FinalProjCode.R

Jasper

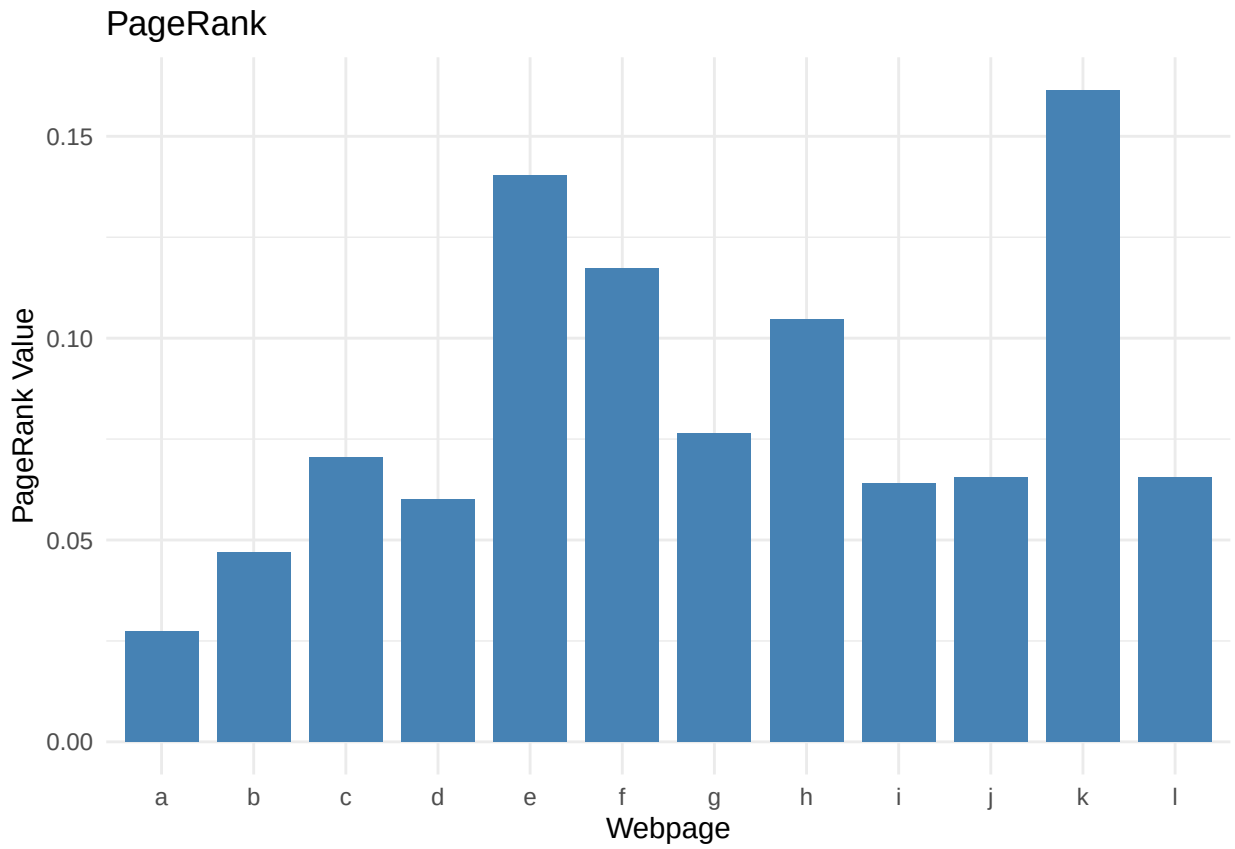
2024-08-04

```
library(MASS)
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

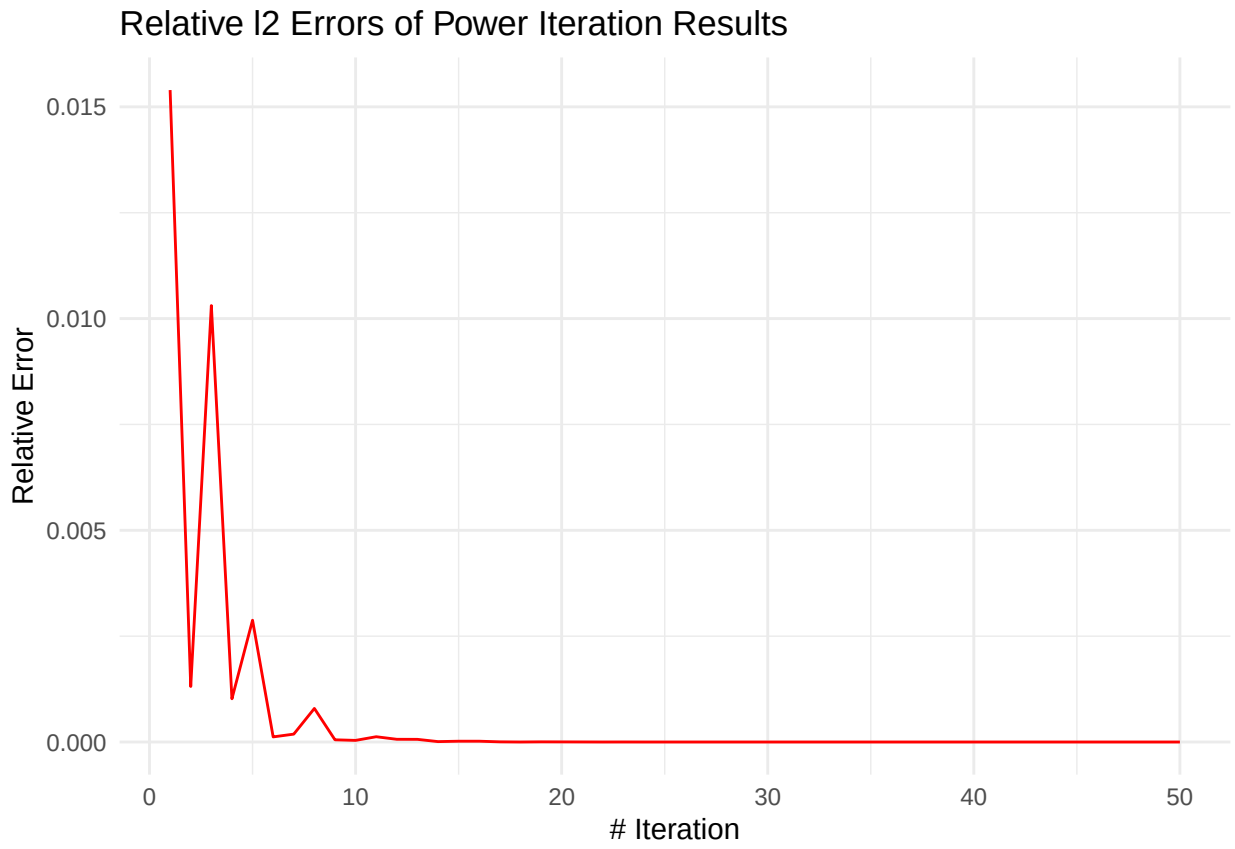
```
# Creating hyperlink matrix
G = t(fractions(matrix(c(0, 0, 0, 1/2, 0, 0, 0, 1/2, 0, 0, 0, 0,
                         1/3, 0, 0, 1/3, 1/3, 0, 0, 0, 0, 0, 0, 0,
                         0, 1/2, 0, 0, 1/2, 0, 0, 0, 0, 0, 0, 0,
                         0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
                         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                         0, 0, 1/2, 0, 1/2, 0, 0, 0, 0, 0, 0, 0,
                         0, 0, 0, 1/4, 1/4, 1/4, 0, 1/4, 0, 0, 0, 0,
                         0, 0, 0, 0, 0, 0, 0, 0, 1/2, 0, 1/2, 0,
                         0, 0, 0, 0, 0, 0, 1/2, 0, 0, 0, 1/2, 0,
                         0, 0, 0, 0, 0, 1/2, 1/2, 0, 0, 0, 0, 0,
                         0, 0, 0, 0, 0, 1/3, 0, 0, 0, 1/3, 0, 1/3,
                         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0),
                         nrow = 12, ncol = 12)))

