Markov Chain Simulations

Ali Taqi

Generating Random Matrices

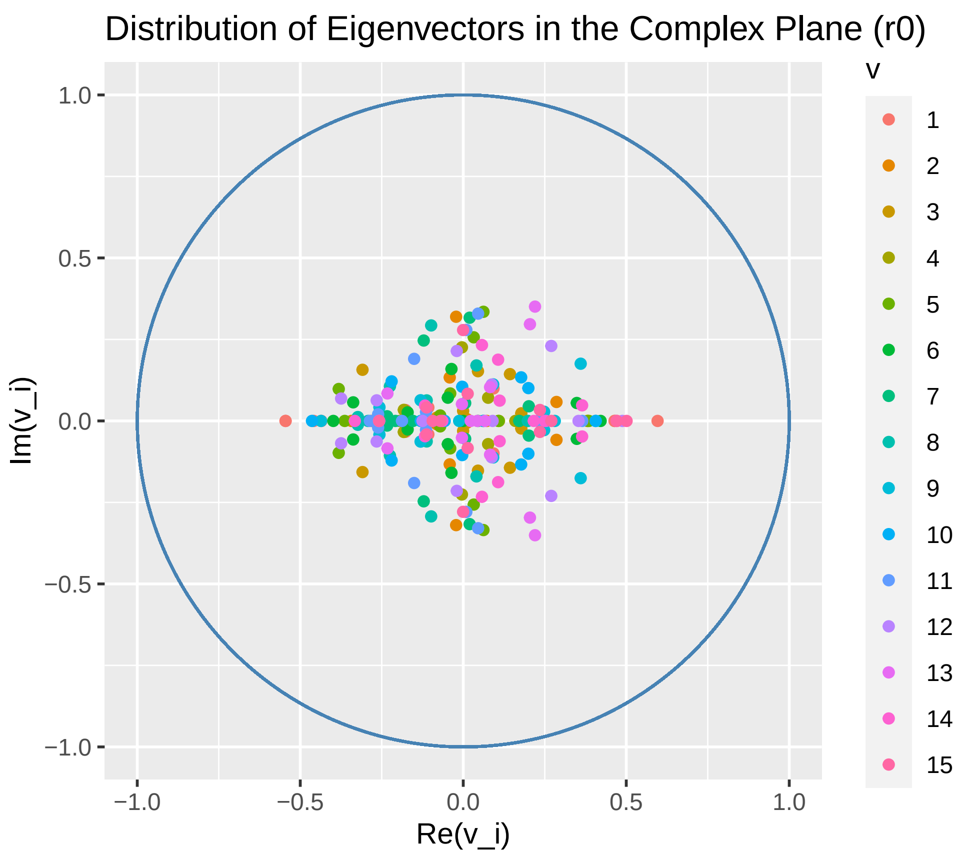
```r
# generates rows of size P which are valid probability distributions
r0 <- function(M){
  prob <- runif(M,0,1)
  prob/sum(prob) # return normalized random row vector
}

r1 <- function(M){
  prob <- runif(M,0,1)
  num_zeros <- sample(1:(M-1),1)
  choices <- sample(1:M, num_zeros)
  prob[choices] <- 0
  prob/sum(prob) # return normalized random row vector
}
```

```r
# initialize random P
rand_M <- function(M,row_fxn){
  P <- matrix(rep(NA, M * M), ncol = M)  # create transition matrix
  for(i in 1:M){P[i,] = row_fxn(M)}
  P
}
```

