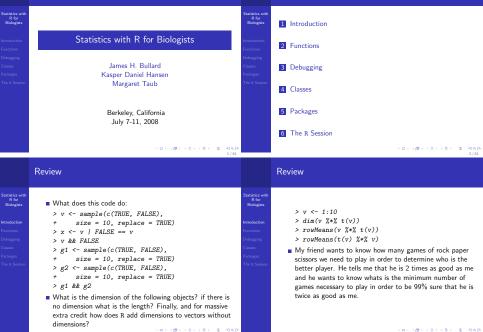
R Programming



Background

Statistics with R for Biologists

- Introduction Functions Debugging Classes Packages The R Section
- The R language provides a mix of features to support functional, procedural, and object oriented programming.
- R is both an environment for statistical computing as well as general purpose programming language.
- R has first-class functions, general data structures, international support, matrix operations, and can be extended via C and other languages.
- R does not have threads, has two systems of classes, but none with explicit syntactic support, R is untyped.
- R has built in support for statistical models incluiding a reasonably complicated formula language.

(D) (B) (2) (2) (2) 2 900

101100101010100000

Functions: Introduction

Statistics wit R for Biologists

Functions

Biologists

- R is "mostly" a functional language, so we really want to understand functions - they are really the fundamental unit of code (this is true whether you call them or write them).
- In R functions are "first class" objects this is demonstrated by how they are defined '<-'. What does this mean?</p>
- R functions are more formally known as 'closures'.
- The last expression of a function is the default return value. Alternatively, we can return from functions using the return function.

0.00 \$ (\$) (\$) (0) (0)

	Functions: Arguments
atistics with R for Biologists roduction nctions bugging isses claages e R Session	<pre>> x <- 2 > fxy <- function(x, y = rep(1, + length(x))) { + return(x'y) + } > fxy(y = seq(2, 16, by = 2), + x = rep(2, 8)) > fxy(rep(2, 8), seq(2, 16, by = 2)) > fxy(rep(2, 8))</pre>

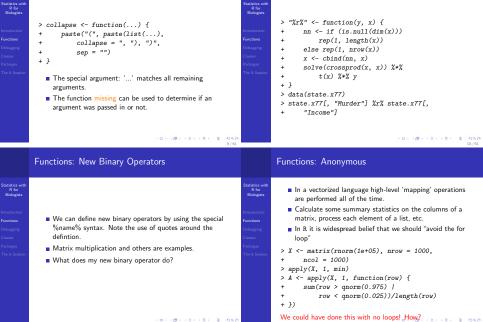
- All arguments to a function are "keyword" arguments
- R has "partial matching" which is something you should try to avoid depending on.

Functions: Arguments

- R has lazy evaluation, what is the length of y when the function is called? (A million points: write a piece of code to demonstrate lazy evaluation)
- R has call by value semantics which means that whatever arguments you pass to a function are passed by value - if you modify objects referred to by parameters then copies will be made.

Functions: A	rguments
--------------	----------

Functions: New Binary Operators



101 (0) (2) (2) (2) 2 (0)

12/61

Functions: Objects

Functions: Scoping

Statistics with R for Biologists Antroduction Functions Debugging Classes Packages The R Session	It is important to be familiar with functions as objects, i.e. we can pass them as arguments, store them in lists, and do much more! Isingly, why do we call functions closures? < < - list(f1 = function(x) { < tmp (= quantile(x, probes = seq(0, + 1, length = 11)) + mean(x[x > tmp[2] & x < + tmp[10]]) + (x - mean(x))/sd(x) +); > dta <- rnorm(1000) > a[[1]](dta)	Statistics with B for Introduction Functions Debugging Classes Packages The R Session	 R is a lexically scoped languge like Python or Scheme. Quiz: What does this program print? a <- 2 setA <- function() { a <- 4 pint(a) pint(a) What about this one:
	Functions: Scoping		Functions: Scoping
Statistics with R for Biologists Introduction Functions Debugging Classes Packages The R Session	<pre>> a <- 2 > setA <- function() { + a <<- 3 + print(a) + } > setA() > print(a) The <<- is syntactic sugar for recursing up the "environment" looking for a binding for "a" and then setting it. We can do this manually by:</pre>	Statistics with R for Biologiss Introduction Functions Debugging Classes Packages The 8 Session	<pre>> a <- 2 > setA <- function() { thisEnv <- parent.env(new.env()) + parentEnv <- parent.env(thisEnv) + assign("a", 4, parentEnv) + print(a) + } > setA() > print(a) In a functional language we want to return values from functions. We call functions for the return values, not their side effects</pre>
	<ロ> (費) くき) くき) くき) ま つの(* 15/61		<ロ> <ラ> < き> < き> < き> < き> まま - つの(や 16/61

