# Introduction to Financial Engineering (HW4)

Organizer: Albert Ferreiro-Castilla
room: PC email: aferreiro@mat.uab.cat

- Please follow the guidelines for assignments given in the Module Handbook.
- All programs should be written in R (compilable without errors or warnings).
- You should submit a write-up (.pdf) of the program as well as the source code (.r).
- File names should be as `yoursurname_yourname_HW4.extension`
- You should submit via moddle.
- Deadline: 10th November 2023 at 10am.

1. We will program a binomial tree algorithm to compute option prices.

   (a) **Building a tree**: The required variables to build up a tree are the stock price $S$, the upward allowed movement $u$ and the downward allowed movement $v$. Remember that $0 < v < 1 < u$. We also need the number of time steps to be used $N$. Write a function with the following prototype that outputs a binomial tree:

```
build_stock_tree = function(S, u, v, N) {
?
?
?
?
?
?
?
  return(tree)
}
```

```
build_stock_tree(S=10, u=1.1, v=0.9, N=2)
     [,1] [,2] [,3]
[1,] 10.0  0.0  0.0
[2,]  9.0 11.0  0.0
[3,]  8.1  9.9 12.1
```

   Rows represent depth of the tree, while columns represent the nodes in a particular depth.

   (b) **Risk neutral probability**:Lets now compute the risk neutral probability of an upward moment $q$. For this we need the values $u$, $v$ and the interest rates $r$ with the time step $dt$ (recall that the probability of a downward movement will be $(1-q)$). Recall from the theory, that by a no-arbitrage argument we have :

$$S(1 + rdt) = quS + (1 - q)vS$$

   Write a function with the following prototype that outputs $q$:

```
q_prob = function(r, u, v, dt) {
?
?
 return(?)
}
```

```
q_prob(0.1,1.1,0.9,1/256)
[1] 0.5019531
```

   (c) **Recursive valuation**: Lets now write a function that computes the value of an option recursively. The first task of the option is to create an empty tree which we will fill with the option value recursively. Fill the last nodes of the tree with the payoff function (the function should take a call or a put option). Now use a loop to fill in the nodes one step backward using the formula:

$$V(t) = \frac{V(t + dt)^+ q + V(t + dt)^- (1 - q)}{(1 + rdt)} \tag{1}$$

where $V(t+dt)^+$ means the value of the option at $t+dt$ in the upward node, and similarly for $V(t+dt)^-$. Write a function with the following prototype:

```
value_binomial_option = function(tree, u, v, r, dt, K, type) {
  q = q_prob(r, u, v, dt)

  option_tree = matrix(0, nrow=nrow(tree), ncol=ncol(tree))
?
?
?
?
?
?
?
?
?
?
  return(option_tree)
}
```

```
S=10
N=5
u=1.01
v=0.99
r=0.1
dt=1/256
K=10
value_binomial_option(tree, u, v, r, dt, K=10, type='call')
           [,1]       [,2]       [,3]      [,4]      [,5]      [,6]
[1,] 0.10367391 0.00000000 0.00000000 0.0000000 0.0000000 0.0000000
[2,] 0.04833025 0.15493435 0.00000000 0.0000000 0.0000000 0.0000000
[3,] 0.01372347 0.08037135 0.22400759 0.0000000 0.0000000 0.0000000
[4,] 0.00000000 0.02642543 0.13032163 0.3108179 0.0000000 0.0000000
[5,] 0.00000000 0.00000000 0.05088385 0.2038846 0.4099448 0.0000000
[6,] 0.00000000 0.00000000 0.00000000 0.0979801 0.3019797 0.5101005
```

(d) **Final price**: Put all previous functions together to build up a function that gives the price of the option. Write a function with the following prototype:

```
binomial_option = function(type, u, v, dt, r, K, S, N) {
?
?
?
  return(price)
}
```

```
binomial_option(type='call', u=1.01, v=0.99, dt=1/256, r=0.1, K=10, S=10, N=5)
[1] 0.1036739
```

2. Use the function you just build, to compute the price of a CALL option with strike ATM, with underlying currently trading at USD 100 and maturing in one year. Use a monthly time step, and an annualized interest rate of 10%. The stock is allowed to move upwards or downwards by 1% each month.