Computational Eigenvector Simulation

Ali Taqi

```
set.seed(23)
# set parameters
M <- 2
mu <- 0
sd <- 1
# generate matrix
P <- RM_stoch(M, symm = T, sparsity = F)
# generate matrix
P_1 <- RM_normal(M, normal_args = c(mu, sd),symm = T)
P
```

```
##           [,1]      [,2]
## [1,] 0.5977228 0.4196847
## [2,] 0.4196847 0.5660087
```

Step 1: Get the batch

Step 2: Get the batch

```
evolve <- function(pi, P, steps){
  steps <- it
  # 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))
  rbind(pi,vals)
}
```

```
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
}
```

Simulate

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

```
#set.seed(23)
pi <- r_zeros(M)
evol <- evolve(pi, P, steps = 10)
#dist_vec <- distance(init,st)
```

```
plot <- ggplot(evol) +
  geom_point(mapping = aes(x = x1, y = x2), color = "red") +
  geom_point(mapping = aes(x = x1, y = x3), color = "blue") +
  geom_point(mapping = aes(x = x1, y = x4), color = "green")
```