Programming: more loop examples

Stat 133 with Gaston Sanchez
Creative Commons Attribution Share-Alike 4.0 International CC BY-SA
Traversing a matrix (or any 2-dim object)
Writing a function

In this slides, we provide several examples of loops and similar operations applied on matrices (or any other 2-dimensional object)

The purpose is to give you a taste of how to use loops (at least conceptually) on rectangular objects.
Traversing cells of a matrix

<table>
<thead>
<tr>
<th>row1</th>
<th>col1</th>
<th>col2</th>
<th>col3</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_{11}</td>
<td>x_{12}</td>
<td>x_{13}</td>
<td></td>
</tr>
<tr>
<td>row2</td>
<td>x_{21}</td>
<td>x_{22}</td>
<td>x_{23}</td>
</tr>
<tr>
<td>row3</td>
<td>x_{31}</td>
<td>x_{32}</td>
<td>x_{33}</td>
</tr>
<tr>
<td>etc</td>
<td>etc</td>
<td>etc</td>
<td></td>
</tr>
</tbody>
</table>

etc
# looping through a matrix
X <- matrix(runif(12), nrow = 4, ncol = 3)

# looping row by row
for (i in 1:nrow(X)) {
    print(X[i, ])
}

# looping column by column
for (j in 1:ncol(X)) {
    print(X[, j])
}
# looping through a matrix
X <- matrix(runif(12), nrow = 4, ncol = 3)

# looping row by row
for (i in 1:nrow(X)) {
    print(X[i, ])
}

# looping column by column
for (j in 1:ncol(X)) {
    print(X[, j])
}
# looping cell by cell

```r
for (i in 1:nrow(X)) {
    for (j in 1:ncol(X)) {
        print(X[i,j])
    }
}
```
# looping cell by cell

for (i in 1:nrow(X)) {
    for (j in 1:ncol(X)) {
        print(X[i,j])
    }
}

# looping cell by cell (equivalent)

for (j in 1:ncol(X)) {
    for (i in 1:nrow(X)) {
        print(X[i,j])
    }
}
Row Sums
## Row Sums

<table>
<thead>
<tr>
<th></th>
<th>col1</th>
<th>col2</th>
<th>col3</th>
</tr>
</thead>
<tbody>
<tr>
<td>row1</td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
<td>$x_{13}$</td>
</tr>
<tr>
<td>row2</td>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$x_{23}$</td>
</tr>
<tr>
<td>row3</td>
<td>$x_{31}$</td>
<td>$x_{32}$</td>
<td>$x_{33}$</td>
</tr>
<tr>
<td>etc</td>
<td>etc</td>
<td>etc</td>
<td>etc</td>
</tr>
</tbody>
</table>

Sum 1  Sum 2  Sum 3
# vector of row sums

```r
row_sums <- rep(0, nrow(X))
```

```r
for (i in 1:nrow(X)) {
    aux <- 0
    for (j in 1:ncol(X)) {
        aux <- aux + X[i,j]
    }
    row_sums[i] <- aux
}
```
Column Sums
## Column Sums

<table>
<thead>
<tr>
<th>row1</th>
<th>col1</th>
<th>col2</th>
<th>col3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
<td>$x_{13}$</td>
</tr>
<tr>
<td>row2</td>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$x_{23}$</td>
</tr>
<tr>
<td>row3</td>
<td>$x_{31}$</td>
<td>$x_{32}$</td>
<td>$x_{33}$</td>
</tr>
<tr>
<td></td>
<td>$etc$</td>
<td>$etc$</td>
<td>$etc$</td>
</tr>
</tbody>
</table>

- **Sum 1**: \[ \text{Sum 1} \]
- **Sum 2**: \[ \text{Sum 2} \]
- **Sum 3**: \[ \text{Sum 3} \]
# vector of column sums

```r
col_sums <- rep(0, ncol(X))

for (j in 1:ncol(X)) {
    aux <- 0
    for (i in 1:nrow(X)) {
        aux <- aux + X[i,j]
    }
    col_sums[j] <- aux
}
```
Built-in functions:

