Module 5:August 1st,2016

Review of Module 4

Resampling:

Resampling refers to the estimation of the precision of the sample statistics (medians, variances) by using subsets of available data (jackknifing), drawing randomly with replacement from a set of data points (***bootstrapping***) or exchanging labels on data points when performing significance tests (***permutation tests***)

Key discussion points in Module 5:

1. Permutation tests
2. Quantifying uncertainty of a financial portfolio through Monte Carlo Simulation

Permutation tests

There are two major questions that come to mind when we deal with uncertainty in statistical modelling:

Q1 - How sure can I be when I make an estimate on a parameter?

How close is the sample statistic to its actual value from the overall population? Using a bootstrap, we can estimate this to our guess with some positive or negative error range(guess +- error)

Q2 - How sure can I be that the association between variables is real and not just merely due to chance?

This is where we discuss the notion of permutation tests. It is relatively assumption free and draws motivation from the bootstrap technique. Hypothesis tests, p values, t tests and chi square tests can also help in answering this question.

Let's understand permutation tests with the example of Titanic survival problem. (R code - Titanic.R)

First we read the titanic file into R.

```
library(mosaic)
TitanicSurvival = read.csv('C:/MSBA/James Scott Statistics/STA380-master/STA380-master/data/TitanicSurvi
```

Then, we create a contingency table of raw counts and proportions (summed to 1 across rows) between sex and survival status.

```
# 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
```

Let's take a look at the relative risk of dying between males and females. The relative risk for males is the ratio of probability of dying|male (interpreted as ratio of dying when male) to the probability of dying|female (interpreted as ratio of dying when female).

```
risk_female = p1[1,1]
risk_male = p1[2,1]
relative_risk = risk_male/risk_female
relative_risk
```

```
## [1] 2.968513
```

From this table, we can see that the relative risk is ~ 3. If there was no association between the variables, we would have expected a relative risk of ~ 1 . Is the observed relative risk actually true or merely due to chance? Permutation tests can help solve this problem.

Permutation tests help in finding out where the threshold of believeability lies (i.e) when the value obtained is no longer consistent with intuition about chance. It works by breaking all associations between variables and testing to see if any systematic correlation exists thereafter. An analogy can be drawn with shuffling a deck of cards.

To demonstrate this, we take two different vesions of the alphabet - sesame_street and the army. There is a clear association between the sesame_street and army alphabets.

```
sesame_street = c('A', 'B', 'C', 'D', 'E')
army = c('alpha', 'bravo', 'charlie', 'delta', 'echo')
data.frame(sesame_street, army)
```

```
##   sesame_street    army
## 1             A   alpha
## 2             B   bravo
## 3             C charlie
## 4             D   delta
## 5             E    echo
```

Now when we reshuffle one of the sets, we see that all asociations are broken.

```
data.frame(shuffle(sesame_street), army)
```

```
##   shuffle.sesame_street.    army
## 1                      B   alpha
## 2                      E   bravo
## 3                      A charlie
## 4                      C   delta
## 5                      D    echo
```

The same concept can be applied to the Titanic dataset. We perform a similar shuffle on the sex variable to see if there is any systematic risk associated with males and death. Generally, the predictor variable is shuffled as opposed to the response. eg. in the case of multiple regression, shuffling the response can break the association between itself and all other variables (not only the variables under consideration).

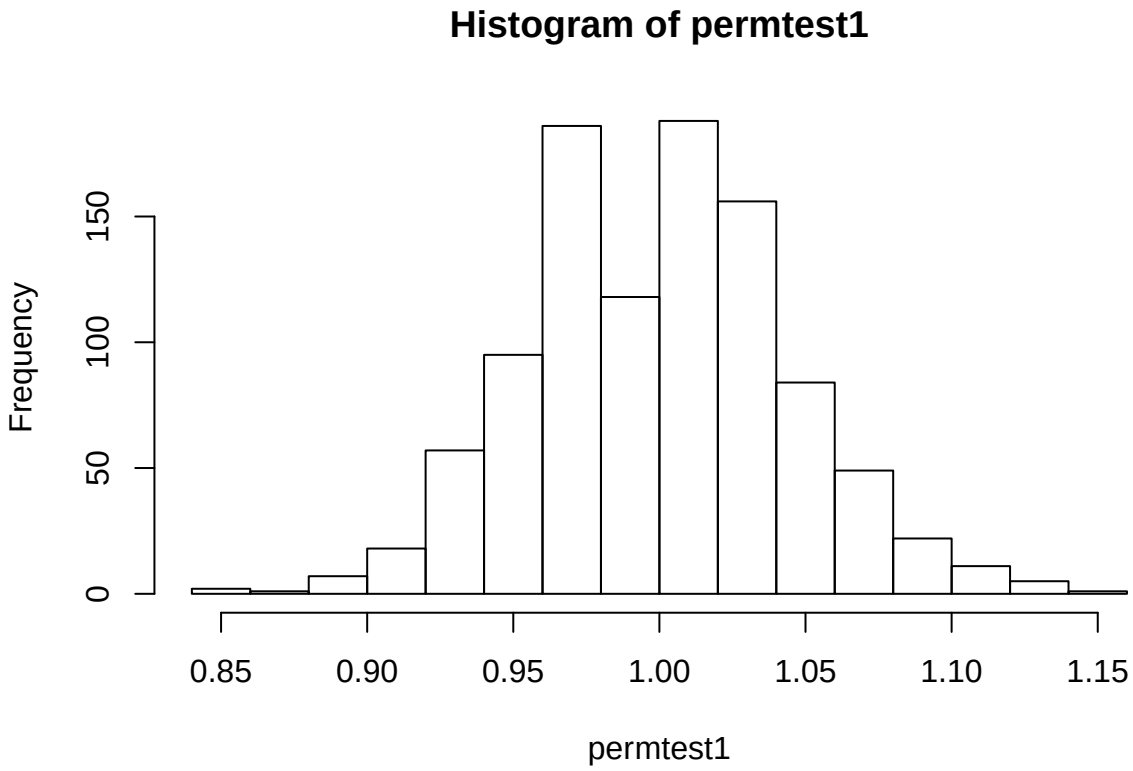
```
titanic_shuffle = data.frame(shuffle(TitanicSurvival$sex), TitanicSurvival$survived)
t1_shuffle = xtabs(~shuffle(sex) + survived, data=TitanicSurvival)
relrisk(t1_shuffle)
```

```
## [1] 1.021879
```

We find that the relative risk is ~1. The same experiment is repeated multiple times.

```
library(foreach)
permtest1 = foreach(i = 1:1000, .combine='c') %do% {
  t1_shuffle = xtabs(~shuffle(sex) + survived, data=TitanicSurvival)
  relrisk(t1_shuffle)
}

# Compare with the observed relative risk
hist(permtest1)
```



A histogram under the null hypothesis of no association between variables is obtained. From this, we can conclude that there is no systematic risk as the relative risk when shuffled is around 1.

Since, the initial relative risk obtained was ~3, the null hypothesis is not plausible (i.e) the value is staistically significant. This concept can be extended to any test statistic that measures association between variables such as log odds ratios, least squares regression coefficients, f statistics etc.

Quantifying uncertainity using Monte Carlo simulation

How do we quantify uncertainty in portfolio allocation?

While making decisions about financial investment, there are multiple sources of randomness such as allocation of budget between stocks/bonds, future value of these assets, which stock/bond to buy among all the different kinds etc. Monte Carlo Simulation helps in giving structure to this process through computer simulations that rely on random sampling.

Monte Carlo simulation is used to simulate the performance of stocks and bonds and calculate the returns

from the portfolio at different points in time.

