Question 3

Answer 3.a:

Given code for Matrix A:

```
rm(list=ls())
A <- matrix(rpois(64,5), 8, 8)
A
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    8    3    3   11    7    4    5    3
## [2,]    6    6    8    6    7    5    8    1
## [3,]    5    6    3    6    3    8    4    4
## [4,]    5    6    4    3    6    5    4    3
## [5,]    6    6    7    5    6    8    6    6
## [6,]    2    0    8    3    7    5    4    2
## [7,]    9    4    9    6    2    7    5    5
## [8,]    5    6    3    3    4    5    6    2
```

After looking for rpois() help using ?rpois we see the following

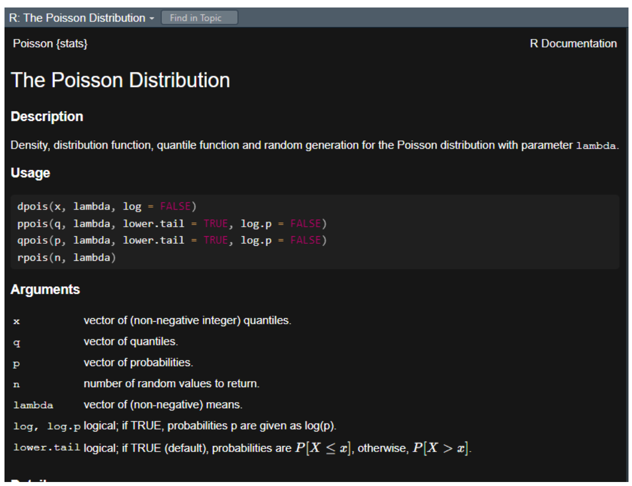

Figure 1: ?rpois

Here are things happening in the give code:

1. rm(list=ls()): This line clears the workspace, removing any existing variables.
2. A <- matrix(rpois(64, 5), 8, 8): This line generates a matrix A with dimensions 8 x 8.
3. It uses the rpois() function, which generates random values from the Poisson distribution. The function takes two arguments: the number of values to generate (in this case, 64) and the mean parameter (in this case, 5). So, each element of matrix A is obtained by generating a random Poisson-distributed value with a mean of 5.

Question 3

Answer 3.b:

Here is the given code for matric C:

```
C= A %*% t(A)
C
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]  302  268  218  194  264  168  259  184
## [2,]  268  311  223  218  294  202  280  211
## [3,]  218  223  211  177  247  137  234  172
## [4,]  194  218  177  172  227  140  205  161
## [5,]  264  294  247  227  318  201  299  214
## [6,]  168  202  137  140  201  171  187  124
## [7,]  259  280  234  205  299  187  317  197
## [8,]  184  211  172  161  214  124  197  160
```

To generate eigen values and eigenvectors of matrix:

```
# eigenvalues and eigenvectors
eigen_result <- eigen(C)
```

Eigenvalues:

```
print(eigen_result$values)
```

```
## [1] 1764.498447   66.799644   57.736922   42.401697   21.153744    6.431771
## [7]    1.876005    1.101770
```

Eigenvectors:

```
print(eigen_result$vectors)
```

```
##            [,1]        [,2]       [,3]         [,4]       [,5]        [,6]
## [1,] -0.3776107  0.59472801  0.6682629  0.115693133 -0.1170024  0.12939585
## [2,] -0.4067870 -0.22374582  0.2383490 -0.328605293  0.5724334 -0.31311857
## [3,] -0.3286263  0.32079381 -0.3626590 -0.017435116 -0.3891652 -0.63092871
## [4,] -0.3024482 -0.02492948 -0.1408745 -0.373456584 -0.1346622  0.62246438
## [5,] -0.4183882 -0.13606119 -0.2719944  0.002198289 -0.3455004  0.22660064
## [6,] -0.2686620 -0.68260834  0.3800344  0.121894933 -0.3914089 -0.16171263
## [7,] -0.4025903 -0.03724359 -0.2722184  0.738012593  0.3989418  0.14346655
## [8,] -0.2883703  0.08277910 -0.2296459 -0.423478526  0.2394180 -0.03998827
##             [,7]        [,8]
## [1,]  0.11306766  0.02305091
## [2,] -0.22643514 -0.37772254
## [3,] -0.31956370  0.07502656
## [4,] -0.55440285  0.18875863
## [5,]  0.43175276 -0.61263698
## [6,]  0.03113688  0.34965796
## [7,] -0.08229062  0.17680382
## [8,]  0.57646133  0.53543551
```

Question 3

Answer 3.c:

Since all of the eigenvalues of matrix C are positive, we can conclude that matrix C is positive definite.

