Module 4: July 31st, 2015

Follow-up from Module 3:

Bootstrap vs. other measures of standard error

-Bootstrap allows assigning measures of accuracy to sample estimates and is used for non-standard statistics.

-One case where you wouldn't need to use it would be estimating the standard error of the sample mean.

Monte Carlo

-Used to simulate random variable with computational algorithms

-Bootstrapping is a subset.

Permutation Tests

Are two quantities associated?

To determine if the association is significant:

-Disassociate the two variables, and see if they are associated

-Sees how a statistic that measures association changes if repeated and dissociated

Example using Titanic dataset

```
library(mosaic)

TitanicSurvival = read.csv('../data/TitanicSurvival.csv')

# A 2x2 contingency table
t1 = xtabs(~sex + survived, data=TitanicSurvival)
t1

##         survived
## sex       no yes
##   female 127 339
##   male   682 161

p1 = prop.table(t1, margin=1)
p1

##         survived
## sex             no       yes
##   female 0.2725322 0.7274678
##   male   0.8090154 0.1909846
```

```
# Calculate the relative risk of dying for both men and women
# in terms of the individual cells of the table
risk_female = p1[1,1]
risk_male = p1[2,1]
relative_risk = risk_male/risk_female
relative_risk

## [1] 2.968513

# Or with mosaic's function
relrisk(t1)

## [1] 2.968513
```

Is there an association between gender and survival rates on the Titanic?

-Based on the above contingency table, we can calculate the relative risk for males.

-The relative risk for males is 3. If there was no association between gender and survival rates, the relative risk would be 1.

-To test this association, we use a permutation test.

First, we need to shuffle one of the two variables and hold the other one constant.

-In the Titanic example, we are shuffling the gender variable and keeping the survival variable unchanged.

```
titanic_shuffle = data.frame(shuffle(TitanicSurvival$sex),
TitanicSurvival$survived)
head(TitanicSurvival)

##                                   X survived    sex     age passengerClass
## 1   Allen, Miss. Elisabeth Walton      yes female 29.0000            1st
## 2  Allison, Master. Hudson Trevor      yes   male  0.9167            1st
## 3    Allison, Miss. Helen Loraine       no female  2.0000            1st
## 4 Allison, Mr. Hudson Joshua Crei       no   male 30.0000            1st
## 5 Allison, Mrs. Hudson J C (Bessi       no female 25.0000            1st
## 6             Anderson, Mr. Harry      yes   male 48.0000            1st

head(titanic_shuffle)

##   shuffle.TitanicSurvival.sex. TitanicSurvival.survived
## 1                         male                      yes
## 2                         male                      yes
## 3                         male                       no
## 4                         male                       no
## 5                       female                       no
## 6                         male                      yes
```

-We create a new contingency table using the shuffled gender variable and the unchanged survival variable and then calculate the new relative risk based on the contingency table.

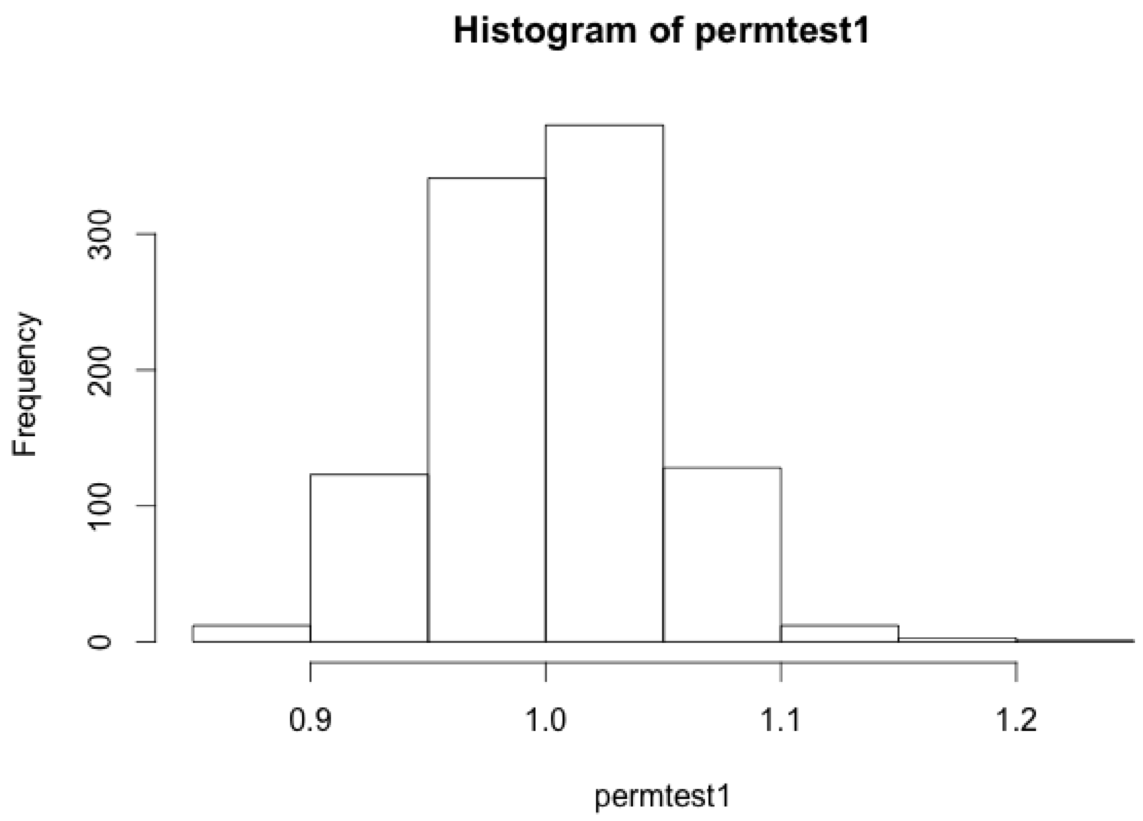
-The resulting relative risk is ~1, meaning that we have dissociated the two variables.

-Note: in this example, the result of the permutation centers around 1, and this is due to the relative risk test statistic. If another test statistic is used, the result may center around a number other than 1.

```
t1_shuffle = xtabs(~shuffle(sex) + survived, data=TitanicSurvival)
relrisk(t1_shuffle)

## [1] 0.9787371
```

-Using a monte carlo simulation, we can repeat this permuation many times to understand the sampling distribution.

```
permtest1 = forreach(i = 1:1000, .combine='c') %do% {

  t1_shuffle = xtabs(~shuffle(sex) + survived, data=TitanicSurvival)

  relrisk(t1_shuffle)

}
```



-This histogram allows us to see the plausible range of values for the test statistic under the null hypothesis.

-In this example, the relative risk numbers range from 0.85 to 1.20. Therefore, it is extremely unlikely that a relative risk of ~3 would occur purely due to chance. We can thus, reject the null hypothesis that there is no association between gender and survival rate on the Titanic.

Characterizing variability in returns for financial portfolios

What happens with a riskless asset over a 40 year trading period?