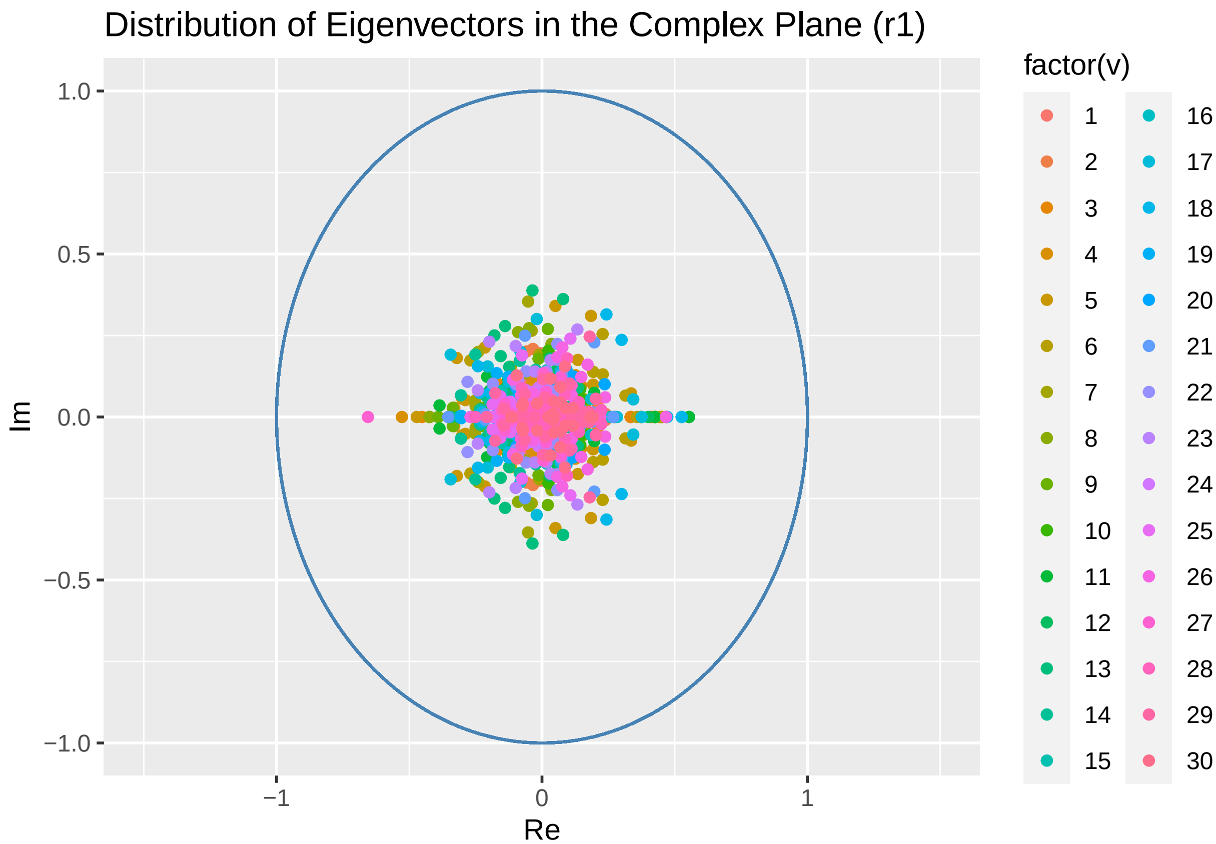
Eigenvectors

```
complex_df <- function(eigenvectors){
  cols <- 3 # set 3 to hold (re,im) pair and whose row it belongs to
  complex <- matrix(rep(NA,cols*M*M), ncol = cols)
  colnames(complex) <- c("Re","Im","v")
  for(i in 1:M){
    for(j in 1:M){
      curr <- eigenvectors[i,j]
      complex[M*(i-1) + j, ] <- c(Re(curr),Im(curr),i)
    }
  }
  data.frame(complex)
}
```

```
M <- 15 # number of states
P <- rand_M(M,r0) # initialize P
eigen_vecs <- data.frame(eigen(P)[2])
complex <- complex_df(eigen_vecs)
complex <- complex
```




```
M <- 30 # number of states
P <- rand_M(M,r1) # initialize P
eigen_vecs <- data.frame(eigen(P)[2])
complex <- complex_df(eigen_vecs)
```



Distribution of Eigenvectors in the Complex Plane (r1)
Im
1.0
0.5
0.0
−0.5
−1.0
−1
0
1
Re
factor(v)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

Convergence

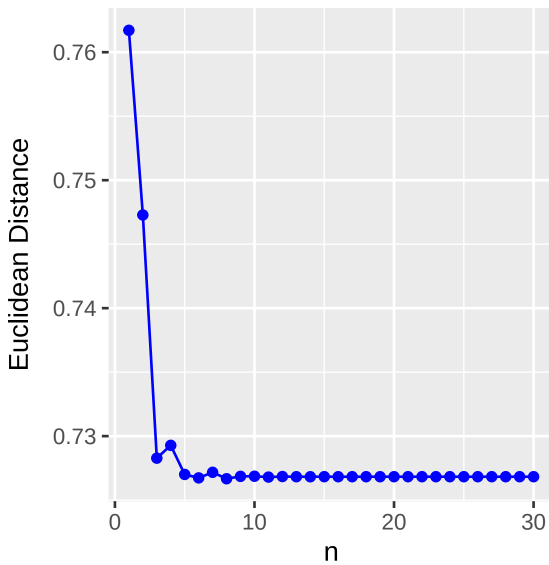
```r
set.seed(23)
M <- 12
P <- rand_M(M,r1)
eigen_vecs <- data.frame(eigen(P)[2])
st <- eigen_vecs[1,1] # choose reference vector to find distance from
it <- 30 # number of iterations of transition matrix
```

