1}{4}+\frac{1}{2n})\sigma^2 \neq \frac{\sigma^2}{n}$.

And it is also NOT approximately normal even for large n. To see this, we can take n=1000 and do a Monte Carlo simulation, with $X_1 = X_2 = \ldots = X_{500}$, $X_{501},\ldots,X_n$ be random samples from the Poisson($\lambda = 2$) distribution.



The data is clearly not normal when we increase n to n=1000.

```
my.data<-rbind(matrix(rep(rpois(1000,2),rep(500,1000)),nrow=500),
matrix(rpois(500*1000,2),nrow=500))
qqnorm(my.stat)
```

Applications of the CLT

The Central Limit Theorem allows us to calculate the probabilities about the sample means using the normal distribution, regardless of the actual distribution of the random variables.

Example 4.5.3.

When n is large, we can use CLT to calculate probability of events about the sample mean $\bar{X}$ regardless of the exact distribution of $X_1,\ldots,X_n$. For example, if $X_1,\ldots,X_{120}$ is a random sample of size n=120 from a distribution with mean=3 and variance=9.

Then we know that $P(2.8<\bar{X}\le 3.1)=P(\bar{X}\le 3.1)-P(\bar{X}\le 2.8)$ can be approximately calculated in R as

```
> pnorm(3.1,mean=3,sd=3/sqrt(120))-pnorm(2.8,mean=3,sd=3/sqrt(120))
[1] 0.4098953
```

Hence $P(2.8<\bar{X}\le 3.1)\approx 0.41$

Example 4.5.4.

Assume the four types of nucleotide bases A,G,C,T occurs with proportions 0.3,0.3,0.2 and 0.2 in a DNA sequence. If the nucleotides on different positions on the DNA are all independent, then the number of the nucleotides of type G among any randomly selected subsequence of length 6 follows a binomial distribution with size=6 and probabilities=0.3. Let us randomly choose 80 subsequences each of length 6. What is the probability that the average number of base G within the subsequences is between 2 and 2.5?

Solution:

$X_1,...,X_{80}$ is a random sample of size n=80 from the binomial(size=6,p=0.3) distribution. The mean and variance of the binomial distribution is mean=6*0.3=1.8 and variance=6*0.3*0.7=1.26.

Hence $P(2<\bar{X}\leq 2.5)$ can be approximated, by CLT, with

```
>
pnorm(2.5,mean=1.8,sd=1.26/sqrt(80))-pnorm(2,mean=1.8,sd=1.26/sqrt
(80))
[1] 0.07784344
```

Hence there is about 8% chance the average number of base G, within 80 subsequences each of length 6, will be between 2 and 2.5.

Unit 4. Summary.

In this module, we focused on the use of Monte Carlo Methods and the Central Limit Theorem to aid in calculating the mean and variances of various distributions, including univariate and multivariate distributions. Next week we will look at statistical inferences and to estimate a parameter from data.