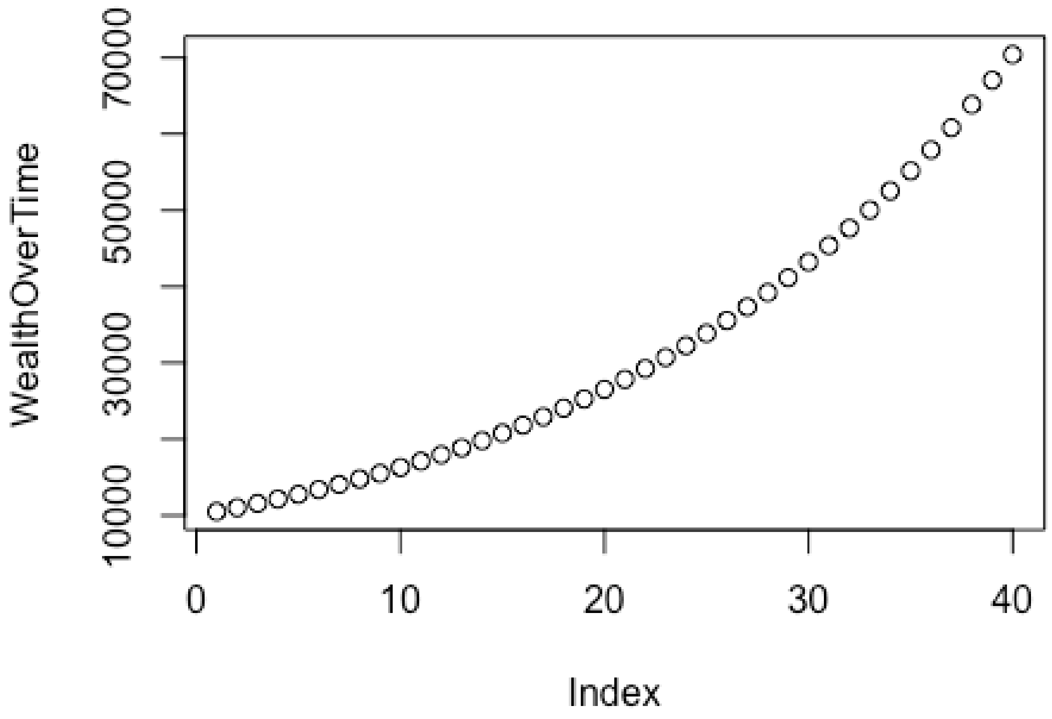
```
library(foreach)  # I typically use this for MC simulations

# First, a riskless asset
Horizon = 40
ReturnAvg = 0.05

Wealth = 10000
# Sweep through each year and update the value of wealth
for(year in 1:Horizon) {
    ThisReturn = ReturnAvg
    Wealth = Wealth * (1 + ThisReturn)
}
Wealth

## [1] 70399.89

# The usual compound-interest formula
10000 * (1+ReturnAvg)^40

## [1] 70399.89
```

-Given a starting investment of $10,000 with an average annual return of 5%, the resulting portfolio value after 40 years is $70,399.89.

-This value does not change when the projection is repeated because there is no uncertainty.

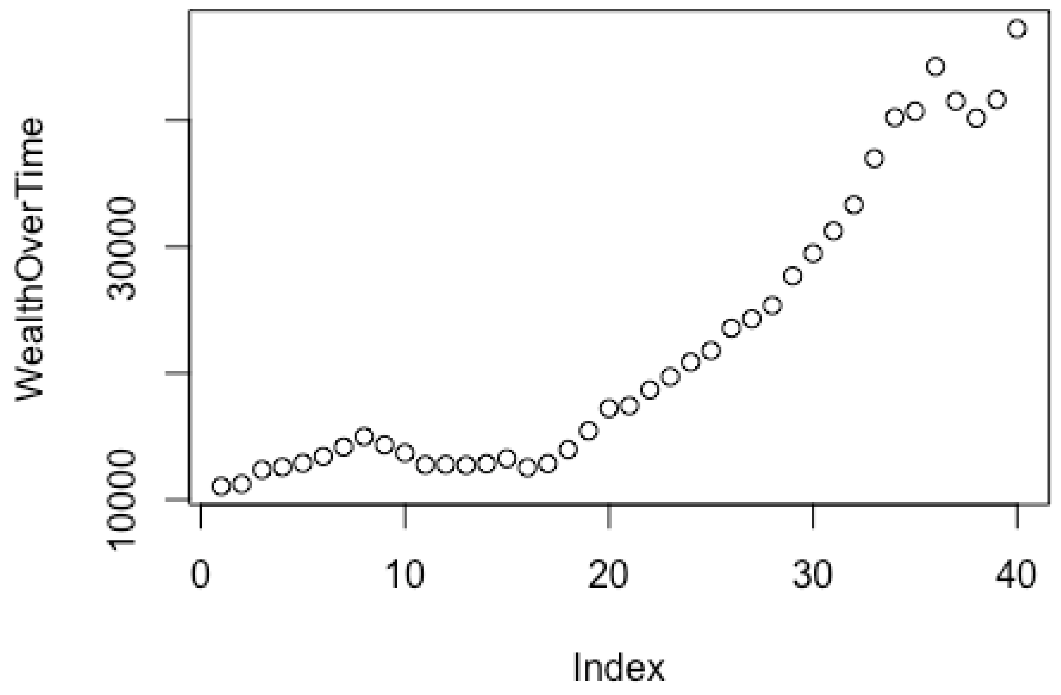
-See below for the change in wealth over time, which is an exponential growth curve with no uncertainty.

```
Wealth = 10000  # Reset initial wealth
WealthOverTime = rep(0, Horizon)  # Allocate some space
# Sweep through each year and update the value of wealth
for(year in 1:Horizon) {
  ThisReturn = ReturnAvg
  Wealth = Wealth * (1 + ThisReturn)
  WealthOverTime[year] = Wealth
}
Wealth

## [1] 70399.89
```



