Assignment 3

Problem 1

Obtain 100 exponential random variables (with rate 1) and arrange them by column in a matrix called M with 10 rows and 10 columns. Use seed 3100.

```
set.seed(3100)
M <- matrix(rexp(100, rate = 1), nrow = 10, ncol = 10)
M
```

```
##             [,1]       [,2]       [,3]      [,4]        [,5]      [,6]
##  [1,] 0.03497781 3.59770206 0.37484718 0.3231024 0.307956601 0.1018388
##  [2,] 0.65841464 0.55292140 0.09646414 1.0362629 0.137434607 1.0467044
##  [3,] 0.10750640 0.42190698 0.31023781 6.5521213 0.003340429 1.3032599
##  [4,] 1.62950324 0.72669854 0.51134436 1.0881958 0.332206606 3.5783108
##  [5,] 0.08794269 1.28418562 0.38409745 0.5002349 0.472627262 2.2685385
##  [6,] 1.32728046 2.43916553 2.97955091 0.9505048 1.074386318 2.1959411
##  [7,] 0.39663445 0.08453929 0.82178693 5.1670862 2.020954758 0.0866258
##  [8,] 1.35320622 2.96892100 0.86430849 0.9533798 2.649080375 0.5828457
##  [9,] 0.19029962 0.30192719 0.74525858 0.1607490 0.916759231 1.4727493
## [10,] 0.01077793 0.35968197 0.75320352 1.0829057 0.455515655 1.4915487
##             [,7]      [,8]      [,9]       [,10]
##  [1,] 0.30443975 0.6095712 2.9428035 2.050638100
##  [2,] 1.19069781 0.6047676 2.4385673 0.085891077
##  [3,] 1.79805133 0.4920903 0.1862567 0.003907331
##  [4,] 0.31659764 0.8666077 1.8679666 0.115830540
##  [5,] 0.05929447 0.7678905 1.6825524 0.350674039
##  [6,] 0.59555819 0.1992251 0.1023265 0.467816124
##  [7,] 0.15875613 0.3380212 0.1918293 1.504048433
##  [8,] 0.40669040 0.2148288 0.2380924 0.045906954
##  [9,] 0.42989111 1.4715676 1.3195414 1.221913577
## [10,] 0.08991129 0.2825832 0.2739411 0.211430582
```

a. Find the transpose of M.

```
M_transpose <- t(M)
M_transpose
```

```
##             [,1]       [,2]        [,3]      [,4]       [,5]      [,6]
##  [1,] 0.03497781 0.65841464 0.107506403 1.6295032 0.08794269 1.3272805
##  [2,] 3.59770206 0.55292140 0.421906978 0.7266985 1.28418562 2.4391655
##  [3,] 0.37484718 0.09646414 0.310237808 0.5113444 0.38409745 2.9795509
##  [4,] 0.32310238 1.03626292 6.552121271 1.0881958 0.50023492 0.9505048
##  [5,] 0.30795660 0.13743461 0.003340429 0.3322066 0.47262726 1.0743863
##  [6,] 0.10183883 1.04670436 1.303259945 3.5783108 2.26853854 2.1959411
```

```
##  [7,] 0.30443975 1.19069781 1.798051327 0.3165976 0.05929447 0.5955582
##  [8,] 0.60957122 0.60476764 0.492090263 0.8666077 0.76789045 0.1992251
##  [9,] 2.94280347 2.43856731 0.186256666 1.8679666 1.68255237 0.1023265
## [10,] 2.05063810 0.08589108 0.003907331 0.1158305 0.35067404 0.4678161
##             [,7]       [,8]      [,9]      [,10]
##  [1,] 0.39663445 1.35320622 0.1902996 0.01077793
##  [2,] 0.08453929 2.96892100 0.3019272 0.35968197
##  [3,] 0.82178693 0.86430849 0.7452586 0.75320352
##  [4,] 5.16708617 0.95337983 0.1607490 1.08290567
##  [5,] 2.02095476 2.64908038 0.9167592 0.45551565
##  [6,] 0.08662580 0.58284568 1.4727493 1.49154868
##  [7,] 0.15875613 0.40669040 0.4298911 0.08991129
##  [8,] 0.33802117 0.21482881 1.4715676 0.28258325
##  [9,] 0.19182934 0.23809239 1.3195414 0.27394110
## [10,] 1.50404843 0.04590695 1.2219136 0.21143058
```

