INTRODUCTION

Why choose R over some other language or software application for bioinformatics?

• Large collection of algorithms available
• High quality numerical routines
• Integrated data visualization tools
• Reproducibility*
• Ability to integrate text and coding
• Ability to interoperate with other languages
• Free, worldwide distribution & development
• Resource rich

Q: Where do I get R?
A: WWW.R-PROJECT.ORG

R version 2.15.2 released 26-OCT-2012

Q: What systems will it run on?
A: UNIX, Windows, MacOS

Q: What parts do I download?
A: Just base system is enough to start (there are hundreds of “packages” available to install for implementation later—460 for bioinformatics alone!).
The R GUI is rather austere and not very inviting for novices.

The R environment for WIN, MAC, & UNIX is also slightly different (MAC shown here).

Let's begin by seeing what R can do.

```
> X = 10
> Y = 5
> X+Y
[1] 15
```

NB: By convention, I will place all R code in a gray box to differentiate from comments or instruction.

What are some of the important elements here?

- `> ` symbol is the R prompt (looking for input)
- R works by assignment (use = or concatenate `<-`)
- \textbf{NOTE: R is case sensitive!} (X is different than x)
- Results are returned as `[ ]`

```
R LANGUAGE FUNDAMENTALS

\begin{itemize}
  \item \texttt{pi}
  \item \texttt{log(14)}
  \item \texttt{exp(2.1)}
  \item \texttt{1/9}
  \item \texttt{12^2}
  \item \texttt{12^2/(3-1)}
\end{itemize}

In its simplest form, R can be used just like a calculator.
```
R LANGUAGE FUNDAMENTALS

> seed <- c(1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4)
> seed
[1] 1.2 1.4 1.6 1.8 2.0 2.2 2.4
> seed[5]
[1] 2
> seed^2
[1] 1.44 1.96 2.56 3.24 4.00 4.84 5.76

• Can assign a value, a vector, or a matrix to a variable name
• Note use of "<-" symbol here (no space between)
• Vector with \( n = 6 \) elements
• Can identify specific elements in vector (e.g., 5th obs)
• Can do vector arithmetic (e.g., square all elements)

R LANGUAGE FUNDAMENTALS

> mean(seed)
[1] 1.8
> length(seed)
[1] 7
> median(seed)
[1] 1.8
> sd(seed)
[1] 0.4320494

A large number of preprogrammed statistical functions are available in R.
Note: Function "length" (of vector) is same as sample size \( (N) \).

R LANGUAGE FUNDAMENTALS

> mySE <- function(x)
+   {v = var(x)
+     n = length(x)
+     return((v/n)^(0.5))
+   }
> mySE(seed)
[1] 0.1632993

• User defined functions are also easy to create using "function" and braces ( ) notation.
• Here, I create a function to calculate the standard error, defined as \( \text{mySE} \), which is not supported by R.
• Note that the + sign occurs when R not to end of line.
• Graphs are simple and easy to execute
• Shown here is a simple histogram
• Program defaults are used, but all features are addressable and customizable
For example...

> hist(seed)

Histogram of seed

0.0
0.5
1.0
1.5
2.0
2.5
3.0

• Graphs are simple and easy to execute
• Shown here is a simple histogram
• Program defaults are used, but all features are addressable and customizable
For example...

> hist(seed, col="blue", cex.lab=1.25, col.axis="red", cex.main=1.25)

dev.print(pdf,'histblue.pdf')

> summary(seed)

Min. 1st Qu.  Median   Mean 3rd Qu. Max.
1.2    1.5    1.8    1.8  2.1  2.4

There are a very large number of graphical options and data summary procedures.
R LANGUAGE FUNDAMENTALS

Housekeeping Issues:

Q: I am getting confused, what objects have I already defined in my work session?
A: Use the code: `objects()`

Q: I have been using `dev.print` to save my graphs as files; where are they going?
A: Use the code: `getwd()`

Q: How do I change my working directory?
A: Use the code: `setwd()`

Q: How do I quit the program?
A: Use the code: `q()` or `quit()`

R LANGUAGE FUNDAMENTALS

Search the entire internet for R-related topics

What if you knew you wanted to make a histogram, but did not know what the function was to do so?