What if we inject some noise/variability?

```
# Now a risky asset with a positive expected return
ReturnAvg = 0.05
ReturnSD = 0.05
Horizon = 40

Wealth = 10000  # Reset initial wealth
WealthOverTime = rep(0, Horizon)  # Allocate some space
# Sweep through each year and update the value of wealth
for(year in 1:Horizon) {
  ThisReturn = rnorm(1, ReturnAvg, ReturnSD)
  Wealth = Wealth * (1 + ThisReturn)
  WealthOverTime[year] = Wealth
}
Wealth

## [1] 47221.4
```

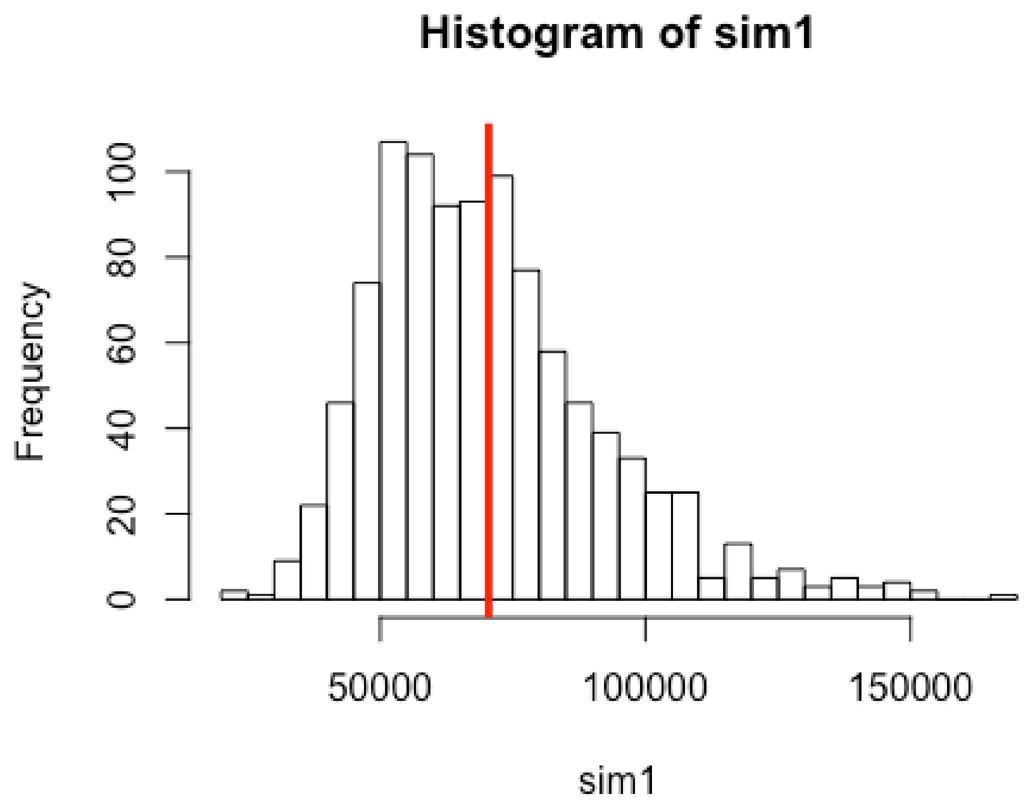


-After adding in a standard deviation of 5% to the average annual return rate, we see a different terminal wealth value with high variability each time you run this simulation.