b. Find (M transpose) x M using matrix multiplication.

```r
M_trans_M <- M_transpose %*% M
M_trans_M
```

```
##            [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
##  [1,]  6.895795  9.182225  6.577170  7.858610  6.676228 10.897245  2.995589
##  [2,]  9.182225 30.596110 12.796745 11.956774 13.130693 15.083719  5.654125
##  [3,]  6.577170 12.796745 12.077880 11.840157  8.659062 12.583436  3.616034
##  [4,]  7.858610 11.956774 11.840157 75.252631 15.491622 19.627866 15.428061
##  [5,]  6.676228 13.130693  8.659062 15.491622 13.751607  8.548422  2.869720
##  [6,] 10.897245 15.083719 12.583436 19.627866  8.548422 30.318102  7.213867
##  [7,]  2.995589  5.654125  3.616034 15.428061  2.869720  7.213867  5.585364
##  [8,]  2.924373  6.049212  3.544181  8.058378  3.867363  9.360110  3.028077
##  [9,]  5.708522 17.002030  4.855606  9.396805  5.120621 16.328033  5.605677
## [10,]  1.862695  9.810732  4.711555  9.763996  5.727894  4.813099  1.871415
##           [,8]      [,9]    [,10]
##  [1,] 2.924373  5.708522 1.862695
##  [2,] 6.049212 17.002030 9.810732
##  [3,] 3.544181  4.855606 4.711555
##  [4,] 8.058378  9.396805 9.763996
##  [5,] 3.867363  5.120621 5.727894
##  [6,] 9.360110 16.328033 4.813099
##  [7,] 3.028077  5.605677 1.871415
##  [8,] 4.765603  8.426663 4.142874
##  [9,] 8.426663 22.881866 9.068805
## [10,] 4.142874  9.068805 8.369795
```

c. Find 2.5M + 5.2I10, where I10 is an identity matrix that has 10 rows and 10 columns.

```r
I_10 <- diag(10)
sum <- (2.5 * M) + (5.2 * I_10)
sum
```

```
##             [,1]      [,2]      [,3]       [,4]        [,5]       [,6]
##  [1,] 5.28744454 8.9942552 0.9371179  0.8077559 0.769891503  0.2545971
```

```
##  [2,] 1.64603661 6.5823035 0.2411603  2.5906573 0.343586518  2.6167609
##  [3,] 0.26876601 1.0547674 5.9755945 16.3803032 0.008351071  3.2581499
##  [4,] 4.07375809 1.8167464 1.2783609  7.9204895 0.830516515  8.9457771
##  [5,] 0.21985672 3.2104640 0.9602436  1.2505873 6.381568155  5.6713464
##  [6,] 3.31820115 6.0979138 7.4488773  2.3762621 2.685965795 10.6898528
##  [7,] 0.99158612 0.2113482 2.0544673 12.9177154 5.052386895  0.2165645
##  [8,] 3.38301555 7.4223025 2.1607712  2.3834496 6.622700938  1.4571142
##  [9,] 0.47574904 0.7548180 1.8631465  0.4018725 2.291898078  3.6818732
## [10,] 0.02694483 0.8992049 1.8830088  2.7072642 1.138789137  3.7288717
##            [,7]      [,8]      [,9]       [,10]
##  [1,] 0.7610994 1.5239280 7.3570087 5.126595250
##  [2,] 2.9767445 1.5119191 6.0964183 0.214727693
##  [3,] 4.4951283 1.2302257 0.4656417 0.009768328
##  [4,] 0.7914941 2.1665191 4.6699165 0.289576350
##  [5,] 0.1482362 1.9197261 4.2063809 0.876685098
##  [6,] 1.4888955 0.4980626 0.2558163 1.169540311
##  [7,] 5.5968903 0.8450529 0.4795734 3.760121083
##  [8,] 1.0167260 5.7370720 0.5952310 0.114767386
##  [9,] 1.0747278 3.6789190 8.4988535 3.054783943
## [10,] 0.2247782 0.7064581 0.6848528 5.728576455
```