# Eigenvalue Decomposition
ev1 = eigen(t(G))
evalue1 = ev1$values[1] # Choosing largest eigenvalue
evector1 = ev1$vectors[,1] # Eigenvector corresponding to largest eigenvalue

# Adding artificial links to dangling node
Gnew = G
Gnew[5,] = rep(1/12, 12)
ev2 = eigen(t(Gnew)) # Eigenvalue decomposition of new G matrix
evalue2 = ev2$values[1]
evector2 = ev2$vectors[,1]

# Normalizing eigenvector
evector2n = evector2/sum(evector2)

# Ordering eigenvector
ranks = data.frame(webpage = letters[1:12], pagerank = Re(evector2n))
ranks_sorted = ranks[order(ranks$pagerank, decreasing = TRUE),]
ranks_plot = ggplot(data = ranks, aes(x = webpage, y = pagerank)) +
                geom_bar(stat = "identity", fill = "steelblue", width = 0.8) +
                labs(title = "PageRank",
```

```
                    x = "Webpage",
                    y = "PageRank Value") +
               theme_minimal()
ranks_plot
```




```
# Power Iteration
powermethod = function(A, k){
  # function that returns largest eigenvalue and eigenvector of matrix A
  # given k iterations
  v = rep(1/nrow(A), nrow(A)) # initialize vector
  for(i in 1:k){
    w = A %*% v
    v = w/norm(w, type = c("1"))
    lambda = t(v) %*% A %*% v
  }
  return(list(eigenvalue = lambda, eigenvector = v))
}

# Iterating through different k's
ret = c()
error = c()
abs = norm(as.matrix(Re(evector2n)), type = "2") # used for calculating errors of iterated eigenvector
for(k in 1:50){
  ev_power = powermethod(t(Gnew), k)
  ret = c(ret, ev_power$eigenvalue)
```

```
  l2 = norm(ev_power$eigenvector, type = "2")
  error = c(error, abs - l2)
}
power_df = data.frame(iteration = c(1:50), eigenvalue = ret, error = error) # converges at 28 iteration
evector_power = powermethod(t(Gnew), 28)$eigenvector # should be equivalent to evector2n

# Plot of l2 errors of power iteration results
error_plot = ggplot(data = power_df, aes(x = iteration, y = abs(error))) +
                geom_line(col = "red") +
                labs(title = "Relative l2 Errors of Power Iteration Results",
                     x = "# Iteration",
                     y = "Relative Error") +
                theme_minimal()
error_plot
```




```
# Part 9
alpha <- 0.85
n <- nrow(G)
E <- matrix(1/n, n, n)
G_tilde <- alpha * G + (1 - alpha) * E

# Part 10
ev3 <- eigen(t(G_tilde))
evalue3 <- ev3$values[1]
evector3 <- abs(ev3$vectors[,1])
```

```r
# Part 11
rankings_comparison <- data.frame(
  webpage = letters[1:12],
  pagerank_G = Re(evector2n),
  pagerank_G_tilde = evector3 / sum(evector3)
)

# Sort pagerank
rankings_comparison_sorted <- rankings_comparison[order(rankings_comparison$pagerank_G_tilde, decreasing
print(rankings_comparison_sorted)
```

```
##    webpage pagerank_G pagerank_G_tilde
## 11       k 0.16149619       0.15965440
## 5        e 0.14033651       0.13683322
## 6        f 0.11740214       0.11527365
## 8        h 0.10456981       0.10651347
## 7        g 0.07644790       0.07700645
## 3        c 0.07039578       0.06958456
## 9        i 0.06397961       0.06537146
## 10       j 0.06552677       0.06533434
## 12       l 0.06552677       0.06533434
## 4        d 0.06010034       0.06207266
## 2        b 0.04689260       0.04761099
## 1        a 0.02732557       0.02941046
```