Functions: Scoping

Statistics with R for Biologists

- Introduction Functions Debugging Classes Packages The R Section
- This is pretty esoteric, but we can understand how R (and for that matter a number of lexically scoped programming languages) work. When we make a new function we are creating a new "environment" in this environment "a" has no binding, when we ask for the value of "a" we look it up in our environment and if we don't find it we follow the link from the current environment to the parent environment.
- The special environment refered to as the "workspace" is called the .GlobalEnv, we can query it, assign to it, and delete from it.
- Hard Question: How do we assign a variable to the .GlobalEnv

1 assign, get, ls, rm, mget

Hashtables

Statistics with R for Biologists

Functions

Biologists

Environments can be used as hashtables and often are. Environments don't behave like normal R objects. First, lets look at how a typical R object behaves. What does the following code print and why?

Now what happens in the case of environments?

```
(D) (D) (2) (2) (2) 2 (0)(0
18/61
```

20/61

Hashtables

Statistics with R for Biologists

Introduction Functions Debugging Classes Packages The R Session

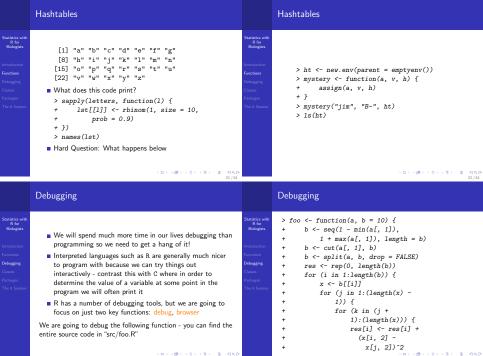
```
> a <- new.env(parent = emptyenv())
> assign("jim", 2, a)
> b <- a
> ls(b)
[1] "jim"
> assign("joe", 3, b)
> ls(a)
```

```
[1] "jim" "joe"
```

Environments don't follow the pass by values semantics or the copy on assignment rule. The environment then can be used if you want a datastructure to pass around but you never want R to copy it (think pointer)

Hashtables

- Bioconductor tends to use environments a lot and in this context it is generally fine to just think about them as hashtables.
- We generally want to use lists to accomplish the same purposes as environments, but we need to know that these things exist in case we want to use them. To construct a hashtable we do the following:



24/61

	Debugging		Browser
Statistics with R for Biologiss Introduction Functions Debugging Classes Packages The R Session	<pre>+</pre>	Statistics with R for Biologists Introduction Functions Debugging Classes Packages The R Session	 Another useful function is: browser. This function can be inserted as a means to debug ode that is misbehaving. browser is essentially breakpoint in other languages. X <- matrix(runif(10000), 10, (10, (10, (10, (10, (10, (10, (10,
	25/st traceback		OOP
Statistics with R for Biologists Introduction Functions Debugging Classes Packages The R Session	 When we commit an error we can look at the call stack by calling traceback. We will see an example below 	Statistics with R for Biologiss Introduction Functions Debugging Classe Pokchages The R Session	 Object oriented programming is a programming paradigm which has become very popular in recent years. Object oriented programming allows us to construct modular pieces of code which can be utilized as building blocks for large systems. R is not a particularly object oriented system, but support exists for programming in an object oriented style. The Bioconductor project has pushed this style and we will need to get familiar with the object system in R in order to work effectively with Bioconductor. Unfortunately R has two class systems known as S3 and S4. These two systems are quite different and don't play well together.
			weir together.

OOP

Statistics with R for Biologists

Introduction Functions Debugging Classes Packages

In both R systems the object oriented system is much more method-centric than languages like Java and Python - R's system is very Lisp-like.

S3 Classes

Statistics with R for Biologists

First we will take a look at S3 classes as they are quite prevalent in day-to-day R programming and need to be introduced in order to get a handle on some of the tricky corners of R.

