MPI Exercises I

1. Write a script that will have each processor randomly take a sample of size 1 of `TRUE` and `FALSE`. Have each processor print its result.
2. Modify the script in Exercise 1 above to determine if any processors sampled `TRUE`. Do the same to determine if all processors sampled `TRUE`. In each case, print the result. Compare to the functions `comm.all()` and `comm.any()`.
3. Generate $50,000,000$ (total) random normal values in parallel on 2, 4, and 8 processors. Time each run.
4.
5. Distribute the matrix `x <- matrix(1:24, nrow=12)` in GBD format across 4 processors and call it `x.spmd`.

MPI Exercises II

1. Add `x.spmd` to itself.
2. Compute the mean of `x.spmd`.
3. Compute the column means of `x.spmd`.

6.

7.

DMAT Exercises I

1. Subsetting, selection, and filtering are basic matrix operations featured in R. The following may look silly, but it is useful for data processing. Let `x.dmat <- ddmatrix(1:30, 10, 3)`. Do the following:

   -
     ```
     y.dmat <- x.dmat[c(1, 5, 4, 3), ]
     y.dmat <- x.dmat[c(10:3, 5, 5), ]
     y.dmat <- x.dmat[1:5, 3:1]
     ```

   -
     ```
     y.dmat <- x.dmat[x.dmat[, 2] > 13, ]
     y.dmat <- x.dmat[x.dmat[, 2] > x.dmat[, 3], ]
     y.dmat <- x.dmat[, x.dmat[2,] > x.dmat[3, ]]
     y.dmat <- x.dmat[c(1, 3, 5), x.dmat[, 2] >
     x.dmat[, 3]]
     ```

DMAT Exercises II

2. The method `crossprod()` is an optimized form of the crossproduct computation `t(x.dmat) %*% x.dmat`. For this exercise, let `x.dmat <- ddmatrix(1:30, nrow=10, ncol=3)`.
    1. Verify that these computations really do produce the same results.
    2. Time each operation. Which is faster?
3. The `prcomp()` method returns rotations for all components. Computationally verify by example that these rotations are orthogonal, i.e., that their crossproduct is the identity matrix.
4.