Introduction to Scientific Graphics

Using ‘R’ for Simple Data Analysis
2D Plots
Later this term: Scientific Visualization

Today’s Lecture

- The topic for today: a brief introduction to R
  - general purpose language for data analysis
  - includes functions for 2D plots, histograms, etc
  - goal: sufficient expertise in R to analyze output of Mandelbrot and solar system projects

- Reading:
  - PDFs on-line:
    - Introduction to R
    - The R Programming Language

Background

- R is a public domain implementation of the S language
- S is an extensive commercial software package
  - originally intended for statistical analysis (competes with SPSS)
  - has wide variety of data modeling, matrix algebra, graphics functions
- R project home page: http://www.r-project.org
  - manuals, FAQs, ...
  - download source and precompiled binaries (e.g. for Mac OS/X)
  - CRAN (Comprehensive R Archive Network)
  - major packages (e.g. BioConductor for gene expression data)

Interaction

- R is an interactive programming language
- Use a shell-like interface with a read-eval-print loop
- Example (Mac OS/X):
  - `> help()`
    - pops up a text window with general help information
  - `> help.start()`
    - opens browser window with local copy of HTML documentation
  - `> demo(graphics)`
    - runs a demo showing off various 2D plots and graphics

R prompt
Assignment

- Use a “left arrow” as an assignment operator
  
  ```r
  > a <- 10
  > a
  [1] 10
  > b <- 20
  > a * b
  [1] 200
  > a + b
  [1] 30
  ```

Vector Operations

- The colon operator creates a vector with a specified range of values
  
  ```r
  > a <- 10:19
  > a
  [1] 10 11 12 13 14 15 16 17 18 19
  ```

Vector Functions

- The `c` (concatenate) operator creates a vector filled with the values passed as operands
  
  ```r
  > x <- c(1,2,3,4,0,6,7,8,7,8,7)
  > x
  [1] 1 2 3 4 0 6 7 8 8 7 8 7
  > length(x)
  [1] 12
  > min(x)
  [1] 0
  > mean(x)
  [1] 5.083333
  ```
Dimensions

- Use the `dim` function to assign dimensions to a vector

```r
> a = 1:10
> dim(a) = c(2,5)
> a
[1,]   1   3   5   7   9
[2,]   2   4   6   8  10
```

- Tells R to view `a` as a 2 x 5 array
- Note labels on rows and columns
- Underlying vector is still there...
- ...but now we have a new way to access data

Data Frames

- A convenient data structure in R is the frame
  - “matrix-like structures, in which the columns can be of different types”
- Common use: manage experimental data, using one row per observation
- Example: course grades in a text file

```
Hello nbody quiz final
walrus 100  85   77  95
georgeh 100  98   90  90
rstarr  100 100   80  85
```

Reading a Data Frame

- To read a file and store the data in a frame:

```r
> grades <- read.table("grades.txt")
> grades
   hello nbody quiz final
walrus 100  85   77  95
georgeh 100  98   90  90
rstarr  100 100   80  85
```

- R parses the lines from the file, skipping whitespace
- Note the first line has one fewer item than the remaining lines...

Accessing Frame Elements

- Frames (and all matrices in R) are stored in column-major layout
- Old FORTRAN standard
- Also used in Matlab, BLAS, many other scientific libraries and applications
- A matrix is an array of columns
- The index of the first element is 1, not 0

```r
> grades[1]
   hello
walrus 100
georgeh 100
rstarr  100
```

- R prints row and column labels for columns in frames
Accessing Frame Elements (cont’d)

- Some other operations on frames:

  ```r
  > attributes(grades)
  $names
  [1] "hello" "nbody" "quiz" "final"
  $class
  [1] "data.frame"
  $row.names
  [1] "walrus" "georgeh" "rstarr"
  > names(grades)
  [1] "hello" "nbody" "quiz" "final"
  > row.names(grades)
  [1] "walrus" "georgeh" "rstarr"
  ```

- To access the row and column vectors in a frame:

  ```r
  > grades[1]
  hello
  walrus  100
  georgeh  100
  rstarr  100
  > grades[,1]
  [1] 100 100 100
  > grades[1,]
  hello nbody quiz final
  walrus   100    85   77    95
  ```

Operations on Frames

- Many built-in functions work equally well on frames and matrices

  ```r
  > min(grades)
  [1] 77
  > min(grades[,2])
  [1] 85
  > min(grades[2,])
  [1] 90
  ```

  Lowest grade in frame
  Lowest grade on nbody project (a vector of scores)
  Lowest grade for georgeh (a frame)