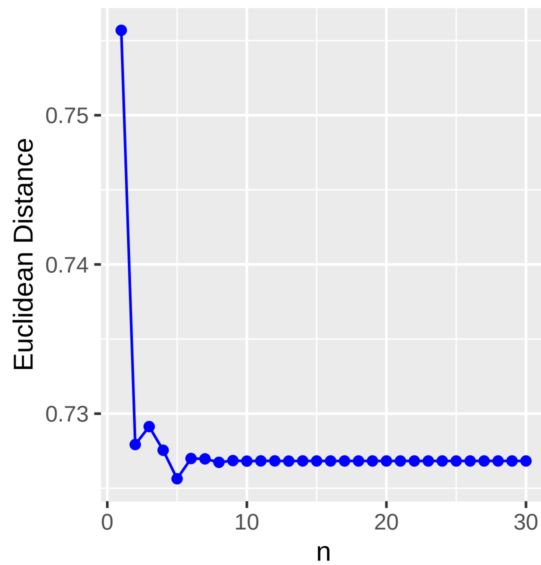
```r
evolve <- function(pi){
  # simulate and record evolution of pi
  vals <- matrix(rep(NA, (M+1) * it), ncol = (M+1))
  # rename the columns
  str_vec <- rep(NA, M)
  for(i in 1:M){str_vec[i] = paste("x",i,sep="")}
  colnames(vals) <- c("n",str_vec)
  # evolve pi
  for(i in 1:it){
    vals[i, ] = c(i, pi %*% matrix.power(P,i))
  }
  #store the values in a dataframe
  vals_ <- data.frame(vals) # store indices as base df in case they are needed
  vals <- subset(vals_, select = -c(n))
  vals
}
```

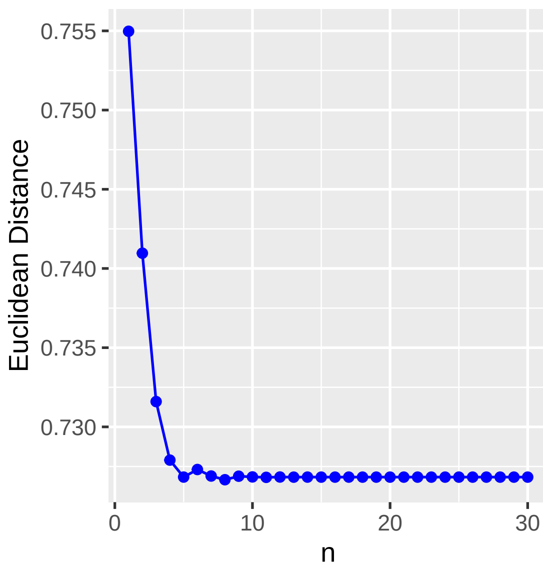
```r
distance <- function(pi,ref_dist){
  #plot difference from a reference/stationary distribution
  diff <- rbind(evolve(pi),ref_dist)
  dist_vec <- rep(0, it)
  for(i in 1:it){
    curr_dist <- stats::dist(diff[c(i,it+1),], method = "euclidean")
    dist_vec[i] <- curr_dist
  }
  data.frame(dist_vec)
}
plot_d <- function(init,ref){
  dist_vec <- distance(init,ref)
  dist_plot <- ggplot(dist_vec, mapping = aes(x = 1:it, y = dist_vec)) +
    geom_point(color = col_str) + geom_line(color = col_str) +
    labs(x = "n", y = "Euclidean Distance")
  dist_plot
}
```

