R Programming II

EPID 799B

Nathaniel MacNell

Overview

- Functions & Arguments
- Creating Vectors and Data Frames
- Using [ ] on the Left
  - Recoding
  - Classifying

- **Activity**: make up a dataset, try regression

Functions and Arguments

- We've been using functions with one **argument** (input). Functions can have multiple arguments!

- Separate arguments with `,` (just like indices in [])
- Specify arguments with =

```
rnorm(n=1000, mean=10, sd=2)
```

Example: Creating Vectors

Functions can combine named and unnamed arguments:

- **Concatenate**

```
c(8,6,7,5,3,0,9)    # arbitrary sequence
```

- **Repeat**

```
rep(9, times=100)    # repeated sequence
```

- **Sequence**

```
seq(from=1,to=1000,by=5) # step sequence
```

Example: Creating Datasets

Sometimes the argument **names** can be used to specify parts of the output:

```
data.frame(a=x , b=y , c=z )
```

Creates a data frame with variables named a, b, and c (or any other names).

Using [] on the LEFT and RIGHT

LEFT Replace

object[subset] **<- x**

- Replace a subset of the object with **x**.
- **x** and the subset must be the same size!

RIGHT Reduce

x <- object[subset]

- Copy a subset into **x**.
- **x** is any size (it's new).

Why does this work?

- The left side of **<-** (or equivalently **=**) is a reference
- If the object referenced doesn't exist, R creates it
- This doesn't always work (it has to make sense and the SIZE of the left and right has to match)

new <- 1+2	OK
vector[2:3] <- c(1,2)	OK
vector[4] <- c(1,2	NO!
1+1 <- 3	NO!

Example: Changing Names

You can rename variables in a dataset using the replace approach:

```
# build new names vector
newnames <- c(“exposure” , ”outcome” , ”covariate”)


# replace old names
names(dataset) <- newnames

# Replace just one name
names(dataset)[2] <- “disease”
```

Example: Classifying Variables

```
ex <- 1:100

# create new empty vector
new <- rep(NA, length(example) )

# fill in empty vector based on old one
new[ ex<30 ] <- “low”
new[ (ex>=30) & (ex<75) ] <- “medium”
New[ ex>=75 ] <= “high”
```

Activity: Simulated Regression

1. Create a dataset with random variables x (Exposure), z (covariate), and e (residual). Use the function **rnorm(n=, mean=, sd=)** with filled in arguments. (specify any distributions you want)
2. Create a variable y based on x, z, and e and a linear model equation of your choice.
3. Create a binary variable splitting y on its mean.
4. Use glm(y ~ x + z) to build a model object.
5. Run summary() on the glm object to get results.
6. Add the family=binomial("logit") to glm and use the binary outcome variable in place of y.