Module 2

Variables

Andrew Jaffe
Instructor

Getting Started

- You should have the latest version of R installed (R 3.0.1 as of 6/7/13)!
- Open R Studio
- Files --> New --> R Script
- Save the blank R script as "day1.R" in a directory of your choosing
- Add a comment header

Commenting in Scripts

Add a comment header to day1.R : '#' is the comment symbol

```
##################
# Title: Demo R Script
# Author: Andrew Jaffe
# Date: 6/10/2013
# Purpose: Demonstrate comments in R
####################

# this is a comment, nothing to the right of it gets read

# this # is still a comment - you can use many #'s as you want

# sometimes you have a really long comment, like explaining what you
#    are doing for a step in analysis. Take it to a second line
```

R as a calculator

```
> 2 + 2
```

```
[1] 4
```

```
> 2 * 4
```

```
[1] 8
```

```
> 2^3
```

```
[1] 8
```

R as a calculator

- The R console is a full calculator
- Try to play around with it:
  - +, -, /, * are add, subtract, multiply, and divide
  - ^ or ** is power
  - (and) work with order of operations

R as a calculator

```
> 2 + (2 * 3)^2
```

```
[1] 38
```

```
> (1 + 3)/2 + 45
```

```
[1] 47
```

R as a calculator

Try evaluating the following:

- `2+2*3/4-3`
- `2*3/4*2`
- `2^4-1`

R variables

- You can create variables from within the R environment and from files on your computer
- R uses "=" or "<-" to assign values to a variable name
- Variable names are case-sensitive, i.e. X and x are different

```
> x = 2
> x
```

```
[1] 2
```

```
> x * 4
```

```
[1] 8
```

```
> x + 2
```

```
[1] 4
```

R variables

- The most comfortable and familiar class/data type for many of you will be `data.frame`
- You can think of these as essentially Excel spreadsheets with rows (usually subjects or observations) and columns (usually variables)

```
> data(iris)
> head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

However, these are a fairly advanced way to store data!

R variables

- We will start with 1 dimensional classes first; these are often referred to as 'vectors'
- Vectors can have multiple observations, but each observation has to be the same class.

```
> class(x)
```

```
[1] "numeric"
```

```
> y = "hello world!"
> print(y)
```

```
[1] "hello world!"
```

```
> class(y)
```

```
[1] "character"
```

R variables

Try assigning your full name to an R variable called `name`

R variables

Try assigning your full name to an R variable called `name`

```
> name = "Andrew Jaffe"
> name
```

```
[1] "Andrew Jaffe"
```

The 'combine' function

The function `c()` collects/combines/joins single R objects into a vector of R objects. It is mostly used for creating vectors of numbers, character strings, and other data types.

```
> x <- c(1, 4, 6, 8)
> x
```

```
[1] 1 4 6 8
```

```
> class(x)
```

```
[1] "numeric"
```

The 'combine' function

Try assigning your first then last name as an R vector called `name2`

The 'combine' function

Try assigning your first then last name as an R vector called `name2`

```
> name2 = c("Andrew", "Jaffe")
> name2
```

```
[1] "Andrew" "Jaffe"
```

R variables

`length()`: Get or set the length of vectors (including lists) and factors, and of any other R object for which a method has been defined.

```
> length(x)
```

```
[1] 4
```

```
> y
```

```
[1] "hello world!"
```

```
> length(y)
```

```
[1] 1
```

R variables

What do you expect for the length of the `name` variable? What about the `name2` variable?

What are the lengths of each?

R variables

What do you expect for the length of the `name` variable? What about the `name2` variable?

What are the lengths of each?

```
> length(name)
```

```
[1] 1
```

```
> length(name2)
```

```
[1] 2
```

R variables

You can perform functions to entire vectors of numbers very easily.

```
> x + 2
```

```
[1]  3  6  8 10
```

```
> x * 3
```

```
[1]  3 12 18 24
```

```
> x + c(1, 2, 3, 4)
```

```
[1]  2  6  9 12
```

R variables

But things like algebra can only be performed on numbers.

```
> name2 * 4
```

```
Error: non-numeric argument to binary operator
```

```
> name + 2
```

```
Error: non-numeric argument to binary operator
```

R variables

And save these modified vectors as a new vector.

```
> y = x + c(1, 2, 3, 4)
> y
```

```
[1]  2  6  9 12
```

Note that the R object `y` is no longer "Hello World!" - It has effectively been overwritten by assigning new data to the variable

R variables

- You can get more attributes than just class

```
> ## ?str
> str(x)
```

```
 num [1:4] 1 4 6 8
```

```
> str(y)
```

```
 num [1:4] 2 6 9 12
```

Basic Summarization

`sum()`: takes the sum of all numeric variables in a vector

`mean()`: takes the mean of all numeric variables in a vector

`median()`: takes the median of all numeric variables in a vector

Review

- Creating a new script
- Using R as a calculator
- Assigning values to variables
- Performing algebra on numeric variables