Module 2

This is a single, concatenated file, suitable for printing or saving as a PDF for offline viewing. Please note that some animations or images may not work.

Module 2 Study Guide and Deliverables

Topics:	**Lecture 3:** Analyzing Risk: An Introduction to Modeling Uncertain Inputs **Lecture 4:** Analyzing Risk: incorporating Uncertainty into the Decisions of the Enterprise
Readings:	Lectures 3 and 4 online content
Discussions:	**Module 2 Discussion**
Assignments:	Team and case assignments **Tutorial:** 2 **Assignment 2**: Individual Assignment covering Lecture 3 and Lecture 4.
Assessments:	**Quiz 2**

Lecture 4

Learning Objectives

After you complete this lecture, you will be familiar with the following:

- Percentiles of the simulation output data
- Constructing confidence intervals for the mean
- Determining number of iterations in a simulation study

- Sensitivity analysis using R
- Creating and interpreting an overlay chart in R
- Creating and interpreting a trend chart in R
- Creating and interpreting a box whisker chart in R

New-Product Development Model – Deterministic Model[1]

In this module, we will be working with the following new-product development model:

> Suppose that ACG Pharmaceuticals has discovered a potential drug breakthrough in the laboratory and needs to decide whether to go forward to conduct clinical trials and seek FDA approval to market the drug. Total R&D costs are expected to reach $700 million, and the cost of clinical trials will be about $150 million.
>
> The current market size is estimated to be 2 million people and is expected to grow at a rate of 3% each year. In the first year, ACG estimates gaining an 8% market share, which is anticipated to grow by 20% each year. As new competitors are expected to enter the market, it is difficult to estimate beyond 5 years. A monthly prescription is anticipated to generate revenue of $130 while incurring variable costs of $40.
>
> A discount rate of 9% is assumed for computing the net present value of the project.
>
> The company needs to know how long it will take to recover its fixed expenses and the net present value over the first 5 years.
>
> **Source:** Evans, J. R. (2016). *Business analytics: Methods, models, and decisions* (2nd edition). Pearson Education, Inc., Chapter 11, Example 11.8, pp. 351-352.

"The model is based on a variety of assumptions, estimates, and known data."[2] "In reality, most of the data used in the model are uncertain; e.g., the market size and market-share growth rates."[3] We will first build a spreadsheet model based on the known data. Then, we will incorporate the uncertainties into our model and analyze the risk associated with the project.

Question: How do we come up with the data given in the problem description?

Answer: Most inputs are obtained from historical data, estimates based on prior market research, and expert opinion. For now, we will take these numbers as given but later we will learn how to come up with these numbers using historical data (Lecture 5).

Individual Exercise:

Before you read further, please try to build the R model on your own.

Figure 4.1 presents the resulting R code and resulting model of the new-product development model.

Figure 4.1

```
library(tidyverse)
library(FinancialMath)

RnD<-700 #Reasearch and Development costs (millions)
CT<-150 #Clinical trial costs (millions)

ny<-5 #number of years in model

MSz<-2 #Market Size In year 1
MSzG<-.03 #Market Size Growth Rate

MSh<-.08 #Market Share
MShG<-.2 #Market Share Growth Rate

UnitRev<-130*12 #Annual Revenue/Customer
UnitCost<-40*12 #Annual Cost/Customer

r<-.09 #required rate of return for calculating NPV

model<-data.frame(year=1:ny)

output<-vector("double",nrow(model)) #create a vector of Market Sizes to eventually copy into the model

output[[1]]<-MSz

for (i in 2:ny) { #loop through and multiply previous Market size by the amount by which it grew
  output[[i]]<-output[[i-1]]*(1+MSzG)
}

model$MSz<-output #load vector of market sizes into model

output[[1]]<-MSh

for (i in 2:ny) { #repeat to incorporate market shares
  output[[i]]<-output[[i-1]]*(1+MShG)
}

model$MSh<-output #load market shares into model

model<-mutate(model,Sales=MSz*MSh,
              Revenue=Sales*UnitRev,
              Costs=Sales*UnitCost, #include Sales Revenues and Costs in the model (millions of $)
              Profit=Revenue-Costs,
              CumProfit=cumsum(Profit)-RnD-CT) #calculate annual profit and cumulative profit (using cumsum)


NPV(cf0=-(RnD+CT),cf=model$Profit,i=r,times=1:ny) #calculate the NPV of the cashflows
```

```
> model
  year      MSz      MSh     Sales  Revenue    Costs   Profit CumProfit
1    1 2.000000 0.080000 0.1600000 249.6000  76.8000 172.8000 -677.2000
2    2 2.060000 0.096000 0.1977600 308.5056  94.9248 213.5808 -463.6192
3    3 2.121800 0.115200 0.2444314 381.3129 117.3271 263.9859 -199.6333
4    4 2.185454 0.138240 0.3021172 471.3028 145.0162 326.2865  126.6532
5    5 2.251018 0.165888 0.3734168 582.5302 179.2401 403.2902  529.9434
```

We start the process by loading the tidyverse library and the FinancialMath library (for the NPV function), and defining parameters:

Figure 4.2

```
library(tidyverse)
library(FinancialMath)

RnD<-700 #Reasearch and Development costs (millions)
CT<-150 #Clinical trial costs (millions)

ny<-5 #number of years in model

MSz<-2 #Market Size In year 1
MSzG<-.03 #Market Size Growth Rate

MSh<-.08 #Market Share
MShG<-.2 #Market Share Growth Rate

UnitRev<-130*12 #Annual Revenue/Customer
UnitCost<-40*12 #Annual Cost/Customer

r<-.09 #required rate of return for calculating NPV
```

We then create vectors for market size and market share, using for loops. A for loop in R will iterate over a vector specified in the parentheses, in this case it runs for each number in the set {2,3,4,5}.

