Assume that we are in the time series data setting, where we have data at equally-spaced times $1, 2, \ldots$ which we denote by random variables $X_1, X_2, \ldots$. The AR(1) model, commonly used in econometrics, assumes that the correlation between X_i and X_j is $\text{Cor}(X_i, X_j) = \rho^{|i-j|}$, where ρ is some parameter that usually has to be estimated.

If we were writing out the full correlation matrix for n consecutive data points $X_1, \ldots, X_n$, it would look something like this:

$$\begin{pmatrix} 1 & \rho & \rho^2 & \ldots & \rho^{n-1} \\ \rho & 1 & \rho & \ldots & \rho^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \ldots & 1 \end{pmatrix}$$

(***Side note:*** This is an example of a correlation matrix which has Toeplitz structure.)

Given ρ, how can we generate this matrix quickly in R? The function below is my (current) best attempt:

```
ar1_cor <- function(n, rho) {
exponent <- abs(matrix(1:n - 1, nrow = n, ncol = n, byrow = TRUE) -
    (1:n - 1))
rho^exponent
}
```

In the function above, `n` is the number of rows in the desired correlation matrix (which is the same as the number of columns), and `rho` is the ρ parameter. The function makes use of the fact that when subtracting a vector from a matrix, R automatically recycles the vector to have the same number of elements as the matrix, and it does so in a column-wise fashion.

Here is an example of how the function can be used:

```
ar1_cor(4, 0.9)
#       [,1] [,2] [,3]  [,4]
# [1,] 1.000 0.90 0.81 0.729
# [2,] 0.900 1.00 0.90 0.810
# [3,] 0.810 0.90 1.00 0.900
# [4,] 0.729 0.81 0.90 1.000
```

Such a function might be useful when trying to generate data that has such a correlation structure. For example, it could be passed as the `Sigma` parameter for `MASS::mvrnorm()`, which generates samples from a multivariate normal distribution.