Question 3

Answer 3.d:

Transferring the values of eigenvectors in a variable U

```
U <- eigen_result$vectors
```

```r
# Creating U transpose
U_transpose <- t(U)

#U*U_transpose product
U_product <- U %*% U_transpose

# Identity matrix of the same size as U_product
I <- diag(nrow(U_product))

# View the product
U_product
```

```
##               [,1]          [,2]          [,3]          [,4]          [,5]
## [1,]  1.000000e+00  8.326673e-17 -1.322727e-17  5.039372e-16  1.037365e-15
## [2,]  8.326673e-17  1.000000e+00 -5.620504e-16 -2.498002e-16  8.326673e-17
## [3,] -1.322727e-17 -5.620504e-16  1.000000e+00  4.666406e-16  9.714451e-17
## [4,]  5.039372e-16 -2.498002e-16  4.666406e-16  1.000000e+00  2.775558e-17
## [5,]  1.037365e-15  8.326673e-17  9.714451e-17  2.775558e-17  1.000000e+00
## [6,]  8.812395e-16  9.159340e-16 -1.346145e-15 -2.081668e-16 -5.551115e-16
## [7,]  8.361367e-16 -1.804112e-16  9.194034e-17 -3.191891e-16 -1.387779e-16
## [8,]  3.295975e-16 -1.110223e-16  5.134781e-16 -1.526557e-16 -1.665335e-16
##               [,6]          [,7]          [,8]
## [1,]  8.812395e-16  8.361367e-16  3.295975e-16
## [2,]  9.159340e-16 -1.804112e-16 -1.110223e-16
## [3,] -1.346145e-15  9.194034e-17  5.134781e-16
## [4,] -2.081668e-16 -3.191891e-16 -1.526557e-16
## [5,] -5.551115e-16 -1.387779e-16 -1.665335e-16
## [6,]  1.000000e+00 -8.187895e-16 -8.326673e-16
## [7,] -8.187895e-16  1.000000e+00  1.387779e-16
## [8,] -8.326673e-16  1.387779e-16  1.000000e+00
```

By looking at the product matrix all the diagonal elements are 1 and non-diagonal elements are close to zero (the values are significantly low). Hence we have verified that $U^T * U = I$ Where I is the unit matrix.

Question 3

Answer 3.e:

Since it isn't clearly mentioned on which matrix should we apply SVD on I am applying SVD on both matrix A and Matrix C

For matrix A:

```r
# Calculate the SVD
svd_result <- svd(A)

# Extract the matrices U, D, and V from svd_result
U <- svd_result$u
D <- diag(svd_result$d)  # Create a diagonal matrix from singular values
V <- svd_result$v

# Verify that A = U * D * t(V)
reconstructed_A <- U %*% D %*% t(V)

# Check if the reconstructed_C is close to the original C (within a tolerance)
is_equal <- all.equal(A, reconstructed_A)
```

```r
# Print the result of verification
if (is_equal) {
  cat("SVD verification: Succeeded\n")
} else {
  cat("SVD verification: Failed\n")
}
```

```
## SVD verification: Succeeded
```

For Matrix C:

```r
# Calculate the SVD
svd_result <- svd(C)

# Extract the matrices U, D, and V from svd_result
U <- svd_result$u
D <- diag(svd_result$d)  # Create a diagonal matrix from singular values
V <- svd_result$v

# Verify that C = U * D * t(V)
reconstructed_C <- U %*% D %*% t(V)

# Check if the reconstructed_C is close to the original C (within a tolerance)
is_equal <- all.equal(C, reconstructed_C)

# Print the result of verification
if (is_equal) {
  cat("SVD verification: Succeeded\n")
} else {
  cat("SVD verification: Failed\n")
}
```

```
## SVD verification: Succeeded
```