* lapply
  + lapply takes three arguments:
    1. the list
    2. the function
    3. other arguments
  + lapply will always return a list
  + runif(x) = generates x random uniform numbers
    - you can use min/max in the 3rd lapply field
  + lapply uses anonymous functions
    - you can write the function on the spot
* sapply
  + if every element is length 1, then a vector is returned
  + if the result is a list where every element is a vector of the same length (> 1), a matrix is returned
  + if not, then list
* apply
  + evaluate a function over the rows/columns of an array
  + you can write everything on one line
  + apply has 4 arguments
    1. the list
    2. the margin (1 for rows, 2 for columns)
    3. the function
    4. other
    - rowSums = apply(x, 1, sum)
    - rowMeans = apply(x, 1, mean)
    - colSums = apply(x, 2, sum)
    - colMeans = apply(x, 2, means)
  + quantile = % quantile given probs = c(.25, .75) as an example
  + if something is 3 dimensional (i.e. c(2, 2, 10)) use c(1, 2) to collapse
* mapply
  + this is a multivariate apply:
    1. function (# of args ≥ # of lists)
    2. lists
    3. more args
    4. simplify (i.e. collapse)
  + e.g. mapply(rep, 1:4, 4:1)
  + e.g. noise = function(n, mean, sd) {rnorm(n, mean, sd)}
* tapply
  + apply a function over a subsets of a vector
    1. the vector
    2. index of factors
    3. function to be applied
    4. other args
    5. simplify
* split
  + like tapply without summarizing it in the end
    1. vecror
    2. factor
    3. drop = if empty factors should be dropped
       1. true or false
  + gl(2, 5) = 5x1, 5x2
  + interaction(x, y) = x.y for each x, y
* debugging - diagnosing