Handout 5

Put your name here!

In this handout, we will learn how to calculate a single loop. In the next week, we will discuss nested loops.

Topics and Concepts Covered

1. Building a loop
    - Getting the inner part correct
    - Adding a counter
    - Adding a container
2. *Declaring*: Creating an object with value NULL, such that we can add elements to it throughout a loop

R Command Covered

- Running a loop with `for`
- Building matrices by binding columns with `cbind` or binding rows with `rbind`
- Rounding numbers with `round`
- Using the command `%in%` to test which elements of one vector are contained in another
- Using `sort` to sort the elements of a vector
- Using `unique` to return the unique elements of a vector

Before beginning this handout, Do not forget to set your working directory!

An Introduction to Loops

In this course, we will be using loops in order to create parsimonious code for conducting large scale calculations. Loops provide a means for performing the same operation over and over, through various subsets of the data. By *loop*, I am actually referring to several distinct pieces of code:

- The expression `for`, which establishes the loop
- A *counter* variable that controls the flow of the loop
- A *container*, for collecting output from the loop

Let's start with a simple loop:

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

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

This loop has a `for` command, and the counter is `i.count`. R interprets this code as doing the following:

1. Set `i.count` to 1.
2. Do whatever is in the brackets.
3. Set `i.count` to the next value.
4. If `i.count` is less than or equal to 5, go back to (2).

Just like when we reserve the variable `y` for an outcome and `x` for an independent variable, we generally reserve the letter `i` for the counter in a loop. I generally prefer informative labels for counters, so we will create counters named `i.name`, where `name` is some informative label.

Now, let's say that we wanted to save the output from the loop in a container. To do so, we must create the container variable. Loops are the only place in R where we will be declaring a variable. So, let's declare and add the container:

```
squares <- NULL
for (i.squares in 1:5) {
  squares[i.squares] <- i.squares^2
}
squares
```

```
[1]  1  4  9 16 25
```

The line `squares <- NULL` is the command we use to declare the variable `squares`, an empty container that will gather our output. Finally, note that upon completing the loop, the counter maintains its last value:

```
i.squares
```

```
[1] 5
```

Coding tip

The container must be declared *outside* the loop that is generating the output. If we put the container inside the loop, it would be reset to `NULL` at each iteration!

Example: The Fibonacci Numbers

The Fibonacci numbers are a sequence of numbers derived recursively, with some surprising and beautiful links to geometry. The first two numbers in the sequence are 1, and each subsequent element is the sum of the previous two. So, the first several elements are:

$$\text{Fib}_1 = 1 \tag{1}$$
$$\text{Fib}_2 = 1 \tag{2}$$
$$\text{Fib}_3 = \text{Fib}_1 + \text{Fib}_2 = 1 + 1 = 2 \tag{3}$$
$$\text{Fib}_4 = \text{Fib}_2 + \text{Fib}_3 = 1 + 2 = 3 \tag{4}$$
$$\text{Fib}_5 = \text{Fib}_3 + \text{Fib}_4 = 2 + 3 = 5 \tag{5}$$

If we wanted to create the first 1,000 Fibonacci numbers through a loop, we would need:

- A counter showing *which* Fibonacci number we were at.
- A `for` command, telling R to create the numbers.
- A container for containing the sequence.

In R, we would use the following code:

```
fibonacci <- NULL
fibonacci[1] <- 1
fibonacci[2] <- 1
for (i.fibonacci in 3:1000) {
  fibonacci[i.fibonacci] <- fibonacci[i.fibonacci - 1] + fibonacci[i.fibonacci - 2]
}
fibonacci[1:30]
```

```
 [1]      1      1      2      3      5      8     13     21     34     55
[11]     89    144    233    377    610    987   1597   2584   4181   6765
[21]  10946  17711  28657  46368  75025 121393 196418 317811 514229 832040
```

Now, we can show mathematically that the ratio of two consecutive Fibonacci numