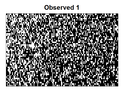
```
library(imager)
library(jpeg)
library(imager)
library(MASS)
library(pracma)
library(gridExtra)
library(latex2exp)

source("ERPCA_optimization.R")
source("ePCA.R")

# Load the reduced defect image
reduced_defect2 <- readJPEG("reduced_defect2.jpg") # Replace with your file path

# Normalize the image to get probabilities
defect2_prob <- reduced_defect2 / max(reduced_defect2)

# Initialize an array to store the samples
defect2_sample <- array(NA, dim = c(nrow(defect2_prob), ncol(defect2_prob), 500))

# Generate samples using binomial distribution
for (i in 1:nrow(defect2_prob)) {
  for (j in 1:ncol(defect2_prob)) {
    defect2_sample[i, j, ] <- rbinom(500, 1, defect2_prob[i, j])
  }
}
# Plot the 'truth' image in a single panel
image(defect2_prob, col = grey(seq(0, 1, length = 256)), xaxt = 'n', yaxt = 'n', main = "Truth")
```

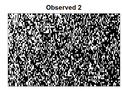
Truth

```
# Setting up the plotting area for 4 images in a 2x2 format
par(mfrow = c(2, 2), mar = c(2, 2, 2, 2))  # Smaller margins

# Plot the first 4 'observed' images in a 2x2 layout
for (k in 1:4) {
  image(defect2_sample[, , k], col = grey(seq(0, 1, length = 256)), xaxt = 'n', yaxt = 'n', main = paste("Observe
d", k))
}
```

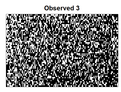
Observed 1



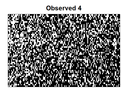
Observed 2



Observed 3



Observed 4



```
# Reset the plotting layout
par(mfrow = c(1, 1), mar = c(5, 4, 4, 2) + 0.1)  # Reset to default margins
```

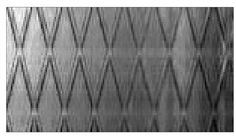
Hyperparameters Tuning for e^{RPCA}

```
train_obs <- defect2_sample[, , 1:10]
m1 <- dim(train_obs)[1]
m2 <- dim(train_obs)[2]

# Initialize hyperparameters
alpha0 <- 1
beta0 <- 1 / sqrt(max(m1, m2))
mu0 <- (m1 * m2) / (4 * sum(abs(train_obs), na.rm = TRUE))

# Hyperparameter adjustment rates
eta_alpha <- 0.5
eta_beta <- 0.08

# Arrays to store selections and results
alpha_select <- rep(-100, 20)
beta_select <- rep(-100, 20)
rank_all <- rep(-100, 20)
sparse_all <- rep(-100, 20)

# Prior values for rank and sparsity
rank_prior <- 10
sparse_prior <- 0.1

# Set initial values
alpha_select[1] <- alpha0
beta_select[1] <- beta0

# Hyperparameter selection loop
for (i in 1:20) { # Assuming 20 runs of hyperparameter selection
  mle_results <- mc_function_mle(defect2_sample, 100, mu0, alpha_select[i], beta_select[i])
  mle_results$S_all[,,100][abs(mle_results$S_all[,,100]) <= 1e-5] <- 0
  rank_all[i] <- rankMatrix(mle_results$L_all[,,100])
  sparse_all[i] <- sum(mle_results$S_all[,,100] != 0) / (m1 * m2)

  # Adjust alpha and beta based on rank and sparsity
  if ((rank_all[i] > rank_prior) & (sparse_all[i] > sparse_prior)) {
    alpha_select[i + 1] <- alpha_select[i] + eta_alpha
    beta_select[i + 1] <- beta_select[i] + eta_beta
  } else if ((rank_all[i] <= rank_prior) & (sparse_all[i] > sparse_prior)) {
    alpha_select[i + 1] <- alpha_select[i]
    beta_select[i + 1] <- beta_select[i] + eta_beta
  } else if ((rank_all[i] > rank_prior) & (sparse_all[i] <= sparse_prior)) {
    alpha_select[i + 1] <- alpha_select[i] + eta_alpha
    beta_select[i + 1] <- beta_select[i]
  } else if ((rank_all[i] <= rank_prior) & (sparse_all[i] <= sparse_prior)) {
    alpha_select[i + 1] <- alpha_select[i]
    beta_select[i + 1] <- beta_select[i]
  }

  # Break loop if no change in hyperparameters
  if (alpha_select[i + 1] == alpha_select[i] & beta_selectt[i + 1] == beta_select[i]) {
    break
  }
}
```

e^{RPCA} Results

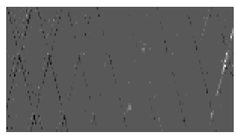
```
# Final selection of alpha and beta from the hyperparameter selection process
alpha0 <- tail(alpha_select[alpha_select != -100], 1)
beta0 <- tail(beta_select[beta_select != -100], 1)
mu0 <- (m1 * m2) / (4 * sum(abs(train_obs), na.rm = TRUE))

# Running the Monte Carlo function for ERPCA with selected hyperparameters
mle_results <- mc_function_mle(defect2_sample, 100, mu0, alpha0, beta0)
mle_results$S_all[,,100][abs(mle_results$S_all[,,100]) <= 1e-5] <- 0

# Plotting the estimated L matrix
image(mle_results$L_all[,,100], col = grey(seq(0, 1, length = 256)), main = TeX(r'(Estimated L ($e^{RPCA}$)
axt = 'n', yaxt = "n", cex.main = 2)
```

Estimated L (e^{RPCA})



```
# Plotting the estimated S matrix
image(mle_results$S_all[,,100], col = grey(seq(0, 1, length = 256)), main = TeX(r'(Estimated S ($e^{RPCA}$)'), x
axt = 'n', yaxt = "n", cex.main = 2)
```

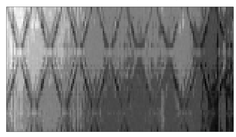
Estimated S (e^{RPCA})



RPCA Results

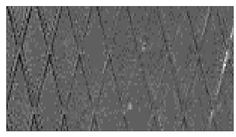
```
# Hyperparameter initialization for RPCA
alpha <- 1
beta <- 1 / sqrt(max(m1, m2))
mu <- (m1 * m2) / (4 * sum(abs(train_obs), na.rm = TRUE))

# Running the Monte Carlo function for RPCA
ls_results <- mc_function_ls(defect2_sample, 100, mu, alpha, beta)

# Plotting the estimated L matrix for RPCA
image(ls_results$L_all[,,100], col = grey(seq(0, 1, length = 256)), main = "Estimated L (RPCA)", xaxt = 'n', yaxt
= "n", font.main = 1, cex.main = 2)
```

Estimated L (RPCA)



```
# Plotting the estimated S matrix for RPCA
image(ls_results$S_all[,,100], col = grey(seq(0, 1, length = 256)), main = "Estimated S (RPCA)", xaxt = 'n', yaxt
= "n", font.main = 1, cex.main = 2)
```

Estimated S (RPCA)



ePCA Results

```
# Running the ePCA algorithm
L_mat <- epca(defect2_sample)

# Plotting the estimated L matrix for ePCA
image(t(L_mat), col = grey(seq(0, 1, length = 256)), main = "Estimated L (ePCA)", xaxt = 'n', yaxt = "n", font.ma
in = 1, cex.main = 2)
```

Estimated L (ePCA)