Arithmetic in R

- R can be viewed as a very fancy calculator
- Can perform the ordinary mathematical operations: `+ - * / ^`
- Will perform these on vectors, matrices, arrays as well as on ordinary numbers
- With matrix arithmetic: `*` does elementwise multiplication and `%*%` does standard matrix multiplication.
- `solve(A, b)` gives solution $\mathbf{x}$ to system of equations: $\mathbf{Ax} = \mathbf{b}$.
- `solve(A)` gives inverse for any nonsingular matrix $\mathbf{A}$

More with Matrices

- Functions `nrow`, `ncol`, `dim` tell you size of a matrix object
- `length` gives total number of elements in matrix
- `rbind` (or `cbind`) binds rows (or columns) together to form a matrix.
- Can bind vectors together to make a matrix, or bind small matrices together to form bigger matrices.
- For all these examples, be careful that the dimensions match up correctly!
- Can coerce a numeric data frame or a vector to be a matrix using `as.matrix()`

Logical Objects in R

- Could be vector, matrix, array, etc., whose elements are *all* either TRUE or FALSE
- Usually obtained by comparing:
  1. Two vectors/matrices to each other
  2. A vector or matrix to a constant number or a set of constants
- Possible comparison operators: <, >, <=, >=, ==, !=
- Comparisons are done element by element (see examples)

Working With Logical Objects

- Can combine logical conditions with "and" operator: `&`, "or" operator: `|`, "not" operator: `!` (see examples)
- "Sequential-and" operator `&&`, "Sequential-or" operator `||` can be useful
- Other useful functions for logical objects: `all()`, `any()`, `all.equal()`, `identical()`
- Numerically, TRUE = 1 and FALSE = 0 (can be useful for doing arithmetic on logical objects)
- `identical()` is TRUE only if the objects compared are identical in *every* respect, including mode.
- `all.equal()` more flexible in checking for *approximate* equality
- Logical conditions often used with `if` statements: If condition is TRUE, do something, otherwise do another thing.
- `ifelse` is a concise way to do conditional operations

Subsetting in R

- Basic way to extract subsets of R objects (like vectors, matrices, etc.) is square brackets: `[ ]`
- Values in square brackets tell R which elements you want to pick out
- Minus sign in brackets says to pick out all elements *except* those specified
- With matrices, row and column indices in square brackets separated with a comma.
- Can extract by row/column *name* rather than number, where applicable.
- If the thing in the square brackets is a logical vector, then the elements extracted will be at those positions for which the logical vector is `TRUE`

Sorting in R

- the function `order()` creates a permutation vector for use in sorting vectors
- Gives the positions within vectors of the smallest, . . ., largest elements
- `rev()` useful for sorting in reverse order
- Can sort matrices/data frames by a particular column (or set of columns) using `order` along with square brackets.
- The built-in function `sort()` does basic sorting of vectors.

Iteration in R

- R is not very efficient with loops – avoid them if you can!
- Usually much more efficient to perform large operations on the elements of vectors or matrices
- The `apply` function is useful for doing repetitive tasks
- Example: Can apply R functions iteratively to all the rows (or all the columns, or all the elements) of a matrix
- Have to be careful of what type of input the function being applied accepts!
- The `tapply` function applies an R function *separately for each group* in the data
- Groups often defined based on factor levels
- `by` is similar to `tapply`, `by` works specifically on data frames and matrices
- Similar commands: `sapply()`, `lapply()`, `aggregate()`

Loops in R (as a last resort)

- When you have to use loops, common types are `for` loops and `while` loops.
- With `for` loops, typically a task is done iteratively for each position in a sequence.
- With `while` loops, typically the task is done as long as some logical condition is met.
- Usually more efficient to set up a "dummy" matrix/vector of results and replace its elements as the loop executes than to create new "results rows" as you go.
- The `outer` function is a neat time-saving command for special tasks.

Dates in R

- The `as.Date` function can read in dates given in a variety of formats.
- `format` can change the format of dates in R
- Other useful functions for dates: `Sys.Date()`, `weekdays()`
- See help files: `help(Dates)`, `help(as.Date)`

Graphics in R

- The `plot` function in R is a generic one that does different plots (e.g., scatter plots, etc.) depending on what type of object is input to it.
- Some useful built-in statistical graphics functions: `hist`, `pie`, `stem`, `barplot`
- The `par` command is used to set many different graphical parameters
- You have a great ability to control the "look" of plots in R.
- Easy to add titles to plots or text inside plots
- Can add text and symbols interactively with `locator()` command

More Graphics in R

- Can overlay a curve (or several curves) onto an existing scatter plot with `lines()`
- Can add new points to a scatter plot with `points`
- Can set up an empty plotting window and fill it in
- Can put multiple plots on one page with `mfrow` or `mfcol` options to `par()`
- Usually a new `plot` command will overwrite existing plots, but ...
- The command `windows()` opens up a new separate plotting window, preserving the previous one
- See examples for exploring different colors, different plotting symbols

3-D Plotting in R

- Useful commands for 3-D plots in R: `contour()`, `filled.contour()`, `persp()`, `image()`
- Fancier commands available in the `lattice` package: Type `library(lattice)` to load it
- Example functions include `cloud`, `wireframe`
- The `lattice` package has a lot of advanced graphics functions

Writing Functions in R

- In addition to using R's built-in functions, you can write your own functions
- Several ways to do this:
  1. Write function commands in a text file. Copy and paste it into R, or use `source` command
  2. Use the `fix()` command to create a new function
  3. Can use `fix()` to alter an old function
- When you close it, `fix()` will tell you if there's an error in the syntax of your function
- Use unique names for your personal functions
- If you alter a built-in R function, change its name!

More About Functions in R

- All function statements lie between two braces: `{ }`
- Argument names (and any default values) specified in parentheses
- Remember: A function will return ONLY ONE object (this may be a vector/list with several values)
- Output named in last line before closing brace, or with `return()` command
- Any intermediate objects created will not be saved after the function is executed
- To view the code for a function, type its name with no parentheses
- The `browser()` command is useful for debugging functions