-To quantify this variabilty, we can use a monte carlo simulation.

```
ReturnAvg = 0.05
ReturnSD = 0.05
Horizon = 40
sim1 = foreach(i=1:1000, .combine='c') %do% {
    Wealth = 10000  # Reset initial wealth
    # Sweep through each year and update the value of wealth
    for(year in 1:Horizon) {
        # Generate a random return
        ThisReturn = rnorm(1, ReturnAvg, ReturnSD)

        # Update wealth
        Wealth = Wealth * (1 + ThisReturn)
    }
    # Output the value of wealth for each simulated scenario
    Wealth
}
```

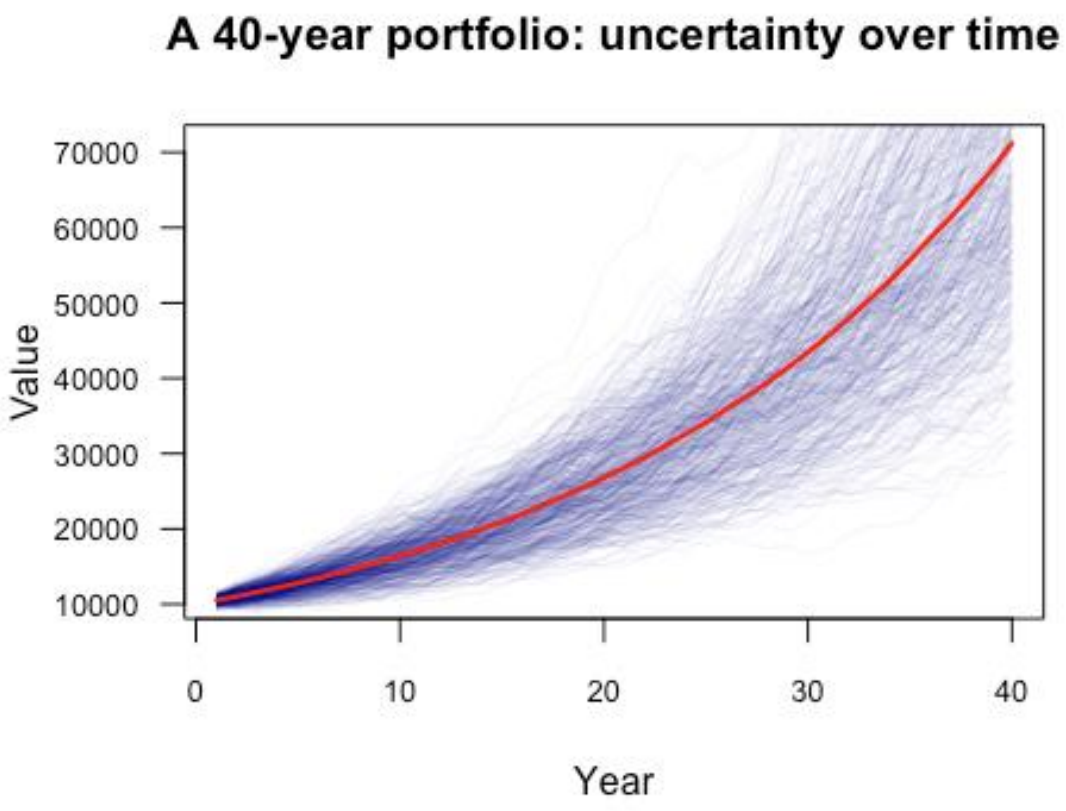
-Using rnorm, we obtain a yearly return that is normally distributed. -The monte carlo simulation repeats 1,000 times and returns the sampling distribution of the terminal wealth over 40 years.



```
mean(sim1)

## [1] 70603.06

sd(sim1)

## [1] 21737.53
```