R code from the file portfolio.R:

fImport directly imports finance data from Yahoo fincance into R, skipping csv downloads. The yahooSeries function downloads daily pricing data for the stocks Merck, Johnson and Johnson and S&P 500.

```
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      29.39734
## 2011-01-04    36.24    36.40   35.85     36.35   13926100      29.65020
## 2011-01-05    36.04    36.58   36.00     36.56   14520300      29.82149
## 2011-01-06    36.56    37.14   36.56     37.06   11990300      30.22934
## 2011-01-07    37.17    37.35   36.86     37.35   12753500      30.46588
## 2011-01-10    37.26    37.62   37.16     37.20   10723700      30.34353
##            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      52.84620
## 2011-01-04    63.13    63.35   62.75     63.35   12346300      53.29206
## 2011-01-05    63.41    63.54   62.95     63.31   11837900      53.25841
## 2011-01-06    63.43    63.53   62.90     63.21    7606000      53.17428
## 2011-01-07    63.20    63.25   62.56     62.60   11084800      52.66113
## 2011-01-10    62.29    62.40   62.00     62.16    9775000      52.29099
##            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      113.4980
## 2011-01-04   127.33   127.37  126.19    126.98  137409700      113.4355
## 2011-01-05   126.58   127.72  126.46    127.64  133975300      114.0251
## 2011-01-06   127.69   127.83  127.01    127.39  122519000      113.8017
## 2011-01-07   127.56   127.77  126.15    127.14  156034600      113.5784
## 2011-01-10   126.58   127.16  126.20    126.98  122401700      113.4355
```

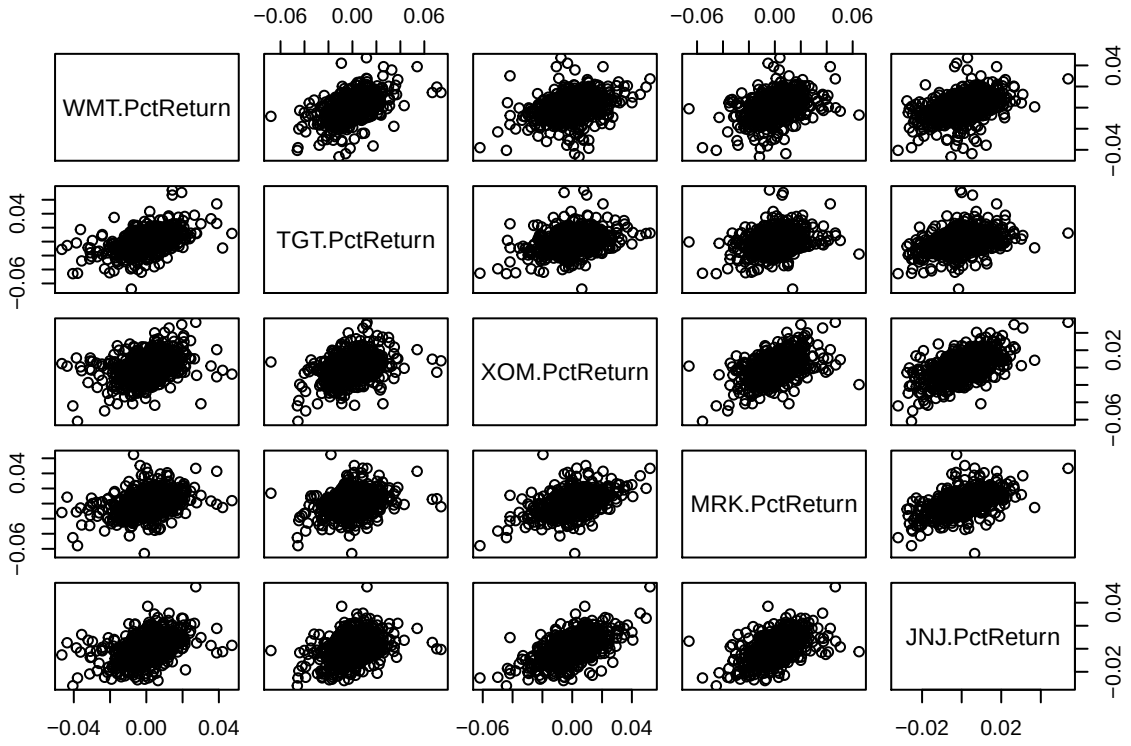
We create a helper function which returns the percentage returns when a series of prices is passed as input:

```
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))
}
```

Let's take 5 stocks with equal weight in our portfolio and check their pairwise correlations:

```
mystocks = c("WMT", "TGT", "XOM", "MRK", "JNJ")
myprices = yahooSeries(mystocks, from='2011-01-01', to='2015-07-30')

# Compute the returns from the closing prices
myreturns = YahooPricesToReturns(myprices)
pairs(myreturns)
```



When the correlations between stocks are positive, our portfolio will witness more up and downswings as opposed to when the correlations are negative as the swings balance each other out.