For each of these numbers, it does what's specified in the braces. Here it assigns to the second element in output 2*1.03, then to the third element (2*1.03)*1.03, and so forth.

This process is repeated for market share, expressed as a percent of the total market size.

Figure 4.3

```
output<-vector("double",nrow(model)) #create a vector of Market Sizes to eventually copy into the model

output[[1]]<-MSz

for (i in 2:ny) { #loop through and multiply previous Market size by the amount by which it grew
  output[[i]]<-output[[i-1]]*(1+MSzG)
}

model$MSz<-output #load vector of market sizes into model

output[[1]]<-MSh

for (i in 2:ny) { #repeat to incorporate market shares
  output[[i]]<-output[[i-1]]*(1+MShG)
}

model$MSh<-output #load market shares into model
```

We then use the mutate function to determine the calculated values for our model. Note the use of the **cumsum** function, which takes a vector and returns a vector of the same size, where each element is the sum of all the elements of the input vector to that point.

We use this to calculate cumulative profit, subtracting the one-time costs of R&D and clinical trials. Figure 4.4 displays the code and results.

Figure 4.4

```
model<-mutate(model,Sales=MSz*MSh,
              Revenue=Sales*UnitRev,
              Costs=Sales*UnitCost, #include Sales Revenues and Costs in the model (millions of $)
              Profit=Revenue-Costs,
              CumProfit=cumsum(Profit)-RnD-CT) #calculate annual profit and cumulative profit (using cumsum)
```

```
> model
  year       MSz      MSh     Sales  Revenue    Costs   Profit CumProfit
1    1 2.000000 0.080000 0.1600000 249.6000  76.8000 172.8000 -677.2000
2    2 2.060000 0.096000 0.1977600 308.5056  94.9248 213.5808 -463.6192
3    3 2.121800 0.115200 0.2444314 381.3129 117.3271 263.9859 -199.6333
4    4 2.185454 0.138240 0.3021172 471.3028 145.0162 326.2865  126.6532
5    5 2.251018 0.165888 0.3734168 582.5302 179.2401 403.2902  529.9434
```

Finally, we use the **NPV** function, incorporating the fixed costs at the start of the project and the cash flows represented by profit to determine the net present value of the deterministic model. The NPV function takes a cash flow at time 0, a vector of cash flows, and a required rate of return. We find it to be $185.4 million.

Figure 4.5

```
> NPV(cf0=-(RnD+CT),cf=model$Profit,i=r,times=1:ny) #calculate the NPV of the cashflows
[1] 185.4049
```

We are now done with building our model.

Look at the CumProfit column where we calculate the cumulative net profit. We observe that the cumulative net profit is negative for the first three years and it becomes positive in the forth year. Thus, we conclude that the product becomes profitable by the fourth year. However, note that much of the data used in this model should be *uncertain.* For instance, the market size, R&D costs, clinical trial costs, etc. may likely not end up at the single value that we chose for them. Thus, it is natural to ask **"*what is the risk associated with this project?*"**

"More specifically, we are interested in answering the following questions:

1. What is the risk that the net present value over the 5 years will not be positive?
2. What are the chances that the product will show a cumulative net profit in the third year?
3. What cumulative profit in the fifth year are we likely to observe with a probability of at least 0.9?"

Source: Evans, J. R. (2016). *Business analytics: Methods, models, and decisions* (2nd edition). Pearson Education, Inc., Chapter 11, Example 11.8, p. 388.

To be able to answer these questions, we will need to simulate the new-product development model. We will build the simulation model in the next section.

Building the Simulation Model[4]

In building our simulation model we will follow the 5 steps that were outlined in Lecture 2. We are already done with Step 1, i.e., we have started with the deterministic model and constructed a model that relates inputs to the output measures.

We will now define the inputs that are uncertain along with their probability distributions, then select one or more outputs to record over the simulation runs, determine number of trials for the simulation, execute the simulation, and finally analyze the results and gain insights.

Defining Inputs

Suppose that we identify the following uncertain variables in the model along with their probability distributions.

- *Market size:* normal with a mean of 2,000,000 units and a standard deviation of 400,000 units.
- *R&D costs:* uniform between $600,000,000 and $800,000,000 (so any value between $600,000,000 and $800,000,000 is equally likely for the R&D costs).
- *Clinical trial costs:* lognormal with a mean of $150,000,000 and standard deviation $30,000,000.
- *Annual market growth factor:* triangular with minimum 2%, maximum 6%, and most likely 3%.
- *Annual market share growth rate:* triangular with minimum 15%, maximum 25%, and most likely 20%.

Source: Evans, J. R. (2016). *Business analytics: Methods, models, and decisions* (2nd edition). Pearson Education, Inc., Chapter 11, Example 11.8, p. 388.

User-Defined Functions

We'll start by loading the tidyverse library, and defining the rtri function and the rlnorm2 function we used in the previous lecture.

