Building Blocks of Coding

Kelly Bodwin

January 14/15, 2019

There are a few major ideas that appear in almost any coding language, and R is no exception. Today you will practice using typical programming building blocks in R.

We will do some examples using the dataset `mtcars()` in R.

```
data(mtcars)
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

Loops

A **loop** is a shorthand way of asking the computer to do a certain calculation several times, usually with a slight difference between each repetition. For example, consider the following code chunk:

```
for(i in 1:5){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

Here, we stepped through the numbers 1 to 5, and printed each one out.

We need not always step through numbers in order. A loop can loop through any vector you provide. For example,

```r
my_vec <- c("Mary", "had", "a", "little", "lamb")

for(i in my_vec){
  print(i)
}
```

```
## [1] "Mary"
## [1] "had"
## [1] "a"
## [1] "little"
## [1] "lamb"
```

The loops you see above are called **for loops**, because they do a certain process *for* each value.

Another type of loop is a **while loop**, which keeps repeating itself as long as a certain condition is met. For example,

```r
i <- 1

while(i < 5){
  print(i)
  i <- i + 5
}
```

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

Be careful using while loops! Consider the code below (which has been set NOT to run in this document). What would happen if we ran it?

```r
i <- 1

while(i < 5){
  print(i)
}
```

apply

An alternative to loops in R is a set of functions based on `apply()`. There are a few slightly different options for this approach:

- 'lapply()' does a calculation for every element of a list, vector, or matrix, and returns a list as the answer.
- 'sapply()' does a calculation for every element of a vector or matrix, and returns a vector as the answer.
- 'apply()' does a calculation for every row (1) or column (2) of a matrix or data frame, and returns a matrix or data frame as the answer

Suppose we want to round all our variables in `mtcars` to the nearest whole number. The examples below will accomplish slightly different things.

```r
lapply(mtcars, round)
```

```
## $mpg
##  [1] 21 21 23 21 19 18 14 24 23 19 18 16 17 15 10 10 15 32 30 34 22 16 15
## [24] 13 19 27 26 30 16 20 15 21
##
## $cyl
##  [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

```
##
## $disp
##  [1] 160 160 108 258 360 225 360 147 141 168 168 276 276 276 472 460 440
## [18]  79  76  71 120 318 304 350 400  79 120  95 351 145 301 121
##
## $hp
##  [1] 110 110  93 110 175 105 245  62  95 123 123 180 180 180 205 215 230
## [18]  66  52  65  97 150 150 245 175  66  91 113 264 175 335 109
##
## $drat
##  [1] 4 4 4 3 3 3 3 4 4 4 4 3 3 3 3 3 3 4 5 4 4 3 3 4 3 4 4 4 4 4 4 4
##
## $wt
##  [1] 3 3 2 3 3 3 4 3 3 3 3 4 4 4 5 5 5 2 2 2 2 4 3 4 4 2 2 2 3 3 4 3
##
## $qsec
##  [1] 16 17 19 19 17 20 16 20 23 18 19 17 18 18 18 18 17 19 19 20 20 17 17
## [24] 15 17 19 17 17 14 16 15 19
##
## $vs
##  [1] 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 0 0 0 1
##
## $am
##  [1] 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1
##
## $gear
##  [1] 4 4 4 3 3 3 3 4 4 4 4 3 3 3 3 3 3 4 4 4 3 3 3 3 3 4 5 5 5 5 5 4
##
## $carb
##  [1] 4 4 1 1 2 1 4 2 2 4 4 3 3 3 4 4 4 1 2 1 1 2 2 4 2 1 2 2 4 6 8 2
```

