605HW1

Bin Lin

2017/2/13

1. Calculate the dot product u:v where u = [0:5; 0:5] and v = [3; -4]

```
u <- c(0.5, 0.5)
v <- c(3, -4)
t(u) %*% v
```

```
##      [,1]
## [1,] -0.5
```

2. What are the lengths of u and v? Please note that the mathematical notion of the length of a vector is not the same as a computer science definition.

```
ulength <- sqrt(t(u)%*%u)
vlength <- sqrt(t(v)%*%v)
ulength
```

```
##           [,1]
## [1,] 0.7071068
```

```
vlength
```

```
##      [,1]
## [1,]    5
```

3. What is the linear combination: 3u - 2v

```
3 * u - 2 * v
```

```
## [1] -4.5  9.5
```

4. What is the angle between u and v

```
theta <- acos( sum(v*u) / ( sqrt(sum(v * v)) * sqrt(sum(u * u))))
theta
```

```
## [1] 1.712693
```

2. Set up a system of equations with 3 variables and 3 constraints and solve for x. Please write a function in R that will take two variables (matrix A & constraint vector b) and solve using elimination. Your function should produce the right answer for the system of equations for any 3-variable, 3-equation system. You don't have

to worry about degenerate cases and can safely assume that the function will only be tested with a system of equations that has a solution. Please note that you do have to worry about zero pivots, though.

```
A <- matrix(c(1, 2, -1, 1, -1, -2, 3, 5, 4), 3, 3)
b <- c(1, 2, 6)

upper_triangular_matrix <- function(A, b)
  {
  Ab <- cbind(A,b)
  r <- dim(Ab)[1]
  c <- dim(Ab)[2]

  n <-1
  while(Ab[n,n] == 0 & n <= r)
    {
    Ab[n,] <- Ab[n+1,]
    n <- n+1
  }

  for (j in 1:(c-1))
    {
    Ab[j,] <- Ab[j,] / Ab[j,j]
    if (j == (c-1))
    {
      break
    }

    for (i in j:(r-1))
      {
        multiplier <- (Ab[i+1,j]/Ab[j,j])
        Ab[i+1,] <- Ab[i+1,] - (multiplier * Ab[j,])
    }
    }
  return (Ab)
}


back_substitution <- function(x)
  {
  y <- matrix(c(rep(0, dim(x)[1])))
  r <- dim(x)[1]
  c <- dim(x)[2]

  y[r] <- x[r,c]

  for (m in (r-1):1)
  {
    row_sum <-0
    for (n in (m+1):r)
    {
      row_sum <- row_sum + x[m, n] * y[n]
      y[m] <- (x[m, c] - row_sum) / x[m, m]
    }
  }
  return (y)
}
```

```
upper_triangular_matrix(A, b)
```

```
##                            b
## [1,] 1 1 3.0000000 1.0000000
## [2,] 0 1 0.3333333 0.0000000
## [3,] 0 0 1.0000000 0.9545455
```

```
x <- upper_triangular_matrix(A, b)
back_substitution(x)
```

```
##            [,1]
## [1,] -1.5454545
## [2,] -0.3181818
## [3,]  0.9545455
```