Figure 4.6

```
library(tidyverse)

rtri<-function(n,min,ml,max){ #we start with functions to generate triangular random
  cvt<-function(U){           #variables and lognormal random variables, from Lecture 3
    F<-(ml-min)/(max-min)
    if (U<F) {min+(U*(max-min)*(ml-min))^.5}
    else {max-((1-U)*(max-min)*(max-ml))^.5}
  }
  y<-runif(n)
  sapply(y,cvt)
}

rlnorm2<-function(n,mean,sd){
  rlnorm(n,log(mean*(1+sd^2/mean^2)^-.5),log(1+sd^2/mean^2)^.5)
}
```

Predetermined Inputs

According to the text of the problem, the number of years, required rate of return, year 1 market share, unit revenue, and unit cost all remain fixed, as in the deterministic problem. We also choose here to run 10,000 trials.

Figure 4.7

```
n<-10000 #number of iterations to run
ny<-5 #number of years in model
r<-.09 #required rate of return for calculating NPV
MSh<-.08 #starting market share remains at .08

UnitRev<-130*12 #Annual Revenue/Customer
UnitCost<-40*12 #Annual Cost/Customer
```

Uncertain Inputs

Here we start to build our data frame, each row of which represents one trial of the Monte Carlo simulation. We include a trial number, R&D costs, clinical trial costs, and the starting market size, all as defined by the parameters of the problem.

The first ten trials are displayed in Figure 4.8, using the head function in the console.

Figure 4.8

```
> head(smodel,10)
   trial      RnD       CT      MSz
1      1 665.3620 160.9290 1.951991
2      2 683.4122 106.0716 1.478591
3      3 683.7225 226.4441 2.094842
4      4 688.6522 153.6253 2.108232
5      5 690.8385 153.8873 2.933164
6      6 664.9409 176.8833 2.588457
7      7 721.1163 147.9164 1.779019
8      8 614.3408 157.6258 1.212151
9      9 772.6851 114.8244 2.050520
10    10 700.5592 160.3479 2.120938
```

```
smodel<-data.frame(trial=1:n,RnD=runif(n,600,800),CT=rlnorm2(n,150,30),MSz=rnorm(n,2,.4))
```

We next create a list of data frames, which we'll eventually nest in our stochastic model. Each of these data frames contains the initial uncertain inputs that we need to define for each year: the growth of the market size and market share for years two through five.

To produce this list, we use the **replicate** function, telling it to define a data frame 10,000 times. The result of the first replication is reproduced in Figure 4.9.

To simplify our code, we take the step of adding 1 to the growth factors now.

Figure 4.9

```
> output[1]
[[1]]
  year     MSzG     MShG
1    1 1.000000 1.000000
2    2 1.034102 1.190675
3    3 1.027446 1.234398
4    4 1.028500 1.228915
5    5 1.033171 1.220813
```

```
output<-replicate(n = n, #model variables that change year over year
                  expr = {data.frame(year=1:ny,
                                     MSzG=c(1,rtri(4,1.02,1.03,1.06)),
                                     MShG=c(1,rtri(4,1.15,1.2,1.25)))},
                  simplify = F)
```

Calculating Outputs

We now use a for loop to go through each line of smodel and calculate sales for each year, profit for each year, and cumulative profit for each year in each of the 10,000 trials in "output".

To prevent R from repeatedly overwriting the entirety of output, we use a temporary data frame named “a” to store these values while the calculation is being performed. Note the cumproduct function, which has the same functionality of cumsum for multiplication.

We remove “a” when the process is finished. The first entry in the newly calculated “output” is reproduced in Figure 4.10.

Figure 4.10

```
> output[1]
[[1]]
  year     MSzG     MShG     Sales   Profit CumProfit
1    1 1.000000 1.000000 0.1561593 168.6521 -657.6390
2    2 1.034102 1.190675 0.1922756 207.6577 -449.9813
3    3 1.027446 1.234398 0.2438589 263.3676 -186.6137
4    4 1.028500 1.228915 0.3082227 332.8806  146.2669
5    5 1.033171 1.220813 0.3887639 419.8650  566.1319
```

```
for (row in 1:n){ #incorporate calculated values from yearly rv's
  a<-output[[row]]
  a<-mutate(a,Sales=MSh*smodel$MSz[[row]]*cumprod(MSzG)*cumprod(MShG),
         Profit=Sales*(UnitRev-UnitCost),
         CumProfit=cumsum(Profit)-smodel$RnD[[row]]-smodel$CT[[row]])
  output[[row]]<-a
}
rm(a)
```

We now nest our finished output lists into our original “smodel” data frame, under the name “nmodel”, and remove the now redundant “output” list.

We also create a quick user-defined function for NPV, that will be used to calculate the profit cash flows in nmodel. This is necessary because the NPV function in the FinancialMath library cannot readily be used with sapply. The sapply function will be explained later in this section.

Our version of the NPV function calculates each year’s total discount, then takes the cross product (like sumproduct in Excel) of this and the list of cashflows.

Figure 4.11

```
smodel$nmodel<-output #include
rm(output) #up until now, we n

NPV<-function(cfs,r){ #quick f
  yr<-1:length(cfs)
  disc<-(1+r)^-yr
  crossprod(unlist(cfs),disc)
}
```

lapply, sapply, and vapply

One very useful feature of R is that it can call other functions *as one of their arguments.* **lapply**, **sapply**, and **vapply** all take advantage of this property. As arguments, they take a vector and a function. They return, depending on the function, either a list or a vector: the result of the function input being called on each element of the vector input. You can also pass additional arguments for the function input into the apply function.

These functions can duplicate the functionality of a for loop, but in a way that's easier to read and code.

Individual Exercise:

Before reading further, enter "?sapply" into the console to bring up the R documentation for these functions and try some of the examples listed until you get the hang of it.

