Module 12

R Programming

Andrew Jaffe
Instructor

R 'programming'

Now we are going to switch gears a little bit, and talk about some of the more traditional programming that you can do in R.

You can do very flexible things, but at a cost of more difficult notation, and having to actually write programming statements. There are slight notation differences as well, including the use of curly `{ }` brackets

We are going to cover `for` loops and `if` statements

'for' Loops

These allow you to iterate over certain observations or subsets of observations

The syntax is:

```
for(*var* in seq) {
do something
}
```

Typically they look something like:

```
for(i in 1:nrow(dat)) {
  something(dat[i,])
}
```

'for' loops

These are essentially fancier `apply` statements

For example,

```
> for (i in 1:10) {
+     print(i)
+ }
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
```

'for' loops

Here's how they can be more flexible:

```
> Index = c(3, 6, 7, 20, 32, 100, 234, 1000, 6543)
> for (i in 1:length(Index)) {
+     print(Index[i])
+ }
```

```
[1] 3
[1] 6
[1] 7
[1] 20
[1] 32
[1] 100
[1] 234
[1] 1000
[1] 6543
```

Note that the first time through the body of the loop, `i` takes the value 1, then evaluates the body. Then, `i` takes the value 2, and evaluates the body, until `i = length(Index)`, then it stops.

'for' loops

They are essentially more useful than apply statements when you are working with two sets of matching datasets or vectors.

```
> myList = vector("list", length = 4)
> mat1 = matrix(rnorm(8), nc = 4)
> mat2 = matrix(rnorm(8), nc = 4)
> mat1
```

```
        [,1]    [,2]   [,3]    [,4]
[1,] -0.5069 -0.6219 0.9216 -0.5670
[2,]  0.8554 -1.2623 2.5747 -0.7054
```

```
> mat2
```

```
         [,1]   [,2]   [,3]   [,4]
[1,]  0.09993  1.558  3.443 -1.306
[2,] -0.74297 -1.843 -0.295  1.693
```

```
> for (i in seq(along = myList)) {
+     myList[[i]] = cbind(mat1[, i], mat2[, i])
+ }
> myList
```

```
[[1]]
        [,1]     [,2]
[1,] -0.5069  0.09993
[2,]  0.8554 -0.74297

[[2]]
        [,1]   [,2]
[1,] -0.6219  1.558
[2,] -1.2623 -1.843

[[3]]
       [,1]   [,2]
[1,] 0.9216  3.443
[2,] 2.5747 -0.295

[[4]]
        [,1]   [,2]
[1,] -0.5670 -1.306
[2,] -0.7054  1.693
```

'for' loops

```
> i = 1
> cbind(mat1[, i], mat2[, i])
```

```
        [,1]     [,2]
[1,] -0.5069  0.09993
[2,]  0.8554 -0.74297
```

```
> i = 2
> cbind(mat1[, i], mat2[, i])
```

```
        [,1]   [,2]
[1,] -0.6219  1.558
[2,] -1.2623 -1.843
```

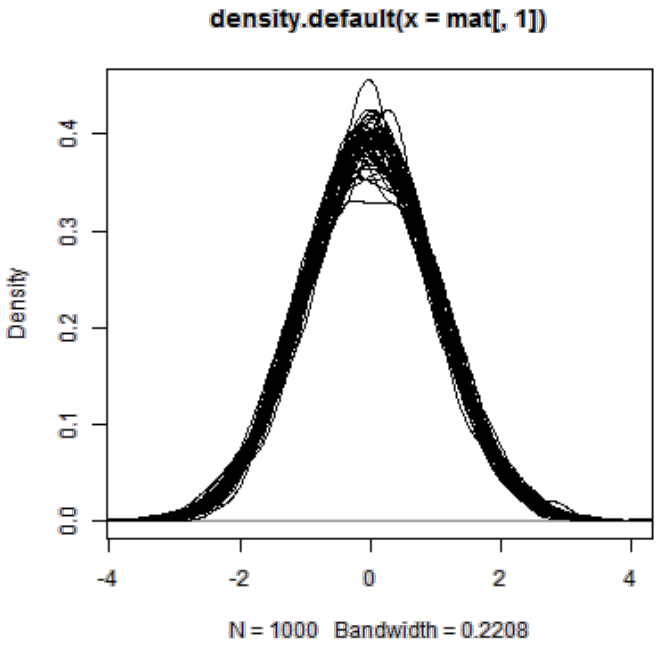
```
> i = 3
> cbind(mat1[, i], mat2[, i])
```

```
       [,1]   [,2]
[1,] 0.9216  3.443
[2,] 2.5747 -0.295
```

'for' loops

These are useful for making many columns worth of density plots

```
> mat = matrix(rnorm(1000 * 50), nc = 50)
> plot(density(mat[, 1]), ylim = c(0, 0.45))
> for (i in 2:ncol(mat)) {
+     lines(density(mat[, i]))
+ }
```



'for' loops

You can also integrate with lists.

```
> outList = vector("list", 10)
> start = 1:10
> end = sample(1:100, 10)
> for (i in seq(along = outList)) {
+     outList[[i]] = start[i]:end[i]
+ }
> outList
```

```
[[1]]
[1] 1 2 3 4 5 6 7 8 9

[[2]]
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
[24] 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
[47] 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70
[70] 71 72 73 74 75 76 77 78 79

[[3]]
 [1]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[24] 26 27 28 29 30 31 32 33 34

[[4]]
 [1]  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
[24] 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
[47] 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
[70] 73

[[5]]
 [1]  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
```