We can simulate the performance of stocks by resampling from past market returns. First, we simulate a single day. The budget is split between our stocks, a joint set of returns is drawn for each of them and holdings are updated based on the returns.

```
# Sample a random return from the empirical joint distribution
# This simulates a random day
return.today = resample(myreturns, 1, orig.ids=FALSE)

# Update the value of your holdings
total_wealth = 10000
holdings = total_wealth*c(0.2,0.2,0.2, 0.2, 0.2)
holdings = holdings + holdings*return.today

# Compute your new total wealth
totalwealth = sum(holdings)
```

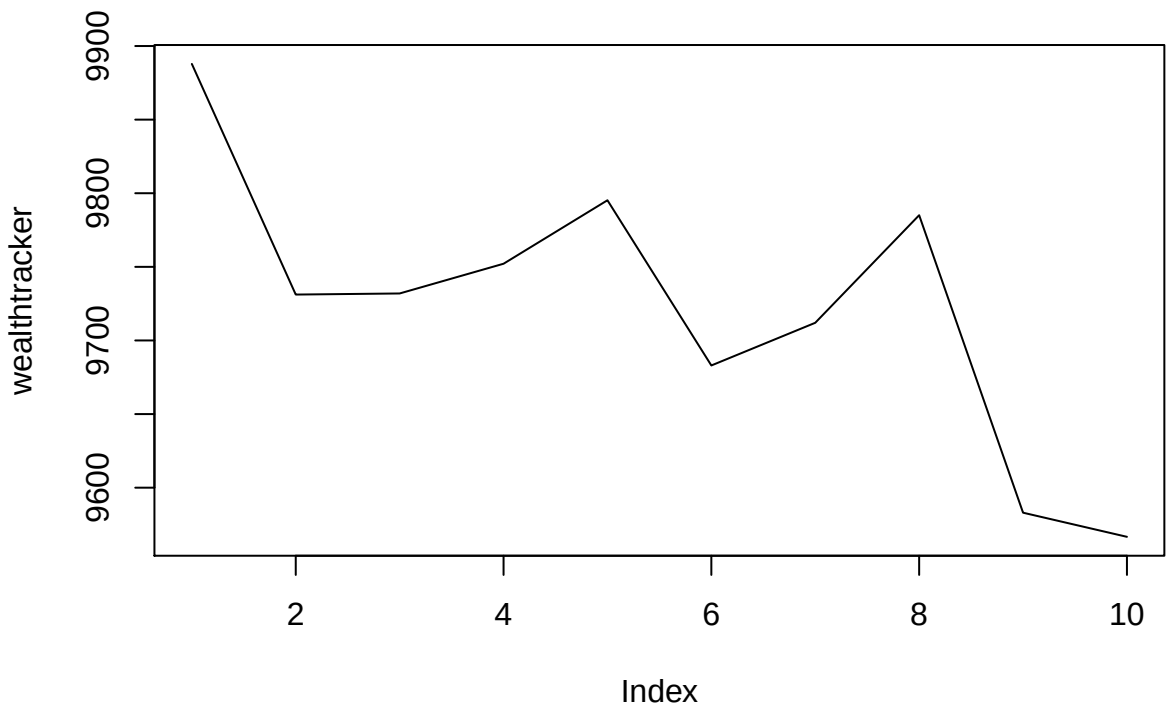
Now, let's extend this simulation to two weeks (10 working days). The total wealth for each day depends on

the performance of the stocks in the previous days.

```
# Now loop over two trading weeks
totalwealth = 10000
weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
holdings = weights * totalwealth
n_days = 10
wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
for(today in 1:n_days) {
    return.today = resample(myreturns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    totalwealth = sum(holdings)
    wealthtracker[today] = totalwealth
}
totalwealth
```

```
## [1] 9566.629
```

```
plot(wealthtracker, type='l')
```