On this line of code, we define a function just to extract the Profit vector from a data frame, and lapply it over smodel$nmodel (the list of nested data frames) to return a list of 10,000 lists of the profit the company makes each year.

We then take this list, and sapply the NPV function to it, resulting in a list of NPVs for years one through five.

Subtracting the initial cash outlays of RnD and CT shows us the total net present value for each trial. This value gets assigned as a new entry in smodel called "NPV".

This does not print to the console nicely, but the first couple entries from View(smodel) are displayed in figure 4.12.

Figure 4.12

trial	RnD	CT	MSz	nmodel	NPV
1	665.3620	160.92900	1.951991	*6 variables*	215.289493
2	683.4122	106.07159	1.478591	*6 variables*	5.106192
3	683.7225	226.44413	2.094842	*6 variables*	161.954379
4	688.6522	153.62526	2.108232	*6 variables*	309.724353

```
smodel$NPV<-lapply(smodel$nmodel,function(model){
  model$Profit}) %>%
  sapply(NPV,r)-smodel$RnD-smodel$CT #calculate th
```

Finally, we create a new data frame, called “ymodel”, where we unnest nmodel within smodel. This results in a data frame with 50,000 observations, putting each year from each trial in its own row. This data frame will be very helpful for calculating statistics and visualizing our results. The ten rows representing the first two trials are reproduced in Figure 4.13.

Figure 4.13

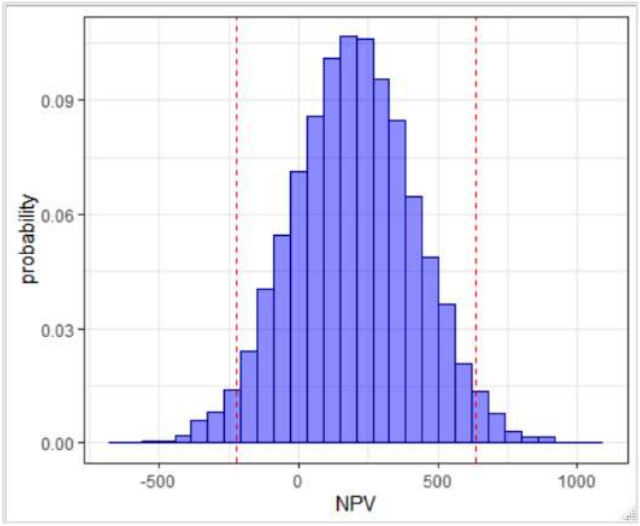
```
> head(ymodel,10)
   trial      RnD       CT      MSz        NPV year     MSzG     MShG     Sales   Profit  CumProfit
1      1 665.3620 160.9290 1.951991 215.289493    1 1.000000 1.000000 0.1561593 168.6521 -657.63896
2      1 665.3620 160.9290 1.951991 215.289493    2 1.034102 1.190675 0.1922756 207.6577 -449.98126
3      1 665.3620 160.9290 1.951991 215.289493    3 1.027446 1.234398 0.2438589 263.3676 -186.61370
4      1 665.3620 160.9290 1.951991 215.289493    4 1.028500 1.228915 0.3082227 332.8806  146.26686
5      1 665.3620 160.9290 1.951991 215.289493    5 1.033171 1.220813 0.3887639 419.8650  566.13189
6      2 683.4122 106.0716 1.478591   5.106192    1 1.000000 1.000000 0.1182873 127.7503 -661.73353
7      2 683.4122 106.0716 1.478591   5.106192    2 1.023939 1.210318 0.1465925 158.3199 -503.41367
8      2 683.4122 106.0716 1.478591   5.106192    3 1.051856 1.213536 0.1871201 202.0897 -301.32392
9      2 683.4122 106.0716 1.478591   5.106192    4 1.038116 1.212576 0.2355458 254.3894  -46.93451
10     2 683.4122 106.0716 1.478591   5.106192    5 1.049556 1.197880 0.2961381 319.8292  272.89466
```

```
ymodel<-smodel%>%
  unnest(nmodel)
```

Analyzing the Outputs

We are now ready to start analyzing the risk profile of the project, in terms of cumulative net profit and net present value. We’ll start by creating a histogram of the net present value. We add dashed lines at the 97.5% quantile and 2.5% quantile of our data using geom_vline. 95% of the trials fell somewhere between these two values. In this case, we had 9,500 trials where the profit was between –$224 million and $637 million.

Figure 4.14




```
ggplot(smodel)+geom_histogram(aes(x=NPV,y=..count../sum(..count..)), #the resultin
                              color="dark blue",fill="blue",alpha=.4)+
  ylab("probability")+
  geom_vline(xintercept=quantile(smodel$NPV,.025),color="red",linetype="dashed")+
  geom_vline(xintercept=quantile(smodel$NPV,.975),color="red",linetype="dashed")+
  theme_bw()
```

The cat Function

We then use the cat function to print to the console some information about our results. The cat function prints a string to the console, consisting of each unlabeled argument you pass the function. You can control what comes between each argument with the “sep” argument, which defaults to a single space. The cat function also allows you to put line breaks in the result wherever it reads a “\n”. Some examples are produced in Figure 4.15.

Individual Exercise:

Practice using the cat function until you feel comfortable using it. Try looking into how to use it in conjunction with the “paste” function.

Figure 4.15

```
> cat("Hello,\nworld!\n")
Hello,
world!
> cat("Hello,","world!",sep="XXXXX")
Hello,XXXXXworld!
```

We use cat to print out statistics about our model. The results can be seen in Figure 4.16.

