1st_assignment.R

d92187

2021-10-08

```
# Create Graphs
# library(igraph)
#subscriptions = graph( edges=c("A","A", "B","B", "C","C", "A","B", "A","C", "B","A", "B",
 "C"))
#subscriptions$weights = c()

# x_y shows transition from platform x to y
# For simplicity define platform C as the unsubscribed users
a_a = 0.5
a_b = 0.3
a_c = 1 - a_a - a_b
b_a = 0.2
b_b = 0.2
b_c = 1 - b_a - b_b
c_a = 0.2
c_b = 0.4
c_c = 1 - c_a - c_b

## Construct matrix A describing the flow trainsition
# Each platform is a row. Each column is a different time step
A = rbind(c(a_a, a_b, a_c), c(b_a, b_b, b_c), c(c_a, c_b, c_c))
rownames(A) = c( "A","B","C")
colnames(A) = c("A","B", "C")
A
```

```
##     A   B   C
## A 0.5 0.3 0.2
## B 0.2 0.2 0.6
## C 0.2 0.4 0.4
```

```
## Now use the Spectram Theorem from class
E = eigen(A) # Compute eigenvalues & eigenvectors
ew = E$values # Store eigenvalues of A
ew
```

```
## [1]  1.0  0.3 -0.2
```

```
D = diag(ew) # Create diagonal matrix containing eigenvalues
D
```

```
##      [,1] [,2] [,3]
## [1,]    1  0.0  0.0
## [2,]    0  0.3  0.0
## [3,]    0  0.0 -0.2
```

```
U = E$vectors # Store eigenvectors of A
U
```

```
##           [,1]       [,2]       [,3]
## [1,] 0.5773503 -0.8703883  0.2220578
## [2,] 0.5773503  0.3481553 -0.8438196
## [3,] 0.5773503  0.3481553  0.4885272
```

```
U_inv = solve(U)
orth = round(U_inv %*% U,digits=0) # Check that eigenvectors are orthogonal
orth
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

```
# Reconstruct matrix A using the eigenvalue decomposition
B = round(U %*% D %*% U_inv,2) # Reconstruct matrix
B == A ## Check if our decomposed matrix equals matrix A
```

```
##      A    B     C
## A TRUE TRUE  TRUE
## B TRUE TRUE FALSE
## C TRUE TRUE  TRUE
```

```
## Define starting point u0 (Initial market shares)
u0 = c(0.30, 0.40, 0.30)

# Define function to compute u_k (transition for kth month)
Compute_uk = function(k,u0,U,D,U_inv) {
  return(U %*% (D^k) %*% U_inv %*% u0)
}

u_1 = Compute_uk(1, u0, U, D, U_inv) # Market shares after one month
u_1
```

```
##      [,1]
## [1,] 0.33
## [2,] 0.32
## [3,] 0.34
```

```
u_2 = Compute_uk(2, u0, U, D, U_inv) # Market shares after two months
u_2
```

```
##       [,1]
## [1,] 0.329
## [2,] 0.334
## [3,] 0.330
```

```
u_3 = Compute_uk(3, u0, U, D, U_inv) # Market shares after three months
u_3
```

```
##        [,1]
## [1,] 0.3307
## [2,] 0.3306
## [3,] 0.3314
```

```
c = U_inv %*% u0 # Equation taken from the live class to later compute u_inf
c
```

```
##             [,1]
## [1,]  0.57322634
## [2,]  0.01641304
## [3,] -0.07505553
```

```
## Compute the steady state market
u_inf = c[1]*U[,1]
u_inf
```

```
## [1] 0.3309524 0.3309524 0.3309524
```

```
# Define method to compute difference between a given month and steady market
Compute_diff = function(u_k, u_inf) {
  return(u_k - u_inf)
}

diff_1 = Compute_diff(u_1, u_inf)
diff_1
```

```
##              [,1]
## [1,] -0.000952381
## [2,] -0.010952381
## [3,]  0.009047619
```

```
diff_2 = Compute_diff(u_2, u_inf)
diff_2
```

```
##              [,1]
## [1,] -0.001952381
## [2,]  0.003047619
## [3,] -0.000952381
```

```
diff_3 = Compute_diff(u_3, u_inf)
diff_3
```

```
##              [,1]
## [1,] -0.000252381
## [2,] -0.000352381
## [3,]  0.000447619
```

```
# Conclusions: The steady market is closed to be reached after the first month.
# For the following months the difference becomes slightly smaller but without huge improveme
nt.
```