```
sapply(mtcars, round)
```

```
##       mpg cyl disp  hp drat wt qsec vs am gear carb
##  [1,]  21   6  160 110    4  3   16  0  1    4    4
##  [2,]  21   6  160 110    4  3   17  0  1    4    4
##  [3,]  23   4  108  93    4  2   19  1  1    4    1
##  [4,]  21   6  258 110    3  3   19  1  0    3    1
##  [5,]  19   8  360 175    3  3   17  0  0    3    2
##  [6,]  18   6  225 105    3  3   20  1  0    3    1
##  [7,]  14   8  360 245    3  4   16  0  0    3    4
##  [8,]  24   4  147  62    4  3   20  1  0    4    2
##  [9,]  23   4  141  95    4  3   23  1  0    4    2
## [10,]  19   6  168 123    4  3   18  1  0    4    4
## [11,]  18   6  168 123    4  3   19  1  0    4    4
## [12,]  16   8  276 180    3  4   17  0  0    3    3
## [13,]  17   8  276 180    3  4   18  0  0    3    3
## [14,]  15   8  276 180    3  4   18  0  0    3    3
## [15,]  10   8  472 205    3  5   18  0  0    3    4
## [16,]  10   8  460 215    3  5   18  0  0    3    4
## [17,]  15   8  440 230    3  5   17  0  0    3    4
## [18,]  32   4   79  66    4  2   19  1  1    4    1
## [19,]  30   4   76  52    5  2   19  1  1    4    2
## [20,]  34   4   71  65    4  2   20  1  1    4    1
## [21,]  22   4  120  97    4  2   20  1  0    3    1
```

```
## [22,]  16   8  318 150    3  4   17  0  0    3    2
## [23,]  15   8  304 150    3  3   17  0  0    3    2
## [24,]  13   8  350 245    4  4   15  0  0    3    4
## [25,]  19   8  400 175    3  4   17  0  0    3    2
## [26,]  27   4   79  66    4  2   19  1  1    4    1
## [27,]  26   4  120  91    4  2   17  0  1    5    2
## [28,]  30   4   95 113    4  2   17  1  1    5    2
## [29,]  16   8  351 264    4  3   14  0  1    5    4
## [30,]  20   6  145 175    4  3   16  0  1    5    6
## [31,]  15   8  301 335    4  4   15  0  1    5    8
## [32,]  21   4  121 109    4  3   19  1  1    4    2
```

```
apply(mtcars, 1, round)
```

```
##      Mazda RX4 Mazda RX4 Wag Datsun 710 Hornet 4 Drive Hornet Sportabout
## mpg         21            21         23             21                19
## cyl          6             6          4              6                 8
## disp       160           160        108            258               360
## hp         110           110         93            110               175
## drat         4             4          4              3                 3
## wt           3             3          2              3                 3
## qsec        16            17         19             19                17
## vs           0             0          1              1                 0
## am           1             1          1              0                 0
## gear         4             4          4              3                 3
## carb         4             4          1              1                 2
##      Valiant Duster 360 Merc 240D Merc 230 Merc 280 Merc 280C Merc 450SE
## mpg       18         14        24       23       19        18         16
## cyl        6          8         4        4        6         6          8
## disp     225        360       147      141      168       168        276
## hp       105        245        62       95      123       123        180
## drat       3          3         4        4        4         4          3
## wt         3          4         3        3        3         3          4
## qsec      20         16        20       23       18        19         17
## vs         1          0         1        1        1         1          0
## am         0          0         0        0        0         0          0
## gear       3          3         4        4        4         4          3
## carb       1          4         2        2        4         4          3
##      Merc 450SL Merc 450SLC Cadillac Fleetwood Lincoln Continental
## mpg          17          15                10                  10
## cyl           8           8                 8                   8
## disp        276         276               472                 460
## hp          180         180               205                 215
## drat          3           3                 3                   3
## wt            4           4                 5                   5
## qsec         18          18                18                  18
## vs            0           0                 0                   0
## am            0           0                 0                   0
## gear          3           3                 3                   3
## carb          3           3                 4                   4
##      Chrysler Imperial Fiat 128 Honda Civic Toyota Corolla Toyota Corona
## mpg                15       32          30             34             22
## cyl                 8        4           4              4              4
## disp              440       79          76             71            120
## hp                230       66          52             65             97
```