d. Find the inverse matrix of M.

```r
# Let's first check if M has an inverse by calculating its determinant
det_M <- det(M)
det_M # As long as det_M is non-zero, it will have an inverse
```

```
## [1] 504.3189
```

```r
M_inverse <- solve(M)
M_inverse
```

```
##              [,1]        [,2]         [,3]        [,4]          [,5]       [,6]
##  [1,] -0.06916467 -0.14796225  0.006666865  0.47506873 -0.0005379057  0.2457801
##  [2,]  0.07705159 -0.27716490  0.163415626 -0.05309415  0.4512337664  0.1106673
##  [3,] -0.48209664  0.16683782 -0.078132351 -0.55403413  1.6136150020  0.8701980
##  [4,] -0.16551208 -0.09132973  0.107384933 -0.10087407  0.7868716129  0.2167269
##  [5,]  0.05801302  0.30267535 -0.216849691 -0.04945287 -0.8163290959 -0.4451764
##  [6,]  0.37008348 -0.02125476 -0.044154108  0.41826086 -1.4780812320 -0.5621898
##  [7,]  0.67458397  0.52085618  0.076874626  0.32998786 -3.2468831309 -0.8833387
##  [8,] -0.91819146 -0.63530934  0.419477273 -0.58254064  3.5517028988  1.0490774
##  [9,] -0.20516427  0.44380080 -0.187501246 -0.26949319  0.9906358452  0.2489686
## [10,]  0.90797938 -0.09707495 -0.121723728  0.69978661 -3.0099641305 -0.8347854
##              [,7]        [,8]         [,9]      [,10]
##  [1,]  0.18144103 -0.04401577  0.007891863 -1.3991497
##  [2,] -0.08821803  0.08542185 -0.060365845 -0.6440452
##  [3,]  0.43920755 -0.45208874 -0.092663628 -2.1794330
##  [4,]  0.25718411 -0.14983810 -0.132992014 -1.1173564
##  [5,] -0.06901623  0.41629819  0.053200248  1.7775451
##  [6,] -0.41283299  0.24564561  0.014774709  2.6843807
##  [7,] -0.88302153  0.57481918  0.368308941  4.4313714
##  [8,]  0.52944971 -0.45119575  0.595973043 -5.8497457
##  [9,]  0.34129740 -0.23295348 -0.275768461 -1.0168013
## [10,] -0.45433757  0.26550313  0.298851125  3.8681683
```

e. Find the overall mean of the elements in M.

```
overall_mean <- mean(M)
overall_mean
```

```
## [1] 0.9518744
```

f. Using which.max(), find the row of M with the largest sum.

```
row_sums <- apply(M, 1, sum)
row_w_largest_sum <- which.max(row_sums)
row_w_largest_sum
```

```
## [1] 6
```

g. For each column in M, find the percentage of elements greater than 1.

```
percentage_above_1 <- apply(M, 2, function(x) sum(x > 1) / length(x) * 100)
percentage_above_1 # Since there are 10 entries per column, expect to see a multiple of 10
```

```
##  [1] 30 40 10 50 30 70 20 10 50 30
```

h. Using string functions, give the names "n1", "n2", ..., "n10" to the rows of M, and "v1", "v2", ..., "v10" to the columns of M. Do not type out the name vectors.

```
rownames(M) <- paste0("n", 1:10)
colnames(M) <- paste0("v", 1:10)
rownames(M)
```

```
##  [1] "n1"  "n2"  "n3"  "n4"  "n5"  "n6"  "n7"  "n8"  "n9"  "n10"
```

```
colnames(M)
```

```
##  [1] "v1"  "v2"  "v3"  "v4"  "v5"  "v6"  "v7"  "v8"  "v9"  "v10"
```

i. Extract a sub matrix of M consisting of rows "n3", "n4", and "n5" and all columns except "v2" and "v4".

```
sub_matrix <- M[c("n3", "n4", "n5"), c(1, 3, 5:10)]
sub_matrix
```

```
##            v1        v3          v5       v6         v7        v8        v9
## n3 0.10750640 0.3102378 0.003340429 1.303260 1.79805133 0.4920903 0.1862567
## n4 1.62950324 0.5113444 0.332206606 3.578311 0.31659764 0.8666077 1.8679666
## n5 0.08794269 0.3840975 0.472627262 2.268539 0.05929447 0.7678905 1.6825524
##            v10
## n3 0.003907331
## n4 0.115830540
## n5 0.350674039
```

j. Create a diagonal matrix L that has the same diagonal elements as M.

```r
L <- diag(diag(M))
L
```

```
##             [,1]      [,2]      [,3]     [,4]      [,5]     [,6]      [,7]
##  [1,] 0.03497781 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.0000000
##  [2,] 0.00000000 0.5529214 0.0000000 0.000000 0.0000000 0.000000 0.0000000
##  [3,] 0.00000000 0.0000000 0.3102378 0.000000 0.0000000 0.000000 0.0000000
##  [4,] 0.00000000 0.0000000 0.0000000 1.088196 0.0000000 0.000000 0.0000000
##  [5,] 0.00000000 0.0000000 0.0000000 0.000000 0.4726273 0.000000 0.0000000
##  [6,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 2.195941 0.0000000
##  [7,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.1587561
##  [8,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.0000000
##  [9,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.0000000
## [10,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.0000000
##            [,8]     [,9]     [,10]
##  [1,] 0.0000000 0.000000 0.0000000
##  [2,] 0.0000000 0.000000 0.0000000
##  [3,] 0.0000000 0.000000 0.0000000
##  [4,] 0.0000000 0.000000 0.0000000
##  [5,] 0.0000000 0.000000 0.0000000
##  [6,] 0.0000000 0.000000 0.0000000
##  [7,] 0.0000000 0.000000 0.0000000
##  [8,] 0.2148288 0.000000 0.0000000
##  [9,] 0.0000000 1.319541 0.0000000
## [10,] 0.0000000 0.000000 0.2114306
```