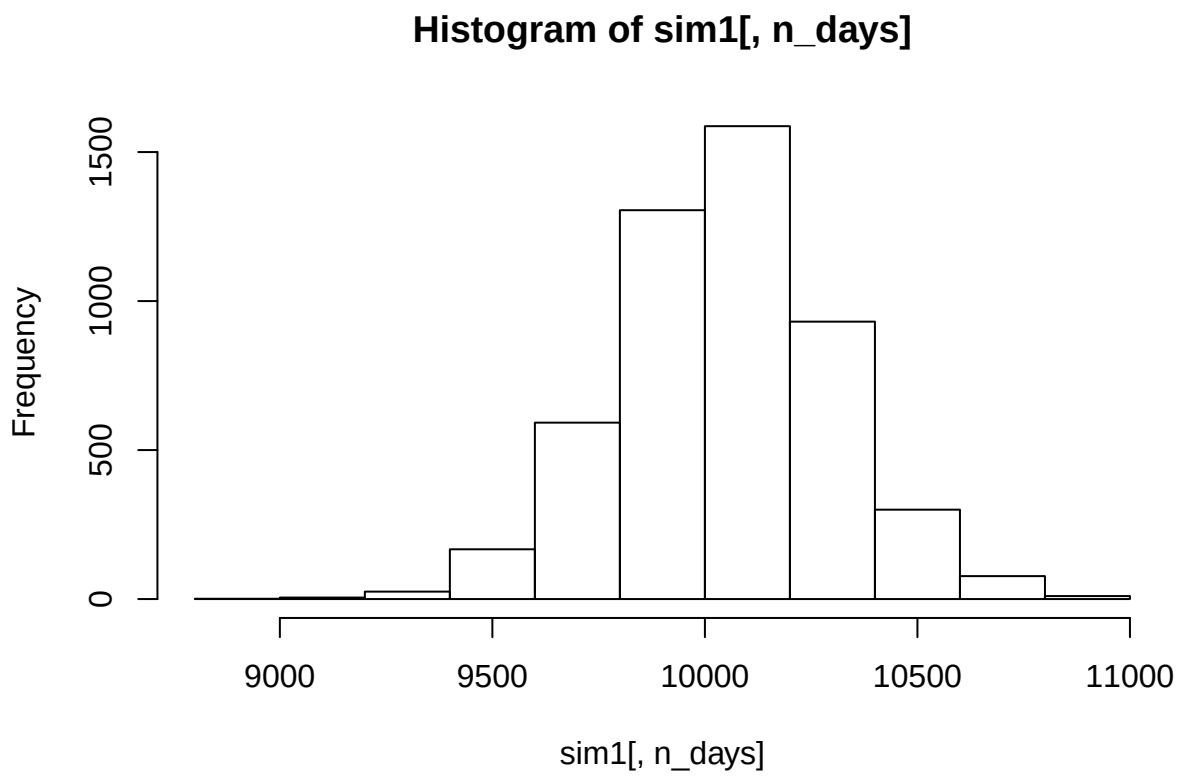
This is fed into a Monte Carlo simulation of 5000 loops while keeping track of our wealth at every step. sim1 contains 5000 rows and 10 columns corresponding to the different monte carlo simulations and the trading days repectively.

```
# Now simulate many different possible trading years!
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    totalwealth = 10000
    weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
    holdings = weights * totalwealth
```

```
    wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
    for(today in 1:n_days) {
        return.today = resample(myreturns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        totalwealth = sum(holdings)
        wealthtracker[today] = totalwealth
    }
    wealthtracker
}
```

To get the probability distribution of the value of our portfolio in 2 weeks, we create a histogram with only column 10 of sim1.

```
hist(sim1[,n_days])
```



The histogram is centered at ~ $10,000 with a huge level of certainity (width of the histogram) that outweighs expected profits. Value at risk is a specified quantile of the profit or loss distribution. For example, to calculate the 5% value at risk, we identify the 5% quantile from the histogram of profits and losses and find its corresponding value. This statistic is very important to large banks as they are required to characterize the risk of their financial portfolio by law.