Figure 4.16

```
cat("The mean NPV is ",mean(smodel$NPV),
    "\nThe standard deviation of NPV is ", sd(smodel$NPV),
    "\nThe likelihood of a negative NPV is ", sum(smodel$NPV<0)/n,
    "\nThe likelihood that the project will break even by the third year is ",
       sum(ymodel$CumProfit>0 & ymodel$year==3)/n,
    "\nWe can be 90% sure the cumulative net profit in year 5 will be at least ",
       (ymodel%>%filter(year==5))$CumProfit %>% quantile(.1),
    "\nThe 95% confidence interval about the mean is [",
       mean(smodel$NPV)-qnorm(.975)*sd(smodel$NPV)/n^.5,", ",
       mean(smodel$NPV)+qnorm(.975)*sd(smodel$NPV)/n^.5,"]",
    "\nThe number of iterations needed to produce a 95% confidence interval with a half-width of 3,000,000 is ",
       (qnorm(.975)*sd(smodel$NPV)/3)^2,
    sep='')
```

```
The mean NPV is 199.7435
The standard deviation of NPV is 219.9272
The likelihood of a negative NPV is 0.1838
The likelihood that the project will break even by the third year is 0.0897
We can be 90% sure the cumulative net profit in year 5 will be at least 180.3223
The 95% confidence interval about the mean is [195.433, 204.054]
The number of iterations needed to produce a 95% confidence interval with a half-width of 3,000,000 is 20644.85
```

Let's look at the first three lines of that output, and how we got them. For the mean and standard deviation of the NPV, we simply used the mean and sd functions in R.

To determine the likelihood of a negative NPV, we first looked at smodel$NPV<0. This returns a vector of 10,000 Booleans, which we take the sum of, coercing TRUE to 1 and FALSE to 0 in the process.

Figure 4.17

```
sum(smodel$NPV<0)/n
```

Dividing by the number of trials gives us the proportion of trials where the NPV was less than 0.

The next line of the output tells us how likely the project is to break even by the third year. To determine that, we need the cumulative profit in the third year.

For this we can use our unnested "ymodel", filtered to year 3.

This is also a probability, so we use the same trick of coercing Booleans to integers then dividing by the number of trials.

Figure 4.18

```
sum(ymodel$CumProfit>0 & ymodel$year==3)/n,
```

The next line of the output gives us the tenth percentile of cumulative profit in year 5.

We know that 90% of trials had a larger cumulative profit than this number.

Figure 4.19

```
(ymodel%>%filter(year==5))$CumProfit %>% quantile(.1),
```

Percentiles as Measures of Risk

"The k^{th} percentile is a value at or below which at least k percent of the observations lie."[4]

Percentiles are typically used to measure the risk and allow us to answer questions such as

- How much money might we lose with, say, 10% probability?
- What profit are we likely to realize with a probability of 95%?
- What is the probability that we make at least, say, $100,000?

In our case, Figure 4.16 tells us there's a 90% chance the cumulative profit in year 5 will be greater than or equal to $180 million. This means there's a 10% chance it will be less than or equal to $180 million, so this is the 10th percentile of the cumulative profit in year 5. We'll see how to calculate this below.

- The 25th, 50th, and 75th percentile are called the first quartile, the median, and the third quartile, respectively.
- The 5th percentile is often called the **value at risk at the 5% level** or **VaR 5%** and the 95th percentile is often called the **value at risk at the 95% level** or **VaR 95%.** Which one to use depends on if you're modeling revenue, profit, or costs. These measures are typically used as a benchmark for how bad things might be in a particularly unlucky scenario. If your model is accurate, 19 times out of 20 you should be better off than the VaR 5%.

Individual Exercise:

Determine the VaR 5% for cumulative net profit in Year 5.

Confidence Intervals for the Mean

When we run a Monte Carlo simulation, we're not seeing the true distribution of the random variable we're studying (NPV for instance). In fact, we're not even seeing the true distribution of the *simulated* random variable. What we can do is make inferences about that simulated random variable.

Thus, we can say the mean of the sample (represented by our trials) is a specific number, but we can't say that the mean of the simulated random variable is equal to that number. We can, however, make inferences about that mean.

In particular, recall from the last lecture that the normal distribution arises from a large sum of identically distributed random variables.

$$\Sigma X_i \rightarrow N(\mu_X, n\sigma_X^2)$$

and

$$\frac{\bar{x} - \mu_X}{s_X/\sqrt{n}} \rightarrow N(0,1)$$

since again, with a large sample, $s_X \rightarrow \sigma_X$. Since the standard normal distribution is symmetrical about 0, we can also say that

$$\frac{\mu_X - \bar{x}}{s_X/\sqrt{n}} \rightarrow N(0,1)$$

Therefore, while we can't say for certain what the mean of the simulated random variable is, what we do know, given a sufficient number of trials, is the distribution of the sample mean. This lets us construct a confidence interval for the true mean of the simulated random variable.