-The expected value of the monte carlo simulation terminal wealth mean is similar to the terminal wealth of a riskless asset. However, by adding variability into the average return rate, we see a large spread of possibilities for the terminal wealth.

```
ReturnAvg = 0.05
ReturnSD = 0.05
Horizon = 40
sim1 = foreach(i=1:500, .combine='rbind') %do% {
  Wealth = 10000  # Reset initial wealth
  WealthOverTime = rep(0, Horizon)  # Allocate some space
  # Sweep through each year and update the value of wealth
  for(year in 1:Horizon) {
    ThisReturn = rnorm(1, ReturnAvg, ReturnSD)
    Wealth = Wealth * (1 + ThisReturn)
    WealthOverTime[year] = Wealth
  }
  WealthOverTime
}
```



-In summary, we can use monte carlo to simulate the variability in the sampling distribution of the terminal wealth of an asset with risk.

-However, having only one asset in a portfolio is unrealistic. When you add another asset, we need to think about the joint distribution of the two assets (describes how the two assets covary). As more assets are added, this gets much more complicated.

On Monday, we will look at how bootstrapping can be a useful tool for short term forecasts to get around this problem.

Appendix: Helper Functions

A helper function can be used to add capabilities that aren't otherwise available or to replicate functionality without loading unnecessary components.

```
library(mosaic)
library(fImport)

## Loading required package: timeDate
## Loading required package: timeSeries

library(foreach)

# Import a few stocks
mystocks = c("MRK", "JNJ", "SPY")
myprices = yahooSeries(mystocks, from='2011-01-01', to='2015-07-30')
# The first few rows
head(myprices)

## GMT
##            MRK.Open MRK.High MRK.Low MRK.Close MRK.Volume MRK.Adj.Close
## 2011-01-03    36.29    36.78   35.99     36.04   19559500      30.41457
## 2011-01-04    36.24    36.40   35.85     36.35   13926100      30.67618
## 2011-01-05    36.04    36.58   36.00     36.56   14520300      30.85340
## 2011-01-06    36.56    37.14   36.56     37.06   11990300      31.27536
## 2011-01-07    37.17    37.35   36.86     37.35   12753500      31.52009
## 2011-01-10    37.26    37.62   37.16     37.20   10723700      31.39350
##            JNJ.Open JNJ.High JNJ.Low JNJ.Close JNJ.Volume JNJ.Adj.Close
## 2011-01-03    62.63    63.18   62.53     62.82   14894800      54.41755
## 2011-01-04    63.13    63.35   62.75     63.35   12346300      54.87665
## 2011-01-05    63.41    63.54   62.95     63.31   11837900      54.84201
## 2011-01-06    63.43    63.53   62.90     63.21    7606000      54.75538
## 2011-01-07    63.20    63.25   62.56     62.60   11084800      54.22697
## 2011-01-10    62.29    62.40   62.00     62.16    9775000      53.84582
##            SPY.Open SPY.High SPY.Low SPY.Close SPY.Volume SPY.Adj.Close
## 2011-01-03   126.71   127.60  125.70    127.05  138725200      115.9587
## 2011-01-04   127.33   127.37  126.19    126.98  137409700      115.8948
## 2011-01-05   126.58   127.72  126.46    127.64  133975300      116.4972
## 2011-01-06   127.69   127.83  127.01    127.39  122519000      116.2690
## 2011-01-07   127.56   127.77  126.15    127.14  156034600      116.0409
## 2011-01-10   126.58   127.16  126.20    126.98  122401700      115.8948

# A helper function for calculating percent returns from a Yahoo Series
# Source this to the console first, and then it will be available to use
# (Like importing a library)
YahooPricesToReturns = function(series) {
    mycols = grep('Adj.Close', colnames(series))
    closingprice = series[,mycols]
    N = nrow(closingprice)
    percentreturn = as.data.frame(closingprice[2:N,]) /
as.data.frame(closingprice[1:(N-1),]) - 1
    mynames = strsplit(colnames(percentreturn), '.', fixed=TRUE)
    mynames = lapply(mynames, function(x) return(paste0(x[1], ".PctReturn")))
    colnames(percentreturn) = mynames
    as.matrix(na.omit(percentreturn))
}
```