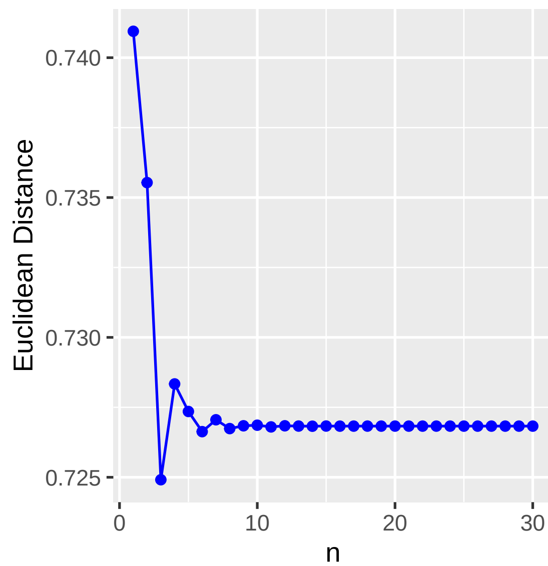
Simulating convergence using initial distributions of row function 1

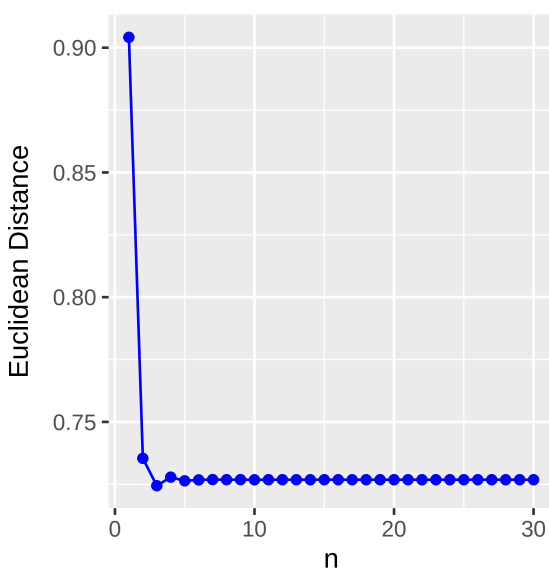
```r
p1 <- plot_d(r1(M),st)
p2 <- plot_d(r1(M),st)
p3 <- plot_d(r1(M),st)
p4 <- plot_d(r1(M),st)
p5 <- plot_d(r1(M),st)
p6 <- plot_d(r1(M),st)
(p1+p2)/(p3+p4)/(p5+p6)
```