Define $\Phi(z)$ as the CDF of the standard normal distribution. By the previous equation, there is a 97.5% chance that

$$\frac{\mu_X - \bar{x}}{s_X/\sqrt{n}} < \Phi^{-1}(.975)$$

Therefore, there is a 97.5% chance that

$$\mu_X < \bar{x} + \frac{s_X * \Phi^{-1}(.975)}{\sqrt{n}}$$

And a 95% chance that

$$\bar{x} + \frac{s_X * \Phi^{-1}(.025)}{\sqrt{n}} < \mu_X < \bar{x} + \frac{s_X * \Phi^{-1}(.975)}{\sqrt{n}}$$

but since the standard normal distribution is symmetrical about 0,

$$\Phi(.025) = -\Phi(.975)$$

so

$$\bar{x} - \frac{s_X * \Phi^{-1}(.975)}{\sqrt{n}} < \mu_X < \bar{x} + \frac{s_X * \Phi^{-1}(.975)}{\sqrt{n}}$$

should be true 95% of the time. This lets us construct our 95% confidence interval about the mean. Note that the qnorm function lets us compute $\Phi^{-1}; \Phi^{-1}(.975) = 1.96$. Figure 4.20 shows the code for the lower and upper bounds of the confidence interval about the mean.

Figure 4.20

```
mean(smodel$NPV)-qnorm(.975)*sd(smodel$NPV)/n^.5
mean(smodel$NPV)+qnorm(.975)*sd(smodel$NPV)/n^.5
```

We can also use these equations to determine how many trials we'd need to run to get a smaller confidence interval. As we saw before, half the width of our 95% confidence interval was:

$$W = \frac{\sigma_X * \Phi^{-1}(.975)}{\sqrt{n}}$$

Solving for *n*, we find

$$n = \left(\frac{\sigma_X * \Phi^{-1}(.975)}{W}\right)^2$$

We therefore compute the number of trials we'd need for a half width of $3 million using the formula in Figure 4.21.

Figure 4.21

```
(qnorm(.975)*sd(smodel$NPV)/3)^2
```

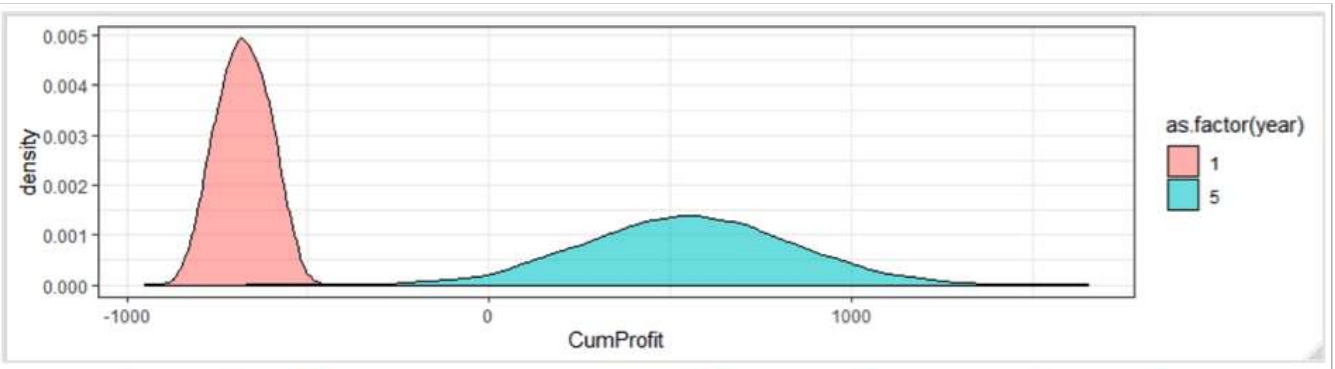
Visualizations

Overlay Charts

Our ymodel data frame makes it very easy for us to compare the distribution of cumulative net profit for different years in one chart. These results are reproduced in Figure 4.22. Here we want to only show years 1 and 5, so we filter ymodel accordingly.

We must specify how our data is grouped in the year and fill arguments to distinguish the things we're comparing. The variance of the cumulative net profit in year 5 is larger than the variance of the cumulative net profit in year 1. This makes sense because there is more uncertainty in making long-term predictions.

Figure 4.22




```
ggplot(ymodel%>%filter(year==1 | year==5))+ #overlay chart of cumulative prof
  geom_density(aes(x=CumProfit, group=year,fill=as.factor(year)),alpha=0.5)+
  theme_bw()
```

Trend Charts

We can also use ymodel to create a trend chart that compares how our data is changing from year to year.

We first summarize our model, determining the mean 75%, and 90%, confidence intervals for each year in a new data frame (summodel). We name these mean, l75, u75, l90, and u90.

For each year, our cumulative net profit has a 75% chance of falling between l75 and u75, and a 90% chance of falling between l90 and u90.

Figure 4.23

```
summodel<-ymodel %>% group_by(year) %>% #create a new dataframe summa
  summarise(mean=mean(CumProfit),       #percentiles, allowing the tr
            l90=quantile(CumProfit,.05),l75=quantile(CumProfit,.125),
            u75=quantile(CumProfit,.875),u90=quantile(CumProfit,.95))
```

```
> summodel
# A tibble: 5 x 6
   year  mean    l90   l75    u75    u90
  <int> <dbl>  <dbl> <dbl>  <dbl>  <dbl>
1     1 -679. -800.  -767. -590.  -558.
2     2 -463. -630.  -582. -346.  -296.
3     3 -196. -434.  -365.  -25.6   45.9
4     4  137. -200.  -105.  378.   478.
5     5  551.   87.2  216.  882.  1022.
```

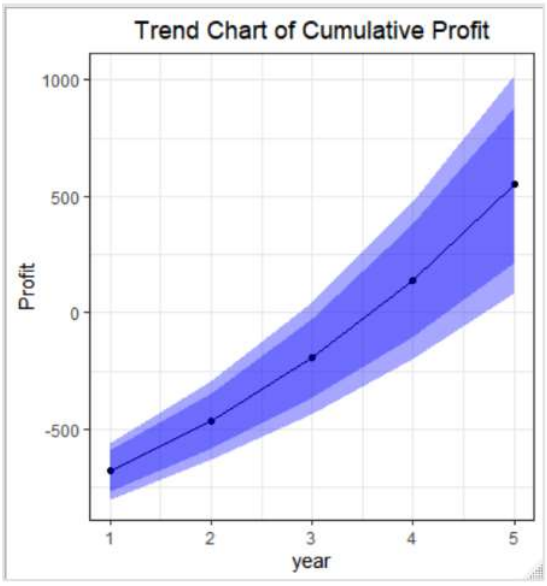
We then plot it in ggplot.

