1. How to write matrices
   1. nameofmatrix <- matrix(data=, nrow=, ncol=, byrow=T/F)
      1. Matrices fill in data by column first by default (byrow=F)
      2. Fmat <- matrix(1:20, 4, 5)
      3. Tmat <- matrix(1:20, 4, 5, T)
      4. As we learned with multiple argument functions, as long as you specify nrow=, ncol etc, the arguments can be in any order
   2. Dimensions
      1. dim(Tmat)
         1. Shows row, columnO
2. Class type
   1. numvec <- 1:6
      1. class(numvec) returns “integer”, meaning they’re all numbers
   2. wordvec <- letters[1:6]
      1. class(wordvec) returns “character”, which means that everything in that vector is strings
      2. wordmat <- matrix(wordvec, 2, 3)
         1. Contains “a”, “b”, etc through “f”, indicating them as strings
3. Having both
   1. a <- c(1, 2, “a”)
   2. g <- c(4, 6, “g”)
      1. When we run g, we see that everything is in quotes “4”, “6”, “g”
      2. When we run class(g), we get “character”
         1. Whenever there's a string in a vector, everything becomes strings
   3. *We can forcibly change the class of a vector*
      1. *class(g) <- “integer”*
         1. *Makes everything numbers*
   4. *So what would happen if we put these datasets into a matrix?*
      1. *amat <- matrix(a, 1, 3) returns “1”, “2”, “a”*
      2. *gmat <- matrix(g, 1, 3) returns 4, 6, NA*
      3. *If we want the numbers to stay numbers, we can't introduce strings. Even if we try to include strings, they will become NA if we try to keep the numbers integers.*
      4. *So, you can’t have both in a matrix*
4. Dataframes
   1. If you want to have more than one class type in a dataset, we use dataframes.
   2. Nameofdataframe <- data.frame(arg1, arg2, …)
   3. name <- c("Mary", "Dana", "Taylor")
   4. major <- c("Bio", "Math", "Stat")
   5. age <- c(22, 21, 20)
   6. demo <- data.frame(name, major, age)
      1. think of it like a vector; name, major and age are arguments, or individual elements, of data.frame
      2. it's like c() except this time the things we're combining also contain their own elements.
      3. Each vector/argument becomes its own column
5. *Grep*
   1. *grep can be used to find a specific string in a vector (column) to return the row*
      1. *grep("Math", demo[,3])*
      2. *grep("168", demo[,2]) == grep(168, demo[,2])*
6. Graphics
   1. We deal with 2-D visualization, with x and y variables
   2. plot(x variable, y variable)
      1. You can do just one variable plot(dataset), where the x variable is index number
   3. stripchart(y variable ~ x variable)
   4. hist(dataset)