'if' statements

You can put 'if' statements inside of 'for' loops

```
for(i in 1:nrow(dat)) {
  if(dat$x > num) {
    dat$y[i] = something
  } else {
    dat$y[i] = something else
  }
}
```

Example

```
> makeIndexes = split(1:nrow(cars), cars$Make)
> lapply(makeIndexes, head, n = 4)[1:3]
```

```
$ACURA
[1] 10039 13026 13631 14250

$BUICK
[1] 185 233 258 346

$CADILLAC
[1] 3372 4517 8500 9664
```

```
> pval = rep(NA, length(makeIndexes))
> for (i in 1:length(makeIndexes)) {
+     ind = makeIndexes[[i]]
+     if (length(ind) > 1) {
+         f = lm(VehBCost ~ VehOdo, data = cars, subset = ind)
+         pval[i] = summary(f)$coef[2, 4]
+     }
+ }
> names(pval) = names(makeIndexes)
>
> i = 1
> ind = makeIndexes[[i]]
> str(ind)
```

```
 int [1:33] 10039 13026 13631 14250 16392 17289 17889 17979 18166 22044 ...
```

```
> f = lm(VehBCost ~ VehOdo, data = cars, subset = ind)
> summary(f)$coef[2, 4]
```

```
[1] 0.4932
```

```
> pval
```

```
       ACURA        BUICK     CADILLAC    CHEVROLET     CHRYSLER
   4.932e-01    1.877e-05    1.064e-06    2.834e-06    1.128e-78
       DODGE         FORD          GMC        HONDA       HUMMER
   1.494e-10    2.584e-27    1.626e-01    2.490e-13           NA
     HYUNDAI     INFINITI       ISUZU         JEEP          KIA
   1.072e-04    9.737e-04    2.697e-04    2.723e-16    3.765e-18
       LEXUS      LINCOLN        MAZDA      MERCURY         MINI
   5.014e-09    7.061e-01    1.364e-41    2.953e-04    8.709e-02
  MITSUBISHI       NISSAN   OLDSMOBILE     PLYMOUTH      PONTIAC
   3.357e-26    3.594e-10    6.956e-08          NaN   3.305e-154
      SATURN        SCION       SUBARU       SUZUKI       TOYOTA
   4.363e-40    3.115e-07    8.062e-01    4.285e-32    1.395e-16
TOYOTA SCION   VOLKSWAGEN        VOLVO
          NA    4.738e-06    1.269e-02
```

Note you can also do with with `sapply`

```
> pval2 = sapply(makeIndexes, function(ind) {
+     if (length(ind) > 1) {
+         f = lm(VehBCost ~ VehOdo, data = cars, subset = ind)
+         summary(f)$coef[2, 4]
+     } else NA
+ })
> all.equal(pval, pval2)
```

```
[1] TRUE
```

Example

Now we can read in many files into a list

```
> fn = list.files("Reports/", pattern = ".txt", full.names = TRUE)
> name = list.files("Reports/", pattern = ".txt", full.names = FALSE)
> head(fn)
```

```
[1] "Reports/April_2009_Report.txt"  "Reports/April_2010_Report.txt"
[3] "Reports/April_2011_Report.txt"  "Reports/August_2009_Report.txt"
[5] "Reports/August_2010_Report.txt" "Reports/August_2011_Report.txt"
```

```
> fileList = lapply(fn, read.delim, header = TRUE, as.is = TRUE)
> names(fileList) = name
> sapply(fileList, dim)[, 1:5]
```

```
     April_2009_Report.txt April_2010_Report.txt April_2011_Report.txt
[1,]                   287                   324                   359
[2,]                    10                    10                    10
     August_2009_Report.txt August_2010_Report.txt
[1,]                    353                   369
[2,]                     10                    10
```

```
> lapply(fileList[1:5], head, n = 2)
```

```
$April_2009_Report.txt
    id    sex   treat   age bgDrugs height weight block recruitDate   bmi
1 1072 Female Control 51.00  asprin  63.84  131.3     d          21 22.64
2 1073 Female Control 54.81 tylenol  66.10  117.2     b           1 18.85

$April_2010_Report.txt
    id    sex treat   age bgDrugs height weight block recruitDate   bmi
1 4337 Female  Case 46.91    none  64.95  140.6     f          25 23.43
2 4338 Female  Case 47.95    none  66.47  143.3     f          14 22.81

$April_2011_Report.txt
    id  sex   treat   age bgDrugs height weight block recruitDate   bmi
1 7780 Male    Case 53.93  asprin  70.12  175.0     f          29 25.02
2 7781 Male Control 62.77 tylenol  71.02  153.1     b          29 21.34

$August_2009_Report.txt
    id  sex   treat   age bgDrugs height weight block recruitDate   bmi
1 2051 Male Control 56.76 tylenol  70.47  168.0     f           2 23.78
2 2052 Male    Case 50.14  asprin  69.56  172.3     c           1 25.04
```