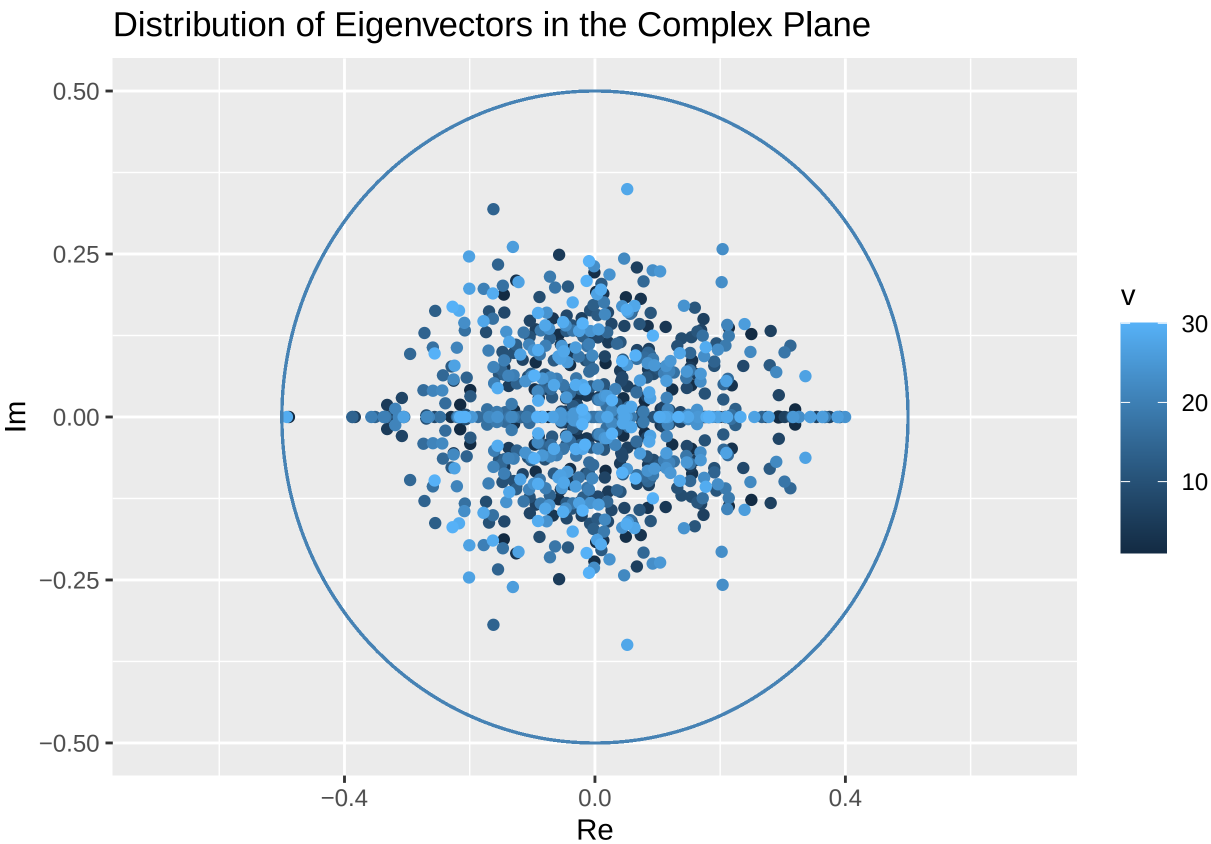
Markov Chain Simulation

```r
set.seed(27)
M <- 30  # assign number of states
P <- matrix(rep(NA, M * M), ncol = M)  # create transition matrix

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

# initialize P
for(i in 1:M){
  P[i,] = r_probdist(M)
  }
```

```r
eig_P <- eigen(P)
eig_vectors <- eig_P[2]
evec <- data.frame(eig_vectors)

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 <- evec[i,j]
    complex[M*(i-1) + j, ] <- c(Re(curr),Im(curr),i)
  }
}
```

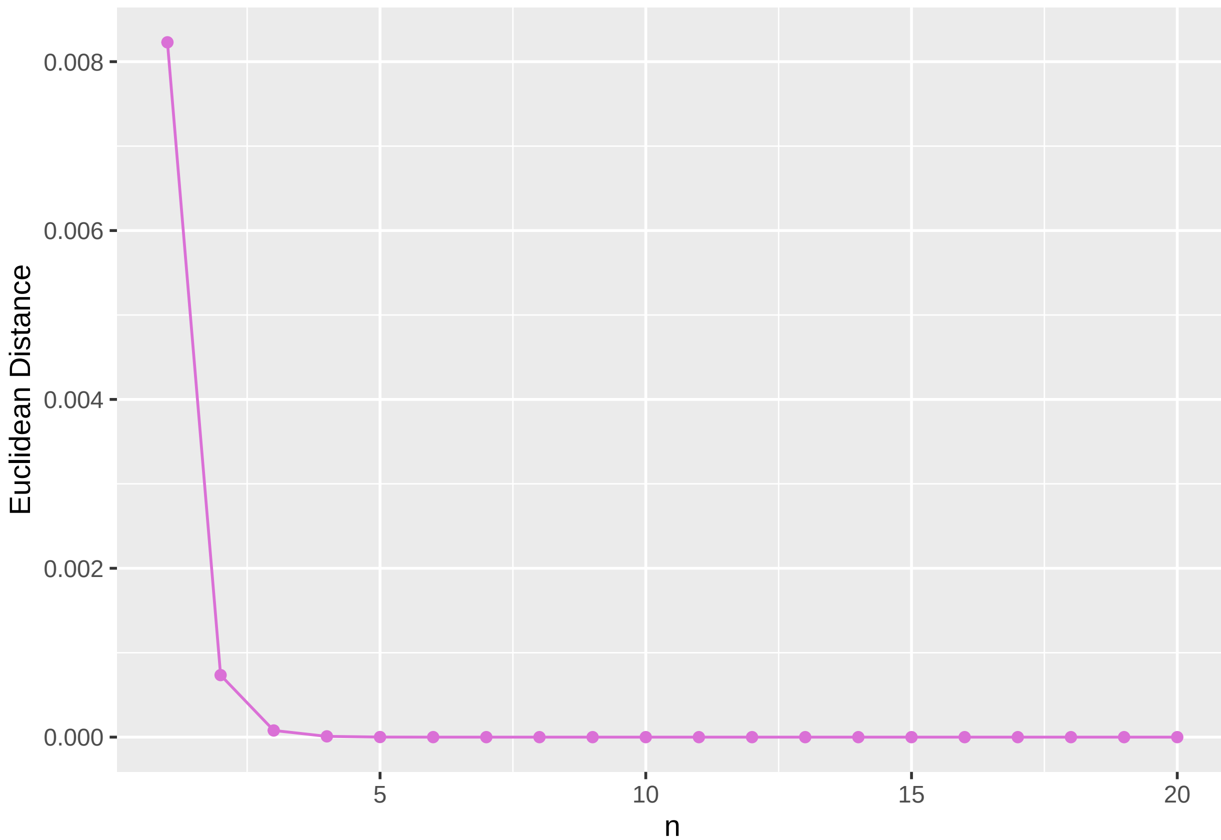
```r
r <- 0.5
ep <- 0.2
ggplot(complex) +
  geom_point(aes(x = Re, y = Im, color = v)) +
  labs(x = "Re", y = "Im", title = "Distribution of Eigenvectors in the Complex Plane") +
  xlim(-(r+ep),r+ep) + ylim(-r,r) +
  ggforce::geom_circle(aes(x0=0,y0=0,r=r), color = "steelblue")
```




```r
set.seed(23)
it <- 20 # set number of iterations of transition matrix
pi <- r_probdist(M) # create some initial distribution

# simulate and record evolution of pi
vals <- matrix(rep(NA, (M+1) * it), ncol = (M+1))
for(i in 1:it){
  vals[i, ] = c(i, pi %*% matrix.power(P,i))
}

# 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)

#store the values in a dataframe
vals_ <- data.frame(vals)
vals <- subset(vals_, select = -c(n))
```

```r
#plot difference from a reference/stationary distribution
ref_dist <- vals[it,]
diff <- rbind(vals,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
```

```
}

dist_vec <- data.frame(dist_vec)
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
```




```
#plot_vals <- ggplot() + geom_point(data = vals, aes(x = x, y = y, color = n))
#plot_vals
```