```r
# Part 12
keywords <- c("Ash", "Butternut", "Cherry", "Elm", "Katsura", "Magnolia", "Teak", "Ginkgo",
              "Fir", "Hickory", "Pine", "Willow", "Redwood", "Sassafras", "Oak", "Spruce",
              "Aspen")

# List of keywords for each webpage
webpages <- list(
  A = c("Ash", "Butternut", "Cherry", "Elm", "Katsura", "Magnolia", "Teak", "Ginkgo"),
  B = c("Butternut", "Fir", "Hickory", "Magnolia", "Pine", "Willow", "Redwood", "Sassafras"),
  C = c("Ash", "Elm", "Hickory", "Katsura", "Oak", "Ginkgo", "Redwood"),
  D = c("Butternut", "Cherry", "Fir", "Spruce", "Teak", "Aspen", "Sassafras"),
  E = c("Cherry", "Hickory", "Oak", "Pine", "Willow", "Redwood"),
  f = c("Ash", "Fir", "Magnolia", "Spruce", "Ginkgo", "Redwood", "Aspen", "Sassafras"),
  G = c("Ash", "Butternut", "Oak", "Spruce", "Ginkgo", "Redwood"),
  H = c("Ash", "Cherry", "Hickory", "Willow", "Redwood", "Aspen"),
  I = c("Elm", "Fir", "Katsura", "Magnolia", "Pine", "Spruce", "Sassafras"),
  J = c("Magnolia", "Oak", "Willow", "Redwood", "Aspen", "Sassafras"),
  K = c("Cherry", "Elm", "Fir", "Hickory", "Teak", "Ginkgo", "Redwood", "Sassafras"),
  L = c("Butternut", "Elm", "Katsura", "Oak", "Pine", "Spruce", "Teak", "Ginkgo", "Aspen", "Sassafras")
)

# term-document matrix
D <- matrix(0, nrow = length(keywords), ncol = length(webpages))
rownames(D) <- keywords
colnames(D) <- names(webpages)