We use a geom_line and geom_point to represent the means, and geom_ribbon to represent the two confidence intervals.

The alpha argument to geom_ribbon controls how transparent the ribbons are. Mean net cumulative profit increases over time.

The 75 percent band and the 90 percent band are getting wider as we move to the right on the horizontal axis. This shows that variation around the net cumulative profit increases over time.

Figure 4.24




```
ggplot(summodel)+geom_line(aes(x=year,y=mean))+ #trend chart of cumu
  geom_point(aes(x=year,y=mean))+
  geom_ribbon(aes(x=year,ymin=l90,ymax=u90),alpha=0.3,fill="blue")+
  geom_ribbon(aes(x=year,ymin=l75,ymax=u75),alpha=0.3,fill="blue")+
  theme_bw()+
  ylab("Profit")+
  ggtitle("Trend Chart of Cumulative Profit")+
  theme(plot.title = element_text(hjust = 0.5))
```

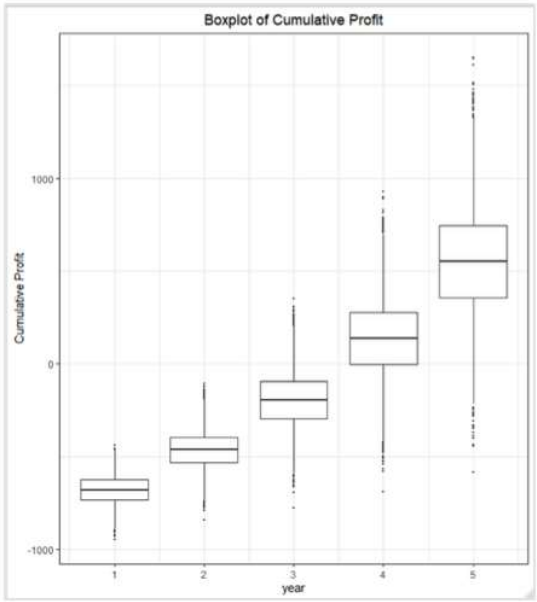
Boxplots

Boxplot charts help to visualize the statistical properties of the output variable.

In R, geom_boxplot will display the 1st and 3rd quartiles, represented by the box; the median, represented as a horizontal line through the box; and vertical lines which are at most 1.5 times the length of the interquartile range, extending above and below the box. Any data that fall past the vertical lines are considered outliers and given their own points.

Again, we group our data by year.

Figure 4.25




```
ggplot(ymodel)+geom_boxplot(aes(y=CumProfit,group=year,x=year),outlier.size=0)+
  theme_bw()+
  ggtitle("Boxplot of Cumulative Profit")+
  ylab("Cumulative Profit")+
  theme(plot.title = element_text(hjust = 0.5))
```

Sensitivity Analysis

One form of sensitivity analysis is to measure the correlation between the inputs and outputs of the model. We will produce a tornado chart to measure the effect of each uncertain input in our model on the NPV. Once it's determined which elements have the highest correlation (positive or negative), a wise organization might prioritize those inputs that have the highest effect on its success.

In order to perform this operation, we need to pull our uncertain inputs from the nested data frames to measure their impact on NPV. We do this for market size growth and market share growth.

We first write a function that pulls the column from nmodel, then lapplies it to smodel. The result is a list of 10,000 lists.

We use unlist to turn it into a vector of 50,000 elements.

Next, we turn it into a matrix with 5 rows and 10,000 columns. We then transpose this into a matrix of 10,000 rows and 5 columns.

Finally, we turn it into a data frame, remove the first column (of all ones), and name the columns after the year they represent.

These steps are precisely the same for both market size and market growth.

Individual Exercise:

Run each step one at a time to check if the eventual data frame is transformed.

The code to run these operations and resulting data frame are reproduced in Figure 4.26.

Figure 4.26

```
> head(MSzGModel,10)
      MSzG2    MSzG3    MSzG4    MSzG5
1  1.034102 1.027446 1.028500 1.033171
2  1.023939 1.051856 1.038116 1.049556
3  1.031980 1.043531 1.025314 1.034813
4  1.034357 1.041968 1.038045 1.041901
5  1.028401 1.046481 1.043231 1.050994
6  1.039259 1.030504 1.023778 1.027376
7  1.037223 1.043815 1.047427 1.040237
8  1.028974 1.045681 1.030072 1.029248
9  1.022376 1.030368 1.028345 1.031318
10 1.043658 1.045690 1.038132 1.037064
```

```
MSzGModel<-lapply(smodel$nmodel,function(model){
  model$MSzG}) %>%
  unlist() %>%
  matrix(ny,n) %>%
  t()%>%
  as.data.frame() %>%
  select(-1)
colnames(MSzGModel)<-paste("MSzG",2:ny,sep='') #
```

Figure 4.26

We are now ready to create a data frame containing all the uncertain inputs we started with and the output we want to measure (NPV). We select NPV RnD, CT, and MSz from smodel, then cbind it to our data frames for market share and market size, creating “CorrelationModel”.