```
## drat                 3        4           5              4             4
## wt                   5        2           2              2             2
## qsec                17       19          19             20            20
## vs                   0        1           1              1             1
## am                   0        1           1              1             0
## gear                 3        4           4              4             3
## carb                 4        1           2              1             1
##      Dodge Challenger AMC Javelin Camaro Z28 Pontiac Firebird Fiat X1-9
## mpg                16          15         13               19        27
## cyl                 8           8          8                8         4
## disp              318         304        350              400        79
## hp                150         150        245              175        66
## drat                3           3          4                3         4
## wt                  4           3          4                4         2
## qsec               17          17         15               17        19
## vs                  0           0          0                0         1
## am                  0           0          0                0         1
## gear                3           3          3                3         4
## carb                2           2          4                2         1
##      Porsche 914-2 Lotus Europa Ford Pantera L Ferrari Dino Maserati Bora
## mpg             26           30             16           20            15
## cyl              4            4              8            6             8
## disp           120           95            351          145           301
## hp              91          113            264          175           335
## drat             4            4              4            4             4
## wt               2            2              3            3             4
## qsec            17           17             14           16            15
## vs               0            1              0            0             0
## am               1            1              1            1             1
## gear             5            5              5            5             5
## carb             2            2              4            6             8
##      Volvo 142E
## mpg          21
## cyl           4
## disp        121
## hp          109
## drat          4
## wt            3
## qsec         19
## vs            1
## am            1
## gear          4
## carb          2
```

```
apply(mtcars, 2, round)
```

```
##                   mpg cyl disp  hp drat wt qsec vs am gear carb
## Mazda RX4          21   6  160 110    4  3   16  0  1    4    4
## Mazda RX4 Wag      21   6  160 110    4  3   17  0  1    4    4
## Datsun 710         23   4  108  93    4  2   19  1  1    4    1
## Hornet 4 Drive     21   6  258 110    3  3   19  1  0    3    1
## Hornet Sportabout  19   8  360 175    3  3   17  0  0    3    2
## Valiant            18   6  225 105    3  3   20  1  0    3    1
## Duster 360         14   8  360 245    3  4   16  0  0    3    4
## Merc 240D          24   4  147  62    4  3   20  1  0    4    2
```

```
## Merc 230             23   4  141  95    4  3   23  1  0    4    2
## Merc 280             19   6  168 123    4  3   18  1  0    4    4
## Merc 280C            18   6  168 123    4  3   19  1  0    4    4
## Merc 450SE           16   8  276 180    3  4   17  0  0    3    3
## Merc 450SL           17   8  276 180    3  4   18  0  0    3    3
## Merc 450SLC          15   8  276 180    3  4   18  0  0    3    3
## Cadillac Fleetwood   10   8  472 205    3  5   18  0  0    3    4
## Lincoln Continental  10   8  460 215    3  5   18  0  0    3    4
## Chrysler Imperial    15   8  440 230    3  5   17  0  0    3    4
## Fiat 128             32   4   79  66    4  2   19  1  1    4    1
## Honda Civic          30   4   76  52    5  2   19  1  1    4    2
## Toyota Corolla       34   4   71  65    4  2   20  1  1    4    1
## Toyota Corona        22   4  120  97    4  2   20  1  0    3    1
## Dodge Challenger     16   8  318 150    3  4   17  0  0    3    2
## AMC Javelin          15   8  304 150    3  3   17  0  0    3    2
## Camaro Z28           13   8  350 245    4  4   15  0  0    3    4
## Pontiac Firebird     19   8  400 175    3  4   17  0  0    3    2
## Fiat X1-9            27   4   79  66    4  2   19  1  1    4    1
## Porsche 914-2        26   4  120  91    4  2   17  0  1    5    2
## Lotus Europa         30   4   95 113    4  2   17  1  1    5    2
## Ford Pantera L       16   8  351 264    4  3   14  0  1    5    4
## Ferrari Dino         20   6  145 175    4  3   16  0  1    5    6
## Maserati Bora        15   8  301 335    4  4   15  0  1    5    8
## Volvo 142E           21   4  121 109    4  3   19  1  1    4    2
```

Explain why the following would give an error.

```
apply(mtcars, round)
```

The role of loops in R

As a rule, one should always avoid using loops in R if at all possible.

Most functions in R are **vectorized}, meaning the calculation will automatically be applied to all the elements, if possible. For example, the following is the easiest way to round the variables in mtcars to the nearest whole number!