k. Add to L an additional row of all zeros after the last row.

```r
L_add_row <- rbind(L, rep(0, ncol(L))) # ncol() outputs the no. of columns
L_add_row
```

```
##             [,1]      [,2]      [,3]     [,4]      [,5]     [,6]      [,7]
##  [1,] 0.03497781 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.0000000
##  [2,] 0.00000000 0.5529214 0.0000000 0.000000 0.0000000 0.000000 0.0000000
##  [3,] 0.00000000 0.0000000 0.3102378 0.000000 0.0000000 0.000000 0.0000000
##  [4,] 0.00000000 0.0000000 0.0000000 1.088196 0.0000000 0.000000 0.0000000
##  [5,] 0.00000000 0.0000000 0.0000000 0.000000 0.4726273 0.000000 0.0000000
##  [6,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 2.195941 0.0000000
##  [7,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.1587561
##  [8,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.0000000
##  [9,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.0000000
## [10,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.0000000
## [11,] 0.00000000 0.0000000 0.0000000 0.000000 0.0000000 0.000000 0.0000000
##            [,8]     [,9]     [,10]
##  [1,] 0.0000000 0.000000 0.0000000
##  [2,] 0.0000000 0.000000 0.0000000
##  [3,] 0.0000000 0.000000 0.0000000
##  [4,] 0.0000000 0.000000 0.0000000
##  [5,] 0.0000000 0.000000 0.0000000
##  [6,] 0.0000000 0.000000 0.0000000
##  [7,] 0.0000000 0.000000 0.0000000
##  [8,] 0.2148288 0.000000 0.0000000
```

```
##  [9,] 0.0000000 1.319541 0.0000000
## [10,] 0.0000000 0.000000 0.2114306
## [11,] 0.0000000 0.000000 0.0000000
```

l. Extract the last column of M as a 1-column matrix and the last row of M as a 1-row matrix. Form a new matrix H by multiplying the last column of M by the last row using matrix multiplication.

```r
last_column <- M[, ncol(M), drop = FALSE]
last_row <- M[nrow(M), , drop = FALSE]
last_column
```

```
##             v10
## n1  2.050638100
## n2  0.085891077
## n3  0.003907331
## n4  0.115830540
## n5  0.350674039
## n6  0.467816124
## n7  1.504048433
## n8  0.045906954
## n9  1.221913577
## n10 0.211430582
```

```r
last_row
```

```
##             v1       v2        v3       v4        v5       v6         v7
## n10 0.01077793 0.359682 0.7532035 1.082906 0.4555157 1.491549 0.08991129
##            v8        v9       v10
## n10 0.2825832 0.2739411 0.2114306
```

```r
H <- last_column %*% last_row
H
```

```
##               v1          v2          v3          v4          v5          v6
## n1  2.210164e-02 0.737577556 1.544547833 2.220647618 0.934097757 3.058626543
## n2  9.257281e-04 0.030893472 0.064693462 0.093011934 0.039124730 0.128110723
## n3  4.211295e-05 0.001405397 0.002943016 0.004231271 0.001779851 0.005827975
## n4  1.248414e-03 0.041662157 0.087243970 0.125433548 0.052762624 0.172766889
## n5  3.779541e-03 0.126131130 0.264128920 0.379746904 0.159737515 0.523047399
## n6  5.042090e-03 0.168265026 0.352360751 0.506600732 0.213097568 0.697770521
## n7  1.621053e-02 0.540979107 1.132854572 1.628742570 0.685117607 2.243361449
## n8  4.947820e-04 0.016511904 0.034577280 0.049712901 0.020911336 0.068472457
## n9  1.316970e-02 0.439500285 0.920349606 1.323217136 0.556600763 1.822543578
## n10 2.278784e-03 0.076047769 0.159250258 0.228959375 0.096309940 0.315359005
##               v7          v8          v9          v10
## n1  0.1843755101 0.579475975 0.561754061 0.4335676069
## n2  0.0077225773 0.024271380 0.023529096 0.0181600005
## n3  0.0003513132 0.001104146 0.001070379 0.0008261294
## n4  0.0104144729 0.032731770 0.031730746 0.0244901185
## n5  0.0315295541 0.099094609 0.096064033 0.0741432162
## n6  0.0420619497 0.132197000 0.128154065 0.0989106354
## n7  0.1352309299 0.425018891 0.412020685 0.3180018355
```

```
## n8  0.0041275533 0.012972536 0.012575802 0.0097061341
## n9  0.1098638219 0.345292307 0.334732352 0.2583498987
## n10 0.0190099957 0.059746741 0.057919527 0.0447028910
```