We then create a correlation matrix for CorrelationModel using the cor function.

Since we’re only interested in correlations with NPV, we select the first row of the matrix.

The first element is the correlation of NPV and itself, guaranteed to be 1. We remove it from the vector.

The code for this and the result can be seen in Figure 4.27.

Figure 4.27

```
> NPVCorrs
        RnD          CT         MSz        MSzG2        MSzG3
-0.26757091 -0.14067619  0.94839325  0.03216569  0.03253089
      MSzG4        MSzG5       MShG2       MShG3       MShG4
 0.01826942 -0.00682923  0.06606250  0.04679181  0.04761816
      MShG5
 0.01859785
```

```
CorrelationModel<-smodel %>% #bind together NP
  select(6,2:4) %>% cbind(MSzGModel,MShGModel)

NPVCorrs<-cor(CorrelationModel)[1,][-1] #take
```

All that remains to do is plot this in a bar chart. We have to change what the bar chart is measuring, using the stat argument, or else it will count elements rather than just displaying the already summarized value.

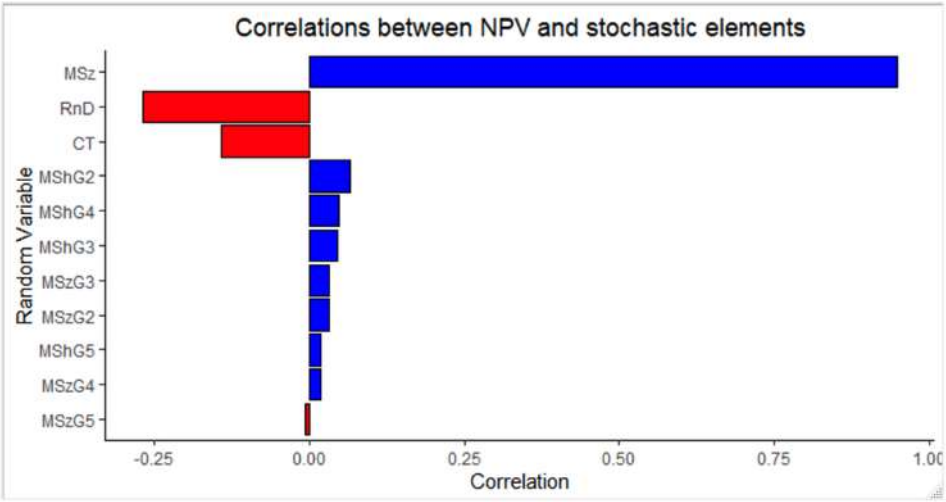
We use the reorder function to put the highest magnitude values at the top, and set the fill to check whether the correlation is positive or negative.

The scale_fill_manual function will then set the color and remove the legend.

coord_flip rotates the graph by 90 degrees.

The result can be seen in figure 4.28.

Figure 4.28




```
ggplot() +
  geom_bar(aes(x=reorder(names(NPVCorrs),abs(NPVCorrs)), #plot
               y=NPVCorrs,fill=(NPVCorrs>0)), #stat='identity'
           stat='identity', color='black')+   #rather than tryi
  coord_flip()+xlab('Random Variable')+   #coord_flip turns the
  ylab('Correlation')+                    #it also flips the X
  scale_fill_manual(values=c('red','blue'),guide='none')+ #we s
  theme_classic()+                                        #this
  ggtitle("Correlations between NPV and stochastic elements")+
  theme(plot.title = element_text(hjust = 0.5))
```

The initial market size has a strong positive correlation of 0.95 with NPV. It has by far the largest impact on NPV.

The R&D cost has a negative correlation of 0.255 with NPV and the clinical trial cost has a negative correlation of 0.130 with NPV.

As market size increases, NPV increases, whereas as R&D cost and clinical trial cost increases, NPV decreases.

All other uncertain input variables have little effect NPV.

If you want to estimate NPV better, you should focus your efforts on obtaining better information about the market size. You should pay close attention to what distribution you use to model the market size as this distribution has a huge impact on the resulting NPV.

The NPV and the market growth factors have negligible correlation. This suggests that we can use constant values instead of probability distributions to model the market growth factors. Using constant values would have little impact on the results.

Lecture 4 Footnotes

[1] Evans, J. R. (2016). *Business analytics: Methods, models, and decisions* (2nd edition). Pearson Education, Inc., Chapter 11, Example 11.8, pp. 351–352.

[2] Evans, J. R. (2016). *Business analytics: Methods, models, and decisions* (2nd edition). Pearson Education, Inc., Chapter 11, Example 11.8, p. 351.

[3] Adapted from Evans, J. R. (2016). *Business analytics: Methods, models, and decisions* (2nd edition). Pearson Education, Inc., Chapter 11, Example 11.8, p. 351.

[4] Evans, J. R. (2016). *Business analytics: Methods, models, and decisions* (2nd edition). Pearson Education, Inc., p. 80.

Lecture 4 References

Albright, S. C., & Winston, W. L. (2015). *Business analytics: Data analysis and decision making* (5th Edition). Cengage Learning.

Evans, J. R. (2016). *Business analytics: Methods, models, and decisions* (2nd edition). Pearson Education, Inc.