```
round(mtcars)
```

```
##                   mpg cyl disp  hp drat wt qsec vs am gear carb
## Mazda RX4          21   6  160 110    4  3   16  0  1    4    4
## Mazda RX4 Wag      21   6  160 110    4  3   17  0  1    4    4
## Datsun 710         23   4  108  93    4  2   19  1  1    4    1
## Hornet 4 Drive     21   6  258 110    3  3   19  1  0    3    1
## Hornet Sportabout  19   8  360 175    3  3   17  0  0    3    2
## Valiant            18   6  225 105    3  3   20  1  0    3    1
## Duster 360         14   8  360 245    3  4   16  0  0    3    4
## Merc 240D          24   4  147  62    4  3   20  1  0    4    2
## Merc 230           23   4  141  95    4  3   23  1  0    4    2
## Merc 280           19   6  168 123    4  3   18  1  0    4    4
## Merc 280C          18   6  168 123    4  3   19  1  0    4    4
## Merc 450SE         16   8  276 180    3  4   17  0  0    3    3
## Merc 450SL         17   8  276 180    3  4   18  0  0    3    3
## Merc 450SLC        15   8  276 180    3  4   18  0  0    3    3
## Cadillac Fleetwood 10   8  472 205    3  5   18  0  0    3    4
## Lincoln Continental 10  8  460 215    3  5   18  0  0    3    4
```

```
## Chrysler Imperial    15   8  440 230    3  5   17  0  0    3    4
## Fiat 128             32   4   79  66    4  2   19  1  1    4    1
## Honda Civic          30   4   76  52    5  2   19  1  1    4    2
## Toyota Corolla       34   4   71  65    4  2   20  1  1    4    1
## Toyota Corona        22   4  120  97    4  2   20  1  0    3    1
## Dodge Challenger     16   8  318 150    3  4   17  0  0    3    2
## AMC Javelin          15   8  304 150    3  3   17  0  0    3    2
## Camaro Z28           13   8  350 245    4  4   15  0  0    3    4
## Pontiac Firebird     19   8  400 175    3  4   17  0  0    3    2
## Fiat X1-9            27   4   79  66    4  2   19  1  1    4    1
## Porsche 914-2        26   4  120  91    4  2   17  0  1    5    2
## Lotus Europa         30   4   95 113    4  2   17  1  1    5    2
## Ford Pantera L       16   8  351 264    4  3   14  0  1    5    4
## Ferrari Dino         20   6  145 175    4  3   16  0  1    5    6
## Maserati Bora        15   8  301 335    4  4   15  0  1    5    8
## Volvo 142E           21   4  121 109    4  3   19  1  1    4    2
```

Sometimes, though, you will want to do something complex enough that you need to force the repetition. In these cases, `apply()` is a better approach, and/or various Tidy tools you will learn later on.

Typically, a *for loop* is only used in R for *simulation*: doing a calculation many times with different randomized data. A *while loop* is usually only used for estimation: you do a calculation until your answer is consistent to within a certain error. More on both of these later in the course!

Conditionals

As you start writing loops and longer code processes, you will find that you sometimes need to check if something is true or not before you proceed. This is accomplished with `if()` and `else()` statements. `if()` statements take a `TRUE/FALSE` condition as input, and proceed only if the result is `TRUE`. For example,

```
for(i in 1:5){

  if(i == 3){

    print(i)

  }

}
```

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

The `else()` and `else if()` options, which always must come after an `if()`, will let you specify multiple things to check:

```
for(i in 1:5){

  if(i == 3){

    print(i)

  }else if(i %% 2 == 0){
    print(paste(i, "is an even number!"))
  }else{
    print("Boring number.")
  }
```

```
}
```

```
## [1] "Boring number."
## [1] "2 is an even number!"
## [1] 3
## [1] "4 is an even number!"
## [1] "Boring number."
```

`if()` statements are not very common in basic data analysis; you are more likely to use these if you are doing fancier coding. However, they can come in handy sometimes.

Try to think through the following code, and guess what the output would be.

