Biostatistics Preparatory Course: Methods and Computing

Lecture 1

Intro, R Basics, Data Types/Structures

Introductions

- Instructor: Izzie Fulcher
- Email: isabelfulcher@g.harvard.edu

Structure of the Course

- Focus on *basic* programming in R
- Statistical topics will be mixed in

Structure of the Course

- Focus on *basic* programming in R
- Statistical topics will be mixed in
- Each class will consist of:
  - A short lecture on topics in R relevant for biostatistics (20-30 minutes)
  - For the remainder of the session, we will complete several exercises, often broken up into groups
- Solutions to exercises and slides will be posted after the each session
- Bringing a laptop is recommended, since programming is most easily learned by *doing*

Topics covered

The first two sessions will go over basics in R programming, then we will start to mix in statistical topics:

- Probability distributions
- Linear regression (if time, generalized linear models)
- Monte Carlo simulations
- Maximum likelihood estimation
- Bootstrap

Topics covered

The first two sessions will go over basics in R programming, then we will start to mix in statistical topics:

- Probability distributions
- Linear regression (if time, generalized linear models)
- Monte Carlo simulations
- Maximum likelihood estimation
- Bootstrap

All programming will be done in R, but we will also use,

- Rmarkdown and LaTeX
- Cluster computing

What is R?

- Free programming language for statistical computing and graphics
- Easy way to distribute one's own packages for novel methods

How to get R and Rstudio

- R — the language and GUI (Graphical User Interface)
  - https://cran.r-project.org

How to get R and Rstudio

- R — the language and GUI (Graphical User Interface)
  - https://cran.r-project.org
- Rstudio — an IDE (Integrated Development Environment)
  - http://www.rstudio.com/products/rstudio/download/

R Basics

Definition (Variables)

Names that are assigned values (which can be of various data types)

```
# Example
a <- 3
```

- Some variables are built in (eg., `pi`)

R Basics

Definition (Variables)

Names that are assigned values (which can be of various data types)

```
# Example
a <- 3
```

- Some variables are built in (eg., `pi`)

Definition (Functions)

A piece of code that performs a task (may take one or more arguments)

```
# Examples
log(3)
exp(3)
```

- Multiple arguments in functions are separated by commas

R Basics: Types of Commands

Definition (Assignment)

The command is evaluated, its value passed to variable and not printed

```
# Examples:
a <- a + 3
b <- log(3)
```

R Basics: Types of Commands

Definition (Assignment)

The command is evaluated, its value passed to variable and not printed

```
# Examples:
a <- a + 3
b <- log(3)
```

Definition (Expression)

The command is evaluated, its value is printed and not retained

```
# Examples:
1 + 1
a + 3
log(a)
```

Removing objects from workspace

- Objects stored in the workspace can be displayed with `ls()`
- Variables can be deleted with `rm()`
- `rm(list = ls())` will remove all objects from the current workspace

R basics: Packages

- Packages are ways to distribute content and may contain:
  - Datasets
  - Functions
- Packages must first be installed and loaded before they can be used:

```
# Example:
install.packages("matrixStats")
library("matrixStats")
```

- You can check the documentation for a package with `help(package = ‘packagename’)`

Exercise 1

- Try out commands of your own, both assignment and expression:
  1. Create a variable x with value 2
  2. Subtract 7 from x
  3. Create a new variable $y = x - 7$
- Install and load a package
  - `ggplot2`
  - `mvtnorm`
  - `stringr`
- Use basic functions, such as:
  `log()`, `exp()`, `abs()`, `sqrt()`, `ceiling()`, `trunc()`
- Practice using the help (? followed by function name)
  - `ceiling`
- Once you have completed the above, remove all objects from your workspace

Common Data Types in R

Definition (Character)

Always in quotations and do not have numerical value

```
# Example:
"abc123"
```

Common Data Types in R

Definition (Character)

Always in quotations and do not have numerical value

```
# Example:
"abc123"
```

Definition (Boolean)

A logical value that takes one of the values TRUE or FALSE

```
# Example:
TRUE
```

Common Data Types in R

Definition (Character)