> `RSiteSearch("histogram")`

A search query has been submitted to http://search.r-project.org
The results page should open in your browser shortly

R LANGUAGE FUNDAMENTALS

Search R base system and all packages

> `help.search("histogram")`
R LANGUAGE FUNDAMENTALS

Get help on specific topic:

> help(hist)

R LANGUAGE FUNDAMENTALS

Another nice feature of R is that it not only provides help at various levels (global, local, specific), it also provides built-in examples:

> example(hist)

R LANGUAGE FUNDAMENTALS

• Sequence generation, vector subsetting, & number generation from a distribution is easy.
• Vector subsetting is accomplished using brackets [ ].
R LANGUAGE FUNDAMENTALS

R has dozens of preprogrammed continuous and discrete probability distributions:

- Beta
- Binomial
- Chi-square
- Gamma
- F
- Hypergeometric
- Logistic
- Lognormal
- Normal
- Poisson
- T
- Tukey
- Weibull

Here we show an example of how to make your own sequences or distributions and use the commenting symbol in R. Let’s make the famous Fibonacci sequence.

```r
> ## Set up a numeric vector of length 12
> Fibonacci <- numeric(12)
> ## Following line updates the first two elements to value of 1
> ## Update subsequent elements according to rule
> ## Loop for 3rd to 12th element
> for (i in 3:12) Fibonacci[i] <- Fibonacci[i-2] + Fibonacci[i-1]
> ## See all 12 values created
> Fibonacci
[1] 1 1 2 3 5 8 13 21 34 55 89 144
```

R LANGUAGE FUNDAMENTALS

R has another appealing feature and that is the use of SCRIPTS. A script is simply a file with an R extension (*.R) that preserves source code. The files can be saved and run via command line whenever needed. They are easily shared, edited, and updated.
So far, we have considered simple data structures; i.e., assigned values, sequences, and vectors. Data can also exist as matrices, arrays, or dataframes. Let’s consider matrices and dataframes specifically, as these are the two most common forms.

Matrices are two dimensional tables of values (rows and columns) much like a spreadsheet.

Data are read into matrices in a columnwise fashion. Let’s do an example...

```R
> m <- matrix(c(38,14,11,51), nrow=2, ncol=2)
> m[,1]
[1] 38 11
> m[,2]
[1] 14 51
> m[1,2]
[1] 11
> m[1,]
[1] 38 11
> m[2,]
[1] 11 51
```

Note columnwise reading of matrix by R!

More commonly, data come in to R as a “dataframe”. Dataframes are very similar to matrices in that they show up as rows and columns, but they differ by having column headings.

Typically, columns are often different variables. Rows usually represent observations. Data usually originate from a spreadsheet..

R LANGUAGE FUNDAMENTALS

R does NOT read Excel spreadsheet files directly!
(R can not read *.XLS or *.XLSX)

However, it is still easy to read spreadsheet files into R,
you just need to pick a format that it can read. ASCII files
are best, so save as *.TXT or better *.CSV and read that
way. An example:

```r
> getwd()
[1] "/Users/brian"
> trees<-read.csv("trees.csv", header=TRUE)
> trees
Diameter Height
1 10 24
2 12 32
3 16 40
4 18 48
5 20 52
```

In order to read a *.TXT
you would need to use the
read.table function
(instead of read.csv).

I do not recommend this
as it is more error prone.
Always proof your data!

Web resources for R are
excellent.
Quick-R is a particularly
good site to get you up
and running "quickly".

www.statemethods.net
R LANGUAGE FUNDAMENTALS

One of the big advantages of R is that there is a BASE program and now over 1000 specialty PACKAGES that exists as add-ons.

The value of the package approach is that specialists and users can all contribute to each discrete area in the sciences.

Bioinformatics has been very active in creating packages for different applications.

What packages are out there? Where are the packages located? How do you get them? How do you install them?
R FUNDAMENTALS: WHAT HAVE WE LEARNED?

1. How to get and install R
2. How to locate resources for use
3. How to create variables and assign data
4. Assignment, sequences, vectors, matrices
5. Do basic arithmetic (use functions)
6. Perform simple statistical descriptions (use functions)
7. Create user defined functions
8. Create basic plots to summarize & interpret data
9. Get help on using R
10. Manage data
11. Creating and using R script files
12. Creating and importing dataframes
13. Installing and using packages

NEXT STEPS

One of R’s drawbacks is its very simple user interface.

There have been a number of options available for remediying this issue over the years, the most recent one looks very promising:

**RSTUDIO**: Rstudio is an integrated development environment (IDE) for R, and contains a more intuitive user interface, with powerful coding tools, and enhanced productivity.

Available for free: [www.rstudio.org](http://www.rstudio.org)