# Fill
for (j in seq_along(webpages)) {
```

```
  for (keyword in webpages[[j]]) {
    D[keyword, j] <- 1
  }
}

print(D)
```

```
##           A B C D E f G H I J K L
## Ash       1 0 1 0 0 1 1 1 0 0 0 0
## Butternut 1 1 0 1 0 0 1 0 0 0 0 1
## Cherry    1 0 0 1 1 0 0 1 0 0 1 0
## Elm       1 0 1 0 0 0 0 0 1 0 1 1
## Katsura   1 0 1 0 0 0 0 0 1 0 0 1
## Magnolia  1 1 0 0 0 1 0 0 1 1 0 0
## Teak      1 0 0 1 0 0 0 0 0 0 1 1
## Ginkgo    1 0 1 0 0 1 1 0 0 0 1 1
## Fir       0 1 0 1 0 1 0 0 1 0 1 0
## Hickory   0 1 1 0 1 0 0 1 0 0 1 0
## Pine      0 1 0 0 1 0 0 0 1 0 0 1
## Willow    0 1 0 0 1 0 0 1 0 1 0 0
## Redwood   0 1 1 0 1 1 1 1 0 1 1 0
## Sassafras 0 1 0 1 0 1 0 0 1 1 1 1
## Oak       0 0 1 0 1 0 1 0 0 1 0 1
## Spruce    0 0 0 1 0 1 1 0 1 0 0 1
## Aspen     0 0 0 1 0 1 0 1 0 1 0 1
```

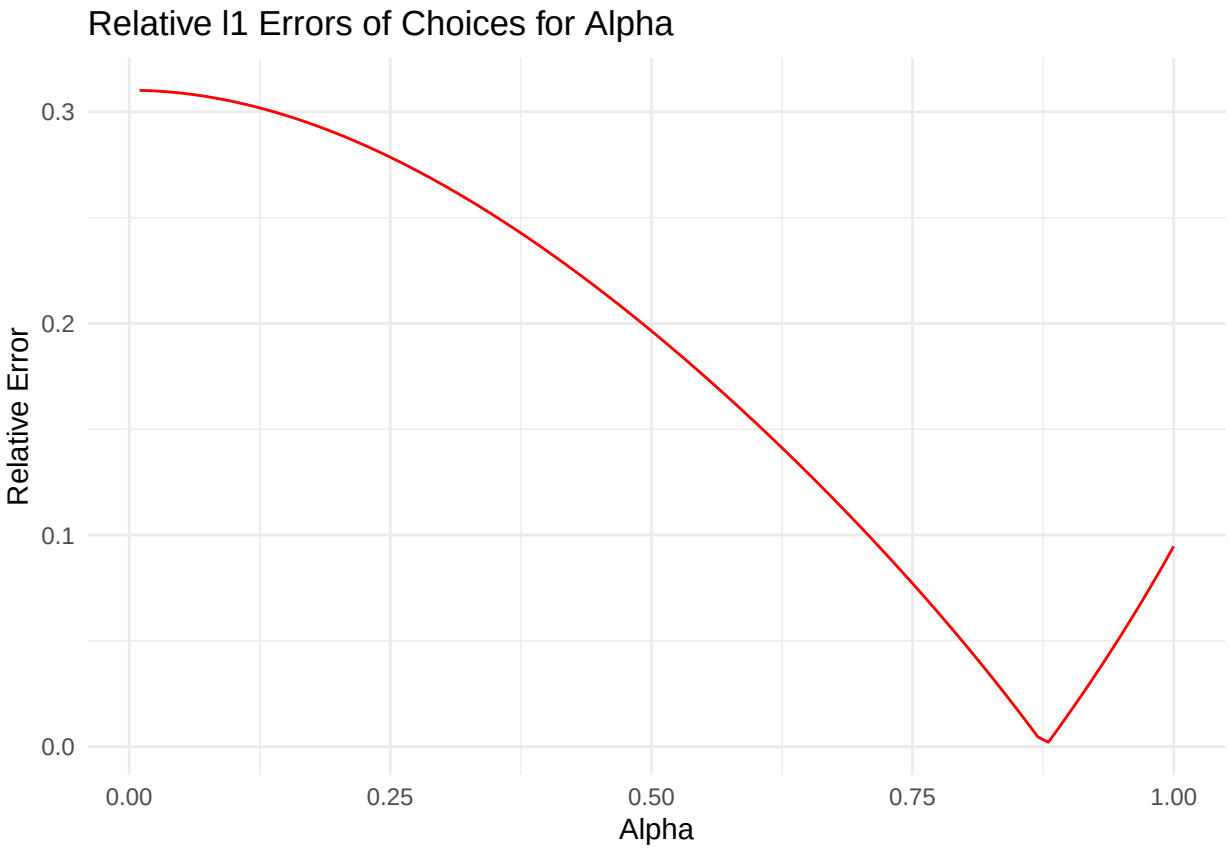
```
# Part 13 and Part 14 and Part 15 for each user query

# 1 "Ash"
q_ash <- rep(0, length(keywords))
names(q_ash) <- keywords
q_ash["Ash"] <- 1
d_ash <- t(q_ash) %*% D
d_ash_df <- data.frame(webpage = colnames(D), score = as.numeric(d_ash))
d_ash_df_sorted <- d_ash_df[order(-d_ash_df$score),]

# 2 "Fir" OR "Hickory"
q_fir_hickory <- rep(0, length(keywords))
names(q_fir_hickory) <- keywords
q_fir_hickory[c("Fir", "Hickory")] <- 1
d_fir_hickory <- t(q_fir_hickory) %*% D
d_fir_hickory_df <- data.frame(webpage = colnames(D), score = as.numeric(d_fir_hickory))
d_fir_hickory_df_sorted <- d_fir_hickory_df[order(-d_fir_hickory_df$score),]