- An S3 class is constructed via the following code: class(obj) <- "class.name"</p>
- Essentially, a class in this setting is nothing more than an attribute that is used by special functions to perform methods dispatch.

- (ロ) (書) (注) (注) 注 のQで 30/61

S3 Classes

Statistics with R for Biologists

Introduction Functions Debugging Classes Packages

"The greatest use of object oriented programming in R is through print methods, summary methods and plot methods. These methods allow us to have one generic function call, plot say, that dispatches on the type of its argument and calls a plotting function that is specific to the data supplied." – R Manual

> cdf <- ecdf(rnorm(1000))

- > class(cdf)
- > plot(cdf)
- > plot(rnorm(1000))
- > print

S3 Classes

Statistics with R for Biologists

lasses

An S3 method or generic is a method like print which when called dispatches on the class attribute of its first argument. If there is no class argument or if there is no matching function for the class then we call xxx.default.

- > vec <- rnorm(100)
 > class(vec)
 > getS3method("plot", "numeric")
 > class(vec) <- "density"</pre>
- > plot(vec)
- > traceback()

100 (B) (2) (2) 2) 2 900

	S3 Classes		S3 Classes
Statistics with R for Biologiss Introduction Functions Debugging Classes Packages The R Session	<pre>> jim <- list(height = 2.54 * +</pre>	Statistics with R for Biologiss Introduction Functions Debugging Classes Packages The R Session	<pre>\$height [1] 1.8288 \$weight [1] 81.81818 \$name [1] "James" attr(,"class") [1] "person"</pre>
	S3 Classes		S3 Classes
Statistics with R for Biologies Introduction Functions Debugging Classes Packages The R Session	<pre>> print.person <- function(x, +) { + cat("name:", x\$name, "\n") + cat("height:", x\$height, + "meters", "\n") + cat("veight:", x\$weight, + "kilograms", "\n") + } > print(jim) name: James height: 1.8288 meters weight: 81.81818 kilograms</pre>	Statistics with R for Biologies Introduction Functions Debugging Classes Packages The R Session	Warning: There is an error below in two places, what is it? What is the object, and what is the class? An S3 class is really just a list with an attribute class associated with it. In order to define a specialized method for our class (such as plot, or summary) we just define a new function with xxx.jim and then when xxx is called on an object with class(object) == "jim" then R calls our method automatically. This is very common and thus leads an R programmer to often just call plot or summary on about anything – generally with sensible results.

Useful S3 Method Functions

Statistics with R for Biologists

Introduction Functions Debugging **Classes** Packages The R Session getS3method("print", "person") : Gets the appropriate method associated with a class, useful to see how a method is implemented. Try: getS3method("residuals", "lm")

In emacs using ESS we can often use tab to determine what methods are available under a certain generic. Try typing "plot." and then hitting tab - hopefully we will see a list of possible completions. This can be quite useful for getting help on the specific method (we will see more of this later)

getAnywhere : getAnywhere("lm")

4 methods : methods("print")

> getS3method("residuals.HoltWinters")

> getAnywhere("residuals.HoltWinters")

S4 Classes

itatistics with

Biologists

Biologis

lasses

Although S3 classes can be quite useful and powerful they do not facilitate the type of modularization and type safety that a true object oriented system intends.

- For this reason S4 classes were introduced. S4 classes are much more of an object oriented system with type checking, multiple-dispatch, and inheritance.
- Again, here we want to forget about the classes and center our attention on the methods.
- In the resources directory you'll find a document describing S4 classes – this document is highly recommended reading if you want to get a deeper understanding of the S4 system.

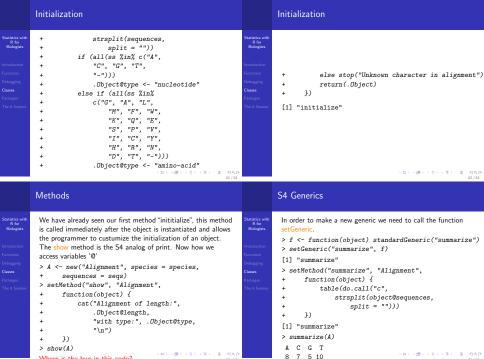
(D) (B) (2) (2) 2 900

S4 declaring a class

Statistics with R for Biologists Introduction Functions Debugging Classes Packages The R Session

```
Lets say we want to construct a class representation for
alignments. What does an alignment contain? At a minimum
we need the names of the species in the alignment, the length
of the alignment, the sequences themselves, and whether we
are dealing with nucleotide data or amino acid data.
> repr <- representation(species = "character".</p>
       sequences = "character".
       length = "integer", type = "character")
> setClass("Alignment", repr)
[1] "Alignment"
> A <- new("Alignment")
> species <- c("tiger", "lion",
       "bear")
> seqs <- sapply(1:length(species),</pre>
       function(i) paste(sample(c("A",
                                          121121-2
           "C" "G" "T") size = 10
```