Always in quotations and do not have numerical value

```
# Example:
"abc123"
```

Definition (Boolean)

A logical value that takes one of the values TRUE or FALSE

```
# Example:
TRUE
```

Definition (Numeric)

A numeric value (integer or decimal)

```
# Examples:
24
pi
```

Common Data Structures in R: Vectors

- Ways to create vectors:

```
vec1 <- c(1,2,3)
vec2 <- 1:10
vec3 <- seq(1,10,0.5)
vec4 <- rep(1,10)
```

Common Data Structures in R: Vectors

- Ways to create vectors:

```
vec1 <- c(1,2,3)
vec2 <- 1:10
vec3 <- seq(1,10,0.5)
vec4 <- rep(1,10)
```

- Vectors are referenced by the index for both assignment and expression:

```
vec1[1]
vec1[1] <- 6
```

Common Data Structures in R: Matrices

- Can create a matrix from scratch:

```
mymat <- matrix(1:10, nrow=2, ncol=5, byrow=TRUE)
mymat2 <- matrix(0, nrow=3,ncol=4)
```

  - If the first argument to `matrix` is a vector, `nrow*ncol` must be the length of the vector
  - If the first argument is a scalar, the matrix will contain only the scalar value

- Can also combine vectors of same length to form matrices:

```
vec1 <- c(1, 2, 3)
vec2 <- c(2, 3, 4)
mat1 <- rbind(vec1, vec2)
mat2 <- cbind(vec1, vec2)
```

Common Data Structures in R: Matrices

- Can create a matrix from scratch:

```
mymat <- matrix(1:10, nrow=2, ncol=5, byrow=TRUE)
mymat2 <- matrix(0, nrow=3,ncol=4)
```

  - If the first argument to `matrix` is a vector, `nrow*ncol` must be the length of the vector
  - If the first argument is a scalar, the matrix will contain only the scalar value

- Can also combine vectors of same length to form matrices:

```
vec1 <- c(1, 2, 3)
vec2 <- c(2, 3, 4)
mat1 <- rbind(vec1, vec2)
mat2 <- cbind(vec1, vec2)
```

- Matrices are referenced by their row and column number for assignment and expression:

```
mymat[1, 2]
mymat[1, 2] <- 6
```

Other Data Structures in R: Combining Data Types

- Vectors and matrices contain one type of data at a time

Definition (List)

A collection of objects of multiple types

```
a <- 1
b <- "a string :)"
mylist <- list(a, b)
mylist[[1]]
```

Other Data Structures in R: Combining Data Types

- Vectors and matrices contain one type of data at a time

Definition (List)

A collection of objects of multiple types

```
a <- 1
b <- "a string :)"
mylist <- list(a, b)
mylist[[1]]
```

Definition (Data frame)

A list of vectors of the same length (can be manipulated like a matrix)

```
x <- c("a", "b", "c")
y <- c(TRUE, TRUE, FALSE)
z <- data.frame(x, y)
z[1, 1]
z$x
```

Some Useful Commands

- `class()` will display the object class
- `dim()` will display the dimension of a matrix
- `length()` will display the number of elements in an object

Exercise 2

- Create a 3×4 matrix, M, from a vector of length 12 containing all 1s
- Create 3 vectors of length 5: one character, one numeric and one boolean called $char.vec$, $num.vec$, and $bool.vec$ respectively
- Create a data frame called $mydata$ with the 3 vectors you created
- Create a list called $mylist$ with the 3 vectors you created
- Convert M to a data frame. Check you have done this using `class()`
- Name the columns of M using `colnames()`

Generating empty structures

- Often, it's useful to make empty "containers" into which you will put results
- To make empty lists or vectors: `vector` function

```
empty.list     <- vector(length=3, mode="list")
empty.num.vec  <- vector(length=2, mode="numeric")
empty.char.vec <- vector(length=5, mode="character'
```

- To make an empty matrix: make a matrix filled with `NA`

```
empty.matrix <- matrix(NA, nrow=5, ncol=6)
```

  - This makes it easy to see if something didn't get filled in