R and Ebola: Meeting 3

Brooke Anderson

November 21, 2014

Ebola data

Pull in your ebola data again so we'll have it ready for examples:

```
## If necessary, use setwd() to get to the right directory
ebola <- read.table("country_timeseries.csv", sep = ",",
                    header = TRUE)
ebola[1:3, 1:5]
```

```
##         Date Day Cases_Guinea Cases_Liberia Cases_SierraLeone
## 1  11/2/2014 225         1731            NA              4759
## 2 10/31/2014 222           NA          6525                NA
## 3 10/29/2014 220         1667            NA              5338
```

Two functions to help with loops

Two functions that are very useful for loops are `print()` and `paste()`:

```
print(1:3)
```

```
## [1] 1 2 3
```

```
paste("Brooke", "Anderson")
```

```
## [1] "Brooke Anderson"
```

```
paste("Brooke", "Anderson", sep = ".")
```

```
## [1] "Brooke.Anderson"
```

Two functions to help with loops

```
rankings.name <- c("First", "Second", "Third")
rankings.num <- c(1:3)
paste(rankings.name, "is what we call #", rankings.num)
```

```
## [1] "First is what we call # 1"  "Second is what we call # 2"
## [3] "Third is what we call # 3"
```

Loop structure

```
for([index] in [vector]){      ## Generic code
        [code you want to run for a single loop]
}
```

Simple loops

```
for(i in 1:3){
        print("For this loop, i is:")
        print(i)
}
```

```
## [1] "For this loop, i is:"
## [1] 1
## [1] "For this loop, i is:"
## [1] 2
## [1] "For this loop, i is:"
## [1] 3
```

Simple loops

```
for(i in 1:3){
        print(paste("For this loop, i is", i))
}
```

```
## [1] "For this loop, i is 1"
## [1] "For this loop, i is 2"
## [1] "For this loop, i is 3"
```

Using `i` to index

Remember how we can use indexing to just get certain parts of a vector or dataframe:

```
my.family <- c("Reeves", "Brooke", "Cord")
my.family[1]
```

```
## [1] "Reeves"
```

Using `i` to index

We can take advantage of this in loops:

```
for(i in 1:3){
        print(my.family[i])
}
```

```
## [1] "Reeves"
## [1] "Brooke"
## [1] "Cord"
```

This turns out to be very powerful…

Using `i` to index

To figure out a loop, try the code inside the loop a few times with `i` equal to different values it would get in the loop:

```
i <- 1
print(my.family[i])
```

```
## [1] "Reeves"
```

```
i <- 2
print(my.family[i])
```

```
## [1] "Brooke"
```

Practice

Now you try:

- Create a vector, `Country` of all the countries in the ebola dataset. Use the `paste()` function to create a vector called `case.colnames` the gives the names of the columns for all countries.
- Create vectors of first and last names of everyone in your group, then write a loop to print "[Name] is in our group" for everyone.
- Create a loop that prints, on each run, "Die roll # [i] is: [random number between 1 and 6]".

Hints: Create separate `first.name` and `last.name` vectors and use `paste()` in your loop to print each person's full name. Use the `sample()` function to get the random number (e.g., `sample(1:6, size = 1)`).

Using loops

According to programmers, you should never repeat yourself.

You can use loops to do the same thing a lot of times. For example, we could use a loop to create a new dataframe with, for each country, the mean number of ebola cases.

Basically, we'll create a empty dataframe and use the loop to fill it in.

```
##        country nonmissing.obs
## 1       Guinea             NA
## 2      Liberia             NA
## 3   SierraLeone             NA
## 4      Nigeria             NA
## 5      Senegal             NA
## 6 UnitedStates             NA
## 7        Spain             NA
## 8         Mali             NA
```

Using loops

First, we'll need a vector of all the column names for cases.

```
Country <- c("Guinea", "Liberia", "SierraLeone", "Nigeria",
             "Senegal", "UnitedStates", "Spain", "Mali")
case.colnames <- paste("Cases", Country, sep = "_")
case.colnames
```

```
## [1] "Cases_Guinea"       "Cases_Liberia"      "Cases_SierraLeone"
## [4] "Cases_Nigeria"      "Cases_Senegal"      "Cases_UnitedStates"
## [7] "Cases_Spain"        "Cases_Mali"
```

Using loops

Next, we'll create a dataframe that we'll fill in with the loop:

```
ebola.cases <- data.frame(country = Country,
                          nonmissing.obs = NA)
ebola.cases
```

```
##         country nonmissing.obs
## 1        Guinea             NA
## 2       Liberia             NA
## 3  SierraLeone             NA
## 4       Nigeria             NA
## 5       Senegal             NA
## 6 UnitedStates             NA
## 7         Spain             NA
## 8          Mali             NA
```

Using loops

Now, let's think about what we want to do for each country:

```
mean(ebola[ , "Cases_Guinea"], na.rm = TRUE)
```

```
## [1] 558.7
```

Equivalently:

```
i <- 1
case.colnames[i]
```

```
## [1] "Cases_Guinea"
```

```
mean.cases <- mean(ebola[ , case.colnames[i]], na.rm = TRUE)
mean.cases
```

```
## [1] 558.7
```

Using loops

Once you've calculated what you want, you can use indexing to put the number in your `ebola.cases` dataframe:

```
head(ebola.cases, 2)
```

```
##   country nonmissing.obs
## 1  Guinea             NA
## 2 Liberia             NA
```

```
ebola.cases[i, "nonmissing.obs"] <- 1
head(ebola.cases, 2)
```

```
##   country nonmissing.obs
## 1  Guinea              1
## 2 Liberia             NA
```

Using loops

Once you've calculated what you want, you can use indexing to put the number in your `ebola.cases` dataframe:

```
ebola.cases[i, "nonmissing.obs"] <- mean.cases
head(ebola.cases, 2)
```

```
##   country nonmissing.obs
## 1  Guinea          558.7
## 2 Liberia             NA
```

Using loops

Now you can put this all together:

```
Country <- c("Guinea", "Liberia", "SierraLeone", "Nigeria",
             "Senegal", "UnitedStates", "Spain", "Mali")
case.colnames <- paste("Cases", Country, sep = "_")
ebola.cases <- data.frame(country = Country,
                          nonmissing.obs = NA)

for(i in 1:length(Country)){
        mean.cases <- mean(ebola[ , case.colnames[i]], na.rm = TRUE)
        ebola.cases[i, "nonmissing.obs"] <- mean.cases
}
```

Using loops

```
head(ebola.cases)
```

```
##         country nonmissing.obs
## 1        Guinea         558.653
## 2       Liberia        1088.343
## 3  SierraLeone         982.203
## 4       Nigeria          15.867
## 5       Senegal           1.118
## 6 UnitedStates           2.700
```

Practice

Now you try:

- Use a loop get the range of each country's number of ebola cases from our data. Put these values in a new dataframe with the columns `Country`, `Min.Cases`, and `Max.Cases`.
- Use a loop to plot ebola cases by day for each of the countries.

Practice

Step-by-step for first exercise:

- Create a vector called `Country` that lists all of our countries
- Create a vector called `case.colnames` the gives the names of the columns for all countries
- Create a dataframe called `Case.Ranges` that uses the `Country` vector as a `country` column and then also has columns for `Min.Cases` and `Max.Cases`. To start, these will just be full of `NA`s.
- Create a loop where `i` goes from 1 to the length of your `Country` vector.
- For each loop, use `range(ebola, na.rm = TRUE)` to get the range of cases for that country.
- For each loop, use indexing to put this range (it will be two values) in the right places in your `Case.Ranges` dataframe.