	Initialization
vith	
s	> setMethod("initialize", "Alignment",
	+ function(.Object, species,
n	+ sequences) {
	+ .Object@species <- species
	+ .Object@sequences <- sequences
	+ names(.Object@species) <- NULL
	+ names(.Object@sequences) <- NULL
ion	+ if (length(sequences) !=
	+ length(species)) {
	+ stop("length(sequences) != length(speci
	+ }
	+ .Object@length <- nchar(sequences[1])
	+ names(.Object@length) <- NULL
	+ ss <- do.call("c",
	<ロ><(型)><(置)><(置)><(差)><(差)>(差)>(2)>(2)>(2)>(40/61)



Where is the bug in this code?

44/61

S4 Nuances

Statistics with Biologists

R has pass-by-value semantics what does that mean for the following code:

```
> deleteSpecies <- function(alignment,
+
      species) {
      a <- which(alignment@species ==
+
          species)
```

```
alignment@species <- alignment@species[-a]
+
```

```
alignment@sequences <- alignment@sequences[-a]
+
```

```
+
      alignment
```

```
+ }
```

```
> B <- deleteSpecies(A, "tiger")
> A@species
```

```
[1] "tiger" "lion" "bear"
```

Replacement Methods

```
-----
```



```
Classes
```

- As we have already seen R has a somewhat strange type of function that allows us to modify objects in place.
- It is uncommon to define new replacement functions, however they are used guite frequently in day to day programming of R.
- Two examples are: names and colnames. Type "colnames" into the R window and hit "tab", notice the function "colnames<-"?

Useful S4 Functions

Statistics with Biologists

showMethods("summarize")

getGeneric("+"), getGenerics()

Example

We want to add a simple method to our alignment class so we can add alignments. Add a new method using setMethod to allow the user to perform the following: A1 + A2 which will construct a new alignment with the species from A1 and A2. Also, make sure that the alignments are of the same length.

101 (B) (2) (2) (2) 2 000 46/61

Replacement Methods

```
> a <- matrix(1:16, nrow = 4,
Biologists
                nco1 = 4)
         > colnames(a) <- paste("V", 1:4.</pre>
                sep = ",")
         +
         > colnames(a)
         > point <- list(x = 1, v = 2)
         > x.val <- function(x, value) {</pre>
               x$x <- value
         +
         + }
         > "x.val<-" <- function(x, value) {
                x$x <- value
         +
                return(x)
         + }
         > x.val(point, 10)
         > print(point)
```

lasses

	Replacement Methods		Packages
Statistics with R for Biologists Introduction Functions Debugging Classes The R Section	<pre>> x.val(point) <- 10 > print(point) What does the first print statement print? What about the second?</pre>	Statistics with R for Biologists Introduction Functions Debugging Classes Packages The R Session	 R is broken down into a number of core or base packages and runtime environment. How would we see what packages are installed? What packages are available on the system?
	- ロ - くろ・くろ・ (き・ マ - つ)QC 49 / 61		・ロ・ < グ ・ (注・) 注 の(C) 50/61
	Package Contents		Package Contents
Statistics with R for Biologics Introduction Functions Debugging Classes Packages The IL Session	 A package consists of the following directories and files: DESCRIPTION : A file describing the contents and version of the package NAMESPACE : A description of what R and C code will be available to the loaded package R : R source code which will be available when the package is installed depending on the NAMESPACE file src : A directory containing C or C++ source code which can be called directly from the R code in the package man : A directory containing the help files for a given package, these can be easily created using prompt() data A directory containing data sets available once the package is loaded 	Statistics with R for Biologists Introduction Functions Debugging Classes Packages The IL Session	In addition to the directories listed above packages can also contain the following directories: <i>demo</i> , <i>exec</i> , <i>inst</i> , <i>po</i> , and <i>tests</i> . In addition the following files can be in the top level directory: <i>INDEX</i> , <i>configure</i> , <i>cleanup</i> , <i>LICENSE</i> , <i>LICENCE</i> , and <i>COPYING</i> . These directories and files are less often used but sometimes important, to make a simple package we need only <i>DESCRIPTION</i> and <i>R</i>
	、		<ロン・(費)、(言)、(言)、(言)、(言)、(言)、(言)、(言)、(言)、(言)、(言

