

# Lecture 7 – *R* Refresher

## Learning Objectives:

1. Become proficient in the use of the *R* language.
  - 1.1 Understand the basic syntax of *R*, the organization of the workspace and environment.
  - 1.2 List the data types available in *R*.
  - 1.3 Become proficient in writing and using functions.
  - 1.4 Learn the basic elements of flow control.

**Announcement: Pick an *R* package to present during class 9/28 & 9/30!**

# Revisit conditionals

## Corrected script:

```
#!/bin/bash

File=${1:?No parameter provided}
if [ "$File" = "example.txt" ]; then
    echo "Here is the text..."
    cat $File
else
    echo "You must use example.txt"
fi
```

if [ "\$File" = "example.txt" ]; then

**These spaces are  
important!!**

- The [ is an alias of test, distinguished from wildcard [x] using whitespace.
- Proper syntax with spaces is result of test, differs from setting variables in bash, must provide spaces to distinguish arguments.

# Revisit conditionals

## Other syntax notes:

- Use double quotes most of the time:

```
[ "$File" = "example.txt" ]    not    [ $File = "example.txt" ]
```

- Don't use single quotes unless you have a wildcard character:

```
[ "$File" = "example.txt" ]    not    [ '$File' = 'example.txt' ]  
  
                                except [ "$File" = 'example*.txt' ]
```

- Use multiple test commands with && and ||:

```
if [ "$x" = "file" ] && [ "$y" = "back" ]; then
```

**not**

```
if [ "$x" = "file" && "$y" = "back" ]; then
```

# About *R*

## What is R?

- GNU-project language build around statistical computing and data-visualization graphics
- Integrated suite of software for handling data, running calculations, displaying graphics
- Open-access implementation of S (lots of S code runs in R).
- Can be linked with C, C++, Fortran (can be manipulated with C, C++, Java, or Python code).
- Many user-developed intermediate tools for analysis, curated and distributed by central repository (CRAN)

## Why use R?

- Free and open source, good for open science, reproducibility, and accessibility
- Publication-quality graphics
- Many statistical tests and models available
- Easy to learn
- Own documentation tool

## Why not use R?

- Slow for a lot of applications (loops)
- Not best at handling large data sets
- Not best at matrices and linear algebra calculations

# Basic Syntax in *R*

## R as a calculator:

```
> 2+2  
[1] 4
```

## Assigning a value:

```
> x <- 2    good anywhere  
> 2 -> x    equivalent to rightward form  
> x = 2     only good at top level/command line  
> x <<- 2   functions, search environments for similar variable, will redefine  
             or assigns to global environment
```

## Creating a vector:

```
> x <- 1:5    vector sequence 1 to 5  
> x <- seq(1,5)  
> x <- c(1,2,3,4,5)  
> y <- rep(1,5) vector of 1's that is 5 long
```

# Basic Syntax in *R*

“Everything that exists is an object; every operation is a function call.”

## Equivalent statements:

```
> x <- 2
> x<-2
> x <-2
> x<- 2
```

whitespace largely  
does not matter

## Incorrect syntax:

```
> x < - 2
> x< - 2
> x < -2
```

operators/functions  
must remain intact

## Functions:

```
> function(arg1,arg2,...)  or  > function ( arg1 ,arg2 , ... )
```

```
> mean(x)
```

```
> seq(1,5,by=0.05)  (1) arguments specified by exact name,
```

```
> seq(1,5,b=0.05)  (2) partial name match, or
```

**lazy  
evaluation!**

```
> seq(1,5,0.05)  (3) arguments specified by position
```

```
> rep(c(0,1),5)
```

arguments with multiple elements must be  
entered as vectors or lists

# Workspace and Scoping

## Examining Your Workspace:

- > `ls()`                      list objects in workspace
- > `rm(list=ls())`    clears workspace

**Environments:** frame that associates data objects, powers scoping

- global environment = user workspace
- environments can be nested
- functions and packages exist in their own environments, enclosed within the environment in which they were created

## Scoping:

lexical scoping

- name masking
- functions and variables
- fresh start

dynamic scoping

- variables are set at the time of a function call, not at the time of function creation

# Data Types

**Remember: everything is an object!**

Data Types of R Objects	Class homogeneous or heterogeneous?	Dimensions?
• Vectors	homogeneous	1D
• Lists	heterogeneous	1D
• Matrices	homogeneous	2D
• Arrays	homogeneous	nD
• Data Frames	heterogeneous	2D
• Factors		
• Functions		
• more!		

**No scalars in R!**

## Data Classes in R

- logical
- numeric
- integer
- complex
- character
- raw

**Check with:** `> class()`  
**or**  
`> typeof()`

**Convert with:** `> as.logical()`  
`> as.numeric()`  
`> as.integer()`  
**etc.**



# Functions

Remember: every operation is a function call!

## Components:

**name** → `f` **formal arguments:** `formals()` → `(x, y)`

```
f <- function(x, y) {  
  # Comment  
  x + y  
}
```

**body:** `body()` → `x + y`

## Types:

- primitive
- first-class

**environment:** `environment()`

**During variable calling:**  
**Beware of name masking!**

## Returns:

- implicit: last calculation
- explicit uses `return()`

## Forms:

prefix:

`mean(x)`

infix:

`x + y`

replacement:

`names(x) <- c("a", "b")`

special:

`for j in`

# Flow control: conditionals

## Simple conditional:

```
if(x == 1){  
  # code  
} else {  
  # other code  
}
```

## vectorized:

```
ifelse(condition,"if true","if false"))  
  
alpha<-c("TRUE","FALSE","TRUE","TRUE")  
ifelse(alpha,"heads","tails"))
```

## Multiple conditions:

```
if(x == 1){  
  # code  
} else if(x == 2) {  
  # other code  
} else {  
  # another block of code  
}
```

```
switch(  
  n,  
  once="Shame on you!",  
  twice="Shame on me!"  
)
```

# Flow control: loops

## Repeat:

```
repeat{print("yay!")}
```

**ctrl + c to kill**

```
repeat{  
  print("yay!")  
  n = n + 1  
  if (n == 100) break  
}
```

## While:

```
while (n < 100){  
  print("yay!")  
  n = n + 1  
}
```

## For:

```
for (n in 1:100) print(paste("yay it's ",n))
```

# More Information

<https://www.shell-tips.com/bash/if-statement/> – Scripting Error-Free Bash if statements

<http://duhi23.github.io/Analisis-de-datos/Cotton.pdf> – *Learning R* book

<https://bookdown.org/ndphillips/YaRrr/> – YaRrr! A Pirate's Guide to *R*

<https://r4ds.had.co.nz/> – *R for Data Science* Book

<http://users.metu.edu.tr/ozancan/R%20Graphics%20Cookbook.pdf> – *R Graphics Cookbook*

<https://adv-r.hadley.nz/> – *Advanced R* book

<http://adv-r.had.co.nz/Style.html> – Style guide (*Advanced R*)