Download Example Code

https://github.com/MorrellLAB/Applied-Bioinformatics-Discussion

Follow Along With the Presentation

http://z.umn.edu/rbds

What's a Pirate's Favorite Programming Language?

*eve*Rything you need to know

Chaochih Liu and Paul Hoffman

applied bioinfoRmatics discussion

July 15th, 2016

how R thinks

Interpreted:

- Do not need to compile source code before running program
- Can enter an 'interactive' mode for R

Object-oriented:

- Can tell the computer what kind of data we're working with
- Can create class-specific functions, or methods, that are specialized for the data we have

Functional:

- R works best when everything is a function
- Functions can be mapped, or applied, over large quantities of data
    - This enables vectorization of functions to increase speed

functional vs impeRative

Functional

```
> numbers <- c(1,2,3,4,5) # Create an array of numbers
> doubled <- sapply(X = numbers, FUN = function(x) {return(x*2)}) # Tell the computer to double all numbers
> doubled
[1]  2  4  6  8 10
```

Imperative

```
>>> numbers = [1, 2, 3, 4, 5] # Create a list of numbers
>>> doubled = [] # Create an empty list for doubled numbers
>>> for i in numbers: # For each number
...     newNumber = i * 2 # Create a new number equal to the current number times two
...     doubled.append(newNumber) # Add this to the list of doubled numbers
...
>>> doubled # Print our list to the screen
```

objects in R

- Computers don't inherently know what data is or how to use it
- **Classes** tell the computer what kind of data we have and what we're allowed to use it for
- We use these data by calling functions for each specific class to manipulate data and perform analyses

objects in R

- R is very good at handling complex data structures
- R has no object for single-values, everything is a collection
- R's base collection is a vector, a single number is a vector of length 1
- R treats functions as first-class objects

(some of) R's common classes

The following are classes that vectors can take

Type	What?
character	Characters or strings, literally whatever you put in: "Hello world"
logical	Boolean values: TRUE, FALSE, and NA
numeric	Numbers, both whole integers and floating point values: 3, 12.9
factor	~~The bane of my existence~~ A variable that can take on a limited set of values

(some of) R's common classes

Type	What?
vector	Simplest collection of other types
list	A collection of vectors that can be called by name
matrix	A collection of vectors in multiple dimensions
data.frame	A collection of vectors in multiple dimensions that can be called by name
null	Empty value, NULL
generic	A function that is not designed to operate on a specific class. Functions have methods attached for specific classes: is(), is.vector(), is.matrix()

(some of) R's common classes

```
> "Hello world" # Character
[1] "Hello world"
> TRUE # Logical
[1] TRUE
> 5 # Numeric
[1] 5
> factor(c(1, 4, 6, 1))
[1] 1 4 6 1
Levels: 1 4 6
```

(some of) R's common classes

```
> c(1, 4, 5) # Vector
[1] 1 4 5
> list(nums= c(1, 4, 6), logs = c(TRUE, FALSE)) # List
$nums
[1] 1 4 6

$logs
[1]  TRUE FALSE
> matrix(data = c(1, 5, 2, 6, 3, 6 , 2, 5 ,2), ncol = 3, byrow = TRUE) # Matrix
     [,1] [,2] [,3]
[1,]    1    5    2
[2,]    6    3    6
[3,]    2    5    2
> data.frame(nums = c(1, 4, 5), logs = c(TRUE, TRUE, FALSE)) # Data Frame
  nums  logs
1    1  TRUE
2    4  TRUE
3    5 FALSE
```

slicing and dicing

Depending on the class of data we have, we create subsets in different ways

```
> c(1, 5, 2, 6, 9, 3)[3:5] # Slicing a vector
[1] 2 6 9
> matrix(data = c(1, 5, 2, 6, 3, 6 , 2, 5 ,2), ncol = 3, byrow = TRUE)[3:5] # Slicing a matrix
[1] 2 5 3
> list(nums= c(1, 4, 6), logs = c(TRUE, FALSE))$logs
[1] TRUE FALSE
> data.frame(nums = c(1, 4, 5), logs = c(TRUE, TRUE, FALSE))$nums
[1] 1 4 5
> data.frame(nums = c(1, 4, 5), logs = c(TRUE, TRUE, FALSE))[3, 'nums']
[1] 5
```

De Chaochih (she got mad at me last time...)

getting help

Help for any generic function and their methods can be accessed with a ?

```
> ?read.table
```

Help for objects doesn't exist, but object structure can be found using str()

```
> str(object = data.frame(nums = c(1, 4, 5), logs = c(TRUE, TRUE, FALSE)))
'data.frame':   3 obs. of  2 variables:
 $ nums: num  1 4 5
 $ logs: logi  TRUE TRUE FALSE
```

functions in R

Functions are their own class in R

- Base functions are called generics
- generics have methods associated with them for other classes
  - as()
  - as.character()
  - as.logical()
- Functions can be passed to other functions

useR-defined functions

```
myFunction <- function(x, y = 2) { # Create a function and assign it to myFunction
    # x does not have a default set
    # y defaults to 2, but can be set otherwise
    z <- x * y # Create a variable called z that's the product of x and y
    return(z) # Return z
}
```

vectoRization

Vectorization is the act of applying a function across vector elements

```
> x <- c(1, 2, 3, 4, 5) # Create an array of numbers
> y <- c(9, 8, 7, 6, 5) # Create another array of numbers
> x * y
[1] 9 16 21 24 25
```

the apply family

The apply() functions are designed to vectorize pre-built or user-defined functions

For-loops in R are slow and cumbersome, the apply() functions are quick and vectorized

Function	Use
apply()	Transform a complex data structure, ie. matrix or data frame; works in multiple dimensions
lapply()	Works with list-structured data
sapply()	Works with vectors and matrices
vapply()	Same as sapply() but has a pre-structured return value, ie. give it a vector to put values in to
mapply()	Multivariate version of sapply()

the apply family

```
> # Double all numbers in a vector
> sapply(X = c(1, 2, 3, 4, 5), FUN = function(x) {return(x*2)})
[1] 2 4 6 8 10
> # Sum the two columns in a data frame on a row-by-row basis
> apply(X = data.frame(V1 = c(1, 4, 5, 8), V2 = c(19, 43, 12, 16)), MARGIN = 1, FUN = sum)
[1] 20 47 17 24
```

wRiting scRipts

A script is a series of commands to perform some kind of analysis, exploration, or transformation

Uses for scripts:

- Take notes of what was done
- Quickly replicate analysis
- Write a program that accepts command-line arguments and performs analysis or transformation on various datasets

tips for scRipting R

- Write functions
- Design functions to work on subsets of data
- Pay attention to classes
- *Don't* use R for text processing
- **Do** use R for data transformation, exploration, and analyses
- Avoid for-loops, vectorize instead

gRaphing (aka *exe*Rcise)!

gRaphing cheat sheets!

Cheat Sheet for Plotting Symbols and Color Palettes

R color cheatsheet

Thetas File Layout

Column	Contents
Chr	Name of contig
WinCenter	Center of window for Thetas Analysis
tW	Raw Watterson's Theta value
tP	Raw Pairwise Theta value
tF	Raw Fu and Li's Theta value
tH	Raw Fay's Theta value
tL	Raw Maximum Likelihood Theta value
Tajima	Tajima's D
fuf	Fu and Li's F
fud	Fu and Li's D
fayh	Fay's H
zeng	Zeng's E
nSites	Effective number of sites

To plot a Theta's value, you need to divide the Theta's column by nSites