- To access the row and column vectors in a frame:

  ```r
  > grades[,1]
  hello
  walrus  100
  georgeh  100
  rstarr  100
  > grades[1,]
  hello nbody quiz final
  walrus   100    85   77    95
  ```

- Read on your own: high-level operations on matrices and frames

  ```r
  > apply(grades,1,sum)
  walrus georgeh rstarr
  357 378 365
  > apply(grades,2,sum)
  hello nbody quiz final
  300 283 247 270
  ```

  Dimension 1 = rows; i.e. compute sum of grades in each row
  Dimension 2 = cols
Plots

- Use R (or Matlab or Mathematica or ...) to verify your code for a project is working
- Example (N-Body project): A simple dot plot of the orbits of the planets in the solar system is a good check
- Example of a dot plot in R:
  ```r
  > data
  [1] 1 2 3 4 0 6 7 8 8 7 8 7
  > x = 1:length(data)
  > plot(x, data)
  ```

Plot Output

- The plot command creates a graphic window (or opens an existing window)
- “High level” commands like plot create a new graphic
- “Low level” commands add information to an existing graphic

Plot Options

- Use additional arguments to specify options
  ```r
  > plot(x, data, pch="*")
  use asterisk as the plot character
  > plot(x, data, pch=4)
  use the 4th predefined plot character
  > plot(x, data, col=6)
  use color number 6
  > plot(x, data, xlab="pretty purple circles", col=6)
  use this string as the x-axis label
  ```

Adding Items to An Existing Plot

- Some examples of “low level” commands to overlay a new information on an existing plot:
  ```r
  > points(x, y, opts)
  adds a new set of points; arguments are the same as those for plot
  > text(x, y, "string")
  plot the string at the specified location
  ```
Using R to Make Images

- For the Mandelbrot project the goal is to draw a picture of the set
- Your program can print integer values for each pixel
  - values range from 0 to 255
  - print in any format, but suggest n rows of n numbers separated by spaces
- Example: run the program to show the upper right part of the image:
  
  % mandelbrot 0 0 .01 100 100 > ur.txt

Read the File, Make a 2D Matrix

```r
> m = scan("ur.txt")
Read 10000 items
```

- Tell R to reshape m as a 100 x 100 matrix (since you know from the command line that made the matrix that nx and ny are 100):
  
  ```r
  > dim(m) = c(100,100)
  > dim(m)
  [1] 100 100
  ```

Use image to Make the Drawing

- The `image` function draws the picture:
  
  ```r
  > pdf("mandel.ur.pdf")
  > image(m)
  > dev.off()
  ```

- Ick -- ugly colors, wrong proportion

Setting Parameters

- There are two ways to set drawing parameters
  - per-drawing: add extra named parameters when you call the function
  - use the `par()` function to make changes that will apply to all subsequent commands
- It takes a bit of practice and experimentation to learn which parameters can be set with `par()` or supplied with the function or both
  - (or, if all else fails, read the manual)
Second Attempt

- Call `par()` to say you want a square plot (equal scale for x and y dimensions)
  ```r
  > par(pty="s")
  ```
- Type `help(par)` to see what other attributes can be set
- Use a different color map:
  ```r
  > image(m,col=terrain.colors(256))
  ```

R and the N-Body Project

- The program you write for the N-Body project will calculate positions for a set of bodies
- The output will be a record of the positions at different time steps
- To debug the program, use test data from the solar system
  - initial configuration: 9 planets, all arranged on the Y axis and moving to the right
  - expected output: clockwise circular orbits
- Use R to plot the orbits, verify the bodies are moving in a circle

R and the N-Body Project (cont'd)

- Put your output data in a table that can be read into a frame
  - One row per time step
  - One column per coordinate (e.g. sunx, suny, mercx, ...)
  ```r
  > orbits <- read.table("orbits.txt")
  ```
- Make an initial empty plot to create a frame big enough to hold the largest orbit
  ```r
  > par(pty="s")
  > m = max(orbits)
  > plot(c(-m,m),c(-m,m),xlab="X",ylab="Y",
        main="Solar System",pch=" ")
  ```

R and the N-Body Project (cont'd)

- Use the `points` command to add each orbit
  ```r
  > points(orbits$mercx,orbits$mercy,col=2)
  ```
- To save the plot in a PDF file:
  ```r
  > pdf("Desktop/merc.pdf")
  > par(pty="s")
  > plot(c(-m,m),c(-m,m),xlab="X",ylab="Y",
       main="Solar System",pch=" ")
  > points(orbits$mercx,orbits$mercy,col=2)
  > dev.off()
  ```
Demo

- Let's watch the R graphics demo
  
  ```R
  > demo(graphics)
  ```