- \texttt{rowSums(X)}
- \texttt{colSums(X)}
## Column means

<table>
<thead>
<tr>
<th></th>
<th>col1</th>
<th>col2</th>
<th>col3</th>
</tr>
</thead>
<tbody>
<tr>
<td>row1</td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
<td>$x_{13}$</td>
</tr>
<tr>
<td>row2</td>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$x_{23}$</td>
</tr>
<tr>
<td>row3</td>
<td>$x_{31}$</td>
<td>$x_{32}$</td>
<td>$x_{33}$</td>
</tr>
<tr>
<td>etc</td>
<td>etc</td>
<td>etc</td>
<td>etc</td>
</tr>
</tbody>
</table>

\[
\bar{X}_1 \quad \bar{X}_2 \quad \bar{X}_3
\]
X <- matrix(runif(12), nrow = 4, ncol = 3)

x1_mean <- mean(X[,1])
x2_mean <- mean(X[,2])
x3_mean <- mean(X[,3])

What if you had many more columns?
# one option

```r
X <- matrix(runif(12), nrow = 4, ncol = 3)

x_means <- c()
for (j in 1:ncol(X)) {
  x_means <- c(x_means, mean(X[, j]))
}

x_means
```
# another option

```r
X <- matrix(runif(12), nrow = 4, ncol = 3)

x_means <- rep(0, ncol(X))
for (j in 1:ncol(X)) {
  x_means[j] <- mean(X[, j])
}

x_means
```
Built-in functions:

- rowMeans(X)
- colMeans(X)
Apply functions
Function applied to columns

<table>
<thead>
<tr>
<th>row1</th>
<th>col1</th>
<th>col2</th>
<th>col3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
<td>$x_{13}$</td>
</tr>
<tr>
<td>row2</td>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$x_{23}$</td>
</tr>
<tr>
<td>row3</td>
<td>$x_{31}$</td>
<td>$x_{32}$</td>
<td>$x_{33}$</td>
</tr>
<tr>
<td>etc</td>
<td>etc</td>
<td>etc</td>
<td>etc</td>
</tr>
</tbody>
</table>

FUN(col 1)  FUN(col 2)  FUN(col 3)
### Function applied to rows

<table>
<thead>
<tr>
<th>row1</th>
<th>col1</th>
<th>col2</th>
<th>col3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
<td>$x_{13}$</td>
</tr>
<tr>
<td>row2</td>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$x_{23}$</td>
</tr>
<tr>
<td>row3</td>
<td>$x_{31}$</td>
<td>$x_{32}$</td>
<td>$x_{33}$</td>
</tr>
<tr>
<td></td>
<td><em>etc</em></td>
<td><em>etc</em></td>
<td><em>etc</em></td>
</tr>
</tbody>
</table>

The function `FUN` is applied to each row, as indicated by the arrows pointing to `FUN(row 1)`, `FUN(row 2)`, and `FUN(row 3)`.
apply(X, MARGIN, FUN, ...)

Gaston Sanchez
apply(X, MARGIN, FUN, ...)

matrix or array

Which margin:
1 = rows
2 = columns
etc

function

Optional arguments to FUN
X <- matrix(runif(12), nrow = 4, ncol = 3)

# row sums
apply(X, 1, sum)

# row sums removing missing values
apply(X, 1, sum, na.rm = TRUE)
```r
X <- matrix(runif(12), nrow = 4, ncol = 3)

# row minima
apply(X, 1, min)

# row minima removing missing values
apply(X, 1, min, na.rm = TRUE)
```
X <- matrix(runif(12), nrow = 4, ncol = 3)

# column maxima
apply(X, 2, max)

# column maxima removing missing values
apply(X, 2, max, na.rm = TRUE)
More statistics
stats <- function(y, na.rm = FALSE) {
  c('min' = min(x, na.rm = na.rm),
    'avg' = mean(x, na.rm = na.rm),
    'max' = max(x, na.rm = na.rm))
}

stats(1:10)

apply(X, 2, stats)