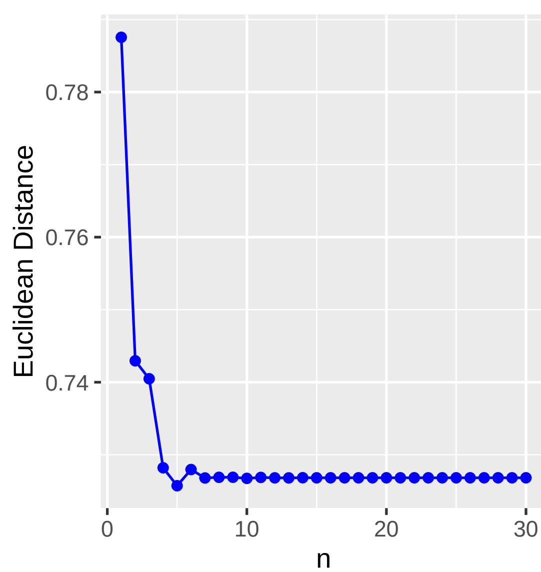

0.76
0.75
0.74
0.73
0
10
20
30
n
Euclidean Distance


0.75
0.74
0.73
0
10
20
30
n
Euclidean Distance


0.755
0.750
0.745
0.740
0.735
0.730
0
10
20
30
n
Euclidean Distance

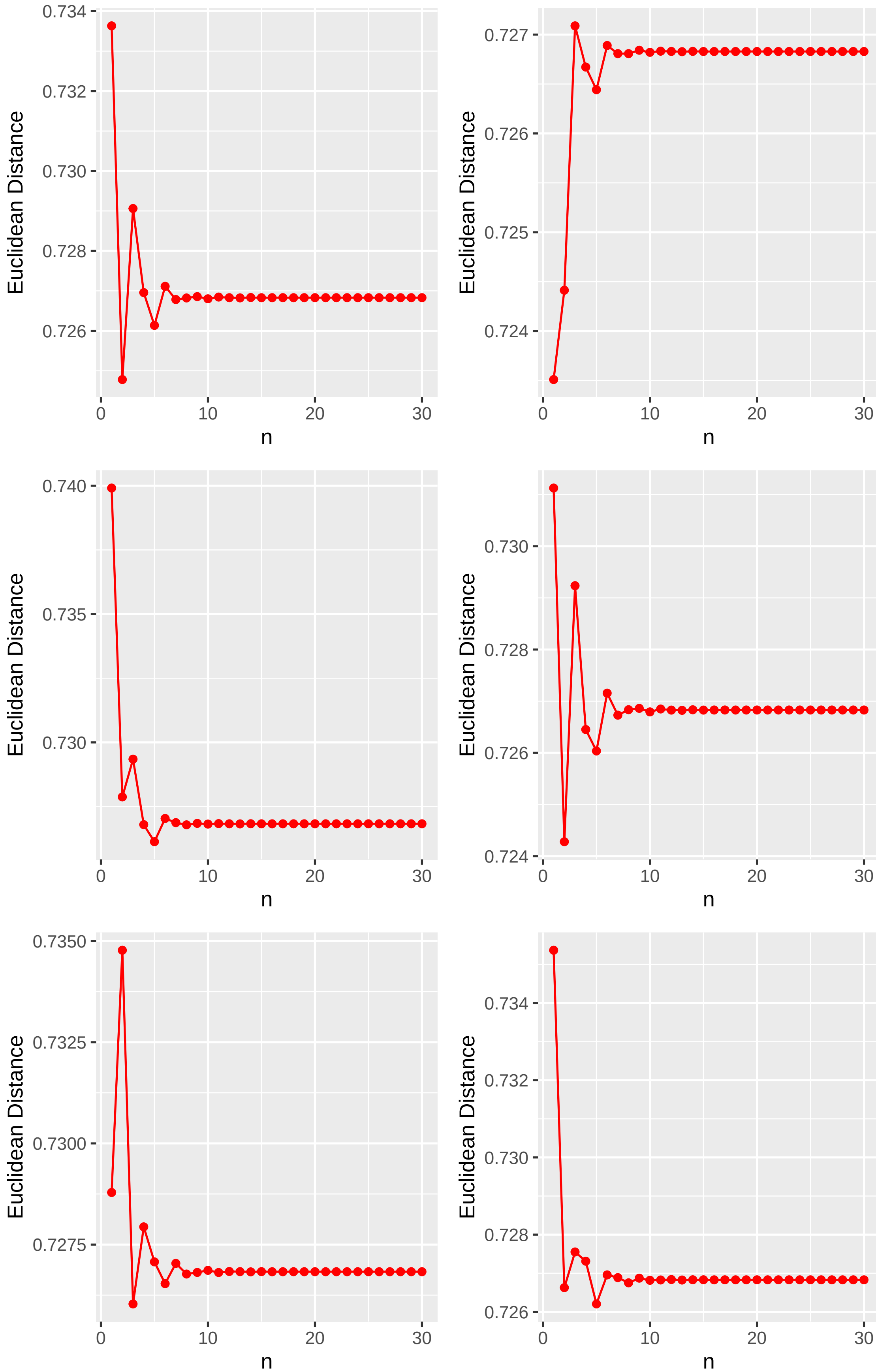

0.740
0.735
0.730
0.725
0
10
20
30
n
Euclidean Distance


0.90
0.85
0.80
0.75
0
10
20
30
n
Euclidean Distance


0.78
0.76
0.74
0
10
20
30
n
Euclidean Distance

Simulating convergence using initial distributions of row function 0

```
p1 <- plot_d(r0(M),st)
p2 <- plot_d(r0(M),st)
p3 <- plot_d(r0(M),st)
p4 <- plot_d(r0(M),st)
p5 <- plot_d(r0(M),st)
p6 <- plot_d(r0(M),st)

(p1+p2)/(p3+p4)/(p5+p6)
```


Euclidean Distance
0.734
0.732
0.730
0.728
0.726
n
0
10
20
30
Euclidean Distance
0.727
0.726
0.725
0.724
n
Euclidean Distance
0.740
0.735
0.730
n
Euclidean Distance
0.730
0.728
0.726
0.724
n
Euclidean Distance
0.7350
0.7325
0.7300
0.7275
n
Euclidean Distance
0.734
0.732
0.730
0.728
0.726
n