```
for(i in 1:length(mtcars)){

  if(mtcars$am[i] == 0){

    my_mpg <- mtcars$mpg[i]
    print(paste("Automatic car with", my_mpg, "miles per gallon."))

  }

}
```

A word about formatting

Although "white space" - spaces, tabs, and new lines - does not matter to R, it is important to format your code in a way that is easy to read. When using loops and conditionals, always make sure to "tab" everything inside the loop or conditional, so that it is easy to visually see which tasks are being repeated and/or done only under certain conditions.

Functions

As we have seen, R is made up of **functions**, which take **arguments**, do calculations behind the scenes, and return certain output. An important skill in R coding is to write your own functions. For example, consider the following:

```
say_hi <- function(name){

  print(paste("Hello, ", name))

}

say_hi("George")
```

```
## [1] "Hello,  George"
```

Here, we wrote our own function, which we chose to name `say_hi`. This function takes a string (`name`) as an argument, and prints out a nice hello message. Once we have run the above code, we can use this function any time:

```
say_hi("Martha")
```

```
## [1] "Hello,  Martha"
```

Be careful, though - functions can do some strange things if you give them unexpected input!

```
say_hi(mtcars)
```

```
##  [1] "Hello,  c(21, 21, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8, 16.4, 17.3, 15.2, 10.4
##  [2] "Hello,  c(6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8, 8, 8, 8, 4, 4, 4, 8
##  [3] "Hello,  c(160, 160, 108, 258, 360, 225, 360, 146.7, 140.8, 167.6, 167.6, 275.8, 275.8, 275.8, 4
##  [4] "Hello,  c(110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180, 205, 215, 230, 66
##  [5] "Hello,  c(3.9, 3.9, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92, 3.07, 3.07, 3.07, 2.9
##  [6] "Hello,  c(2.62, 2.875, 2.32, 3.215, 3.44, 3.46, 3.57, 3.19, 3.15, 3.44, 3.44, 4.07, 3.73, 3.78
##  [7] "Hello,  c(16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20, 22.9, 18.3, 18.9, 17.4, 17.6, 18
##  [8] "Hello,  c(0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0
##  [9] "Hello,  c(1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1
## [10] "Hello,  c(4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 4, 5, 5, 5
## [11] "Hello,  c(4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2, 2, 4, 2, 1, 2, 2, 4
```

Optional arguments

When writing a function, you can also specify an optional argument. This is accomplished by supplying a default value for a certain argument: if the argument is supplied, the function uses the given value, and if not, it uses the default. For example,

```
say_hi <- function(name, greeting = "Hello"){

  print(paste(greeting, ", ", name))

}

say_hi("George")
```

```
## [1] "Hello ,  George"
```

```
say_hi("George", greeting = "Yo")
```

```
## [1] "Yo ,  George"
```

Functions and repetition

Functions can be particularly useful when you want to do a complex process several times. You can write a quick function outlining the steps, and then use loops or `apply()` to repeat your process. For example,

```
report_mpg <- function(mpg){

    print(paste("This car has", round(mpg), "miles per gallon."))

}

sapply(mtcars$mpg[1:6], report_mpg)
```

```
## [1] "This car has 21 miles per gallon."
## [1] "This car has 21 miles per gallon."
## [1] "This car has 23 miles per gallon."
## [1] "This car has 21 miles per gallon."
## [1] "This car has 19 miles per gallon."
## [1] "This car has 18 miles per gallon."
```

```
## [1] "This car has 21 miles per gallon." "This car has 21 miles per gallon."
## [3] "This car has 23 miles per gallon." "This car has 21 miles per gallon."
## [5] "This car has 19 miles per gallon." "This car has 18 miles per gallon."
```

Sourcing scripts

As you write your own functions, for organizational reasons you may wish to store the “source" code separately from your nice analysis in R Markdown or similar. To do this, you can create a separate file, which you use to load in your functions before using them in the analysis. For example,

```
source("./say_hi.R")

say_hi("Bob")
```

```
## [1] "Well hello there,  Bob"
```

What do you think are the contents of the file `say_hi.R`?