# 3"Katsura" AND "Oak"
q_katsura_oak <- rep(0, length(keywords))
names(q_katsura_oak) <- keywords
q_katsura_oak[c("Katsura", "Oak")] <- 1
d_katsura_oak <- t(q_katsura_oak) %*% D
d_katsura_oak_df <- data.frame(webpage = colnames(D), score = as.numeric(d_katsura_oak))
# Ensure both keywords are present by checking if the webpage contains both "Katsura" and "Oak"
webpages_with_both <- apply(D[c("Katsura", "Oak"), ] == 1, 2, all)
d_katsura_oak_df_filtered <- d_katsura_oak_df[webpages_with_both, ]
```

```r
d_katsura_oak_df_sorted <- d_katsura_oak_df_filtered[order(-d_katsura_oak_df_filtered$score),]

# 4 "Aspen" and not "Sassafras"
q_aspen_not_sassafras <- rep(0, length(keywords))
names(q_aspen_not_sassafras) <- keywords
q_aspen_not_sassafras["Aspen"] <- 1
d_aspen_not_sassafras <- t(q_aspen_not_sassafras) %*% D
d_aspen_not_sassafras_df <- data.frame(webpage = colnames(D), score = as.numeric(d_aspen_not_sassafras))
# Exclude webpages containing "Sassafras"
webpages_without_sassafras <- D["Sassafras", ] == 0
d_aspen_not_sassafras_df_filtered <- d_aspen_not_sassafras_df[webpages_without_sassafras, ]
d_aspen_not_sassafras_df_sorted <- d_aspen_not_sassafras_df_filtered[order(-d_aspen_not_sassafras_df_fil

# Display results
list(
  ash = d_ash_df_sorted,
  fir_or_hickory = d_fir_hickory_df_sorted,
  katsura_and_oak = d_katsura_oak_df_sorted,
  aspen_not_sassafras = d_aspen_not_sassafras_df_sorted
)
```

```
## $ash
##    webpage score
## 1        A     1
## 3        C     1
## 6        f     1
## 7        G     1
## 8        H     1
## 2        B     0
## 4        D     0
## 5        E     0
## 9        I     0
## 10       J     0
## 11       K     0
## 12       L     0
##
## $fir_or_hickory
##    webpage score
## 2        B     2
## 11       K     2
## 3        C     1
## 4        D     1
## 5        E     1
## 6        f     1
## 8        H     1
## 9        I     1
## 1        A     0
## 7        G     0
## 10       J     0
## 12       L     0
##
## $katsura_and_oak
##    webpage score
```

```
## 3        C     2
## 12       L     2
##
## $aspen_not_sassafras
##   webpage score
## 8       H     1
## 1       A     0
## 3       C     0
## 5       E     0
## 7       G     0
```

```r
# Iterating through alphas
n = nrow(G)
E = matrix(1/n, n, n)
l1_eigen = norm(as.matrix(Re(evector2), type = "1"))
results = c()
error_alpha = c()
for (alpha in seq(0.01, 1, 0.01)){
  G_tilde = alpha * G + (1 - alpha) * E
  ev_alpha = eigen(t(G_tilde))
  evector_alpha = Re(ev_alpha$vectors[,1])
  l1_alpha = norm(as.matrix(evector_alpha), type = "1")
  results = c(results, l1_alpha)
  error_alpha = c(error_alpha, l1_eigen - l1_alpha)
}

# Calculating alpha errors
df_alpha = data.frame(alpha = seq(0.01, 1, 0.01), error = error_alpha)

alpha_plot = ggplot(data = df_alpha, aes(x = alpha, y = abs(error))) +
  geom_line(col = "red") +
  labs(title = "Relative l1 Errors of Choices for Alpha",
       x = "Alpha",
       y = "Relative Error") +
  theme_minimal()
alpha_plot
```


Relative l1 Errors of Choices for Alpha
Relative Error
0.3
0.2
0.1
0.0
0.00
0.25
0.50
0.75
1.00
Alpha