DESCRIPTION

Statistics with R for Biologists

ntroduction Functions Debugging Classes **Packages** Fhe R Session Package: HMM Title: An HMM for a K state model Version: 1.0 Author: James Bullard Depends: R (>= 2.6.1) Description: A package for the HMM parsing of tiling array data Maintainer: <bullard@stat.berkeley.edu> License: LCPL

The exact rules for this file can be found at: DESCRIPTION

Example: Making a Simple Package

We now want to build and install a simple R package to contain the code from this lecture. It is overkill to develop a package for a simple analysis, but some reasons to develop a package are:

- I keep data, documentation, and code together
- Develop code in other languages for use in R (C, C++)
- 3 Easily distribute the work we have done to collaborators
- 4 Have a standard structure for maintaining our work

One caveat is that sometimes building the package can be more trying because we need additional tools in order to get things to work. In Mac OS X we generally just need XCode and for windows we need the R tools found at:

www.murdoch-sutherland.com/Rtools.

(D) (B) (2) (2) 2 900 64/61

Example: Making a Simple Package

Statistics with R for Biologists

- Introduction Functions Debugging Classes Packages
- First, see if you can install a package from source, download the following URL to a local directory and try to install the package using the command line or the GUI. xtable
- Construct a simple package containing one function and one data set.
- 3 Install the package using R CMD INSTALL.
- Test that we can access our functions and data sets using require or library.
- use prompt to create a helpfile for one of the functions you have created.

The R Workspace

Statistics wit

Biologists

Biologists

- Like most programming environments the startup of R can be controlled by your environment (e.g. environment variables, or startup files in your home directory).
- The most important environment variable is R_LIBS. This environment variable dictates where packages are installed, so if you are on a shared system, or a system where you do not have admin rights then you want to use this variable to control where packages are installed.
- This variable shoulb be set in your .bashrc file (or in your environment variables widget in windows)
- You can check where R will check for packages by using the .libPaths function. This function additionally allows you to add directories to the search path.

101 (0) (2) (2) 2 900

	The R Workspace		Examining the R session
Statistics with R for Biologists Functions Debugging Classes Packages The R Session	<pre>• .Platform, .Machine • Additionally, you have a file called .Rprofile which can be used to set up some initial code. > dirname(.libPaths()) [1] "/Library/Frameworks/R.framework/Resources" > basename(.libPaths()) [1] "library"</pre>	Statistics with R for Biologists Introduction Functions Dubologging Classes Packages The R Session	<pre>• Often we want to know what packages / capabilities / options R is using. There are a number of relevant functions for examining the R session > sessionInfo() R version 2.7.1 RC (2008-06-20 r45965) i386-apple-darwin9.3.0 locale: en_US.UTF-8/en_US.UTF-8/C/C/en_US.UTF-8/en_US.UTF-8 attached base packages: [1] stats graphics grDevices [4] utils datasets methods [7] base Concentration Examining the R session</pre>
Statistics with R for Biologists Introduction Packages Packages The R Session	<pre>> capabilities() jpeg ng tlk TRUE TRUE TRUE X11 aqua http/ftp TRUE TRUE TRUE sockets libzml fifo TRUE TRUE TRUE cledit iconv NLS FALSE TRUE cordenate to the tr</pre>	Statistics with R for Biologists Introduction Punctions Dabagging Classes Packages The R Section	<pre>\$pkgType [1] "mac.binary" \$device [1] "pdf" > R.version platform i386-apple-darwin9.3.0 arch i386 os darwin9.3.0 system i386, darwin9.3.0 status RC major 2 minor 7.1</pre>

	R session

Statistics with R for Biologists

	year	2008
	month	06
	day	20
	svn rev	45965
he R Session	language	R
	version.string	R version 2.7.1 RC (2008-06-20 r45965)

(ロ)(費)(注)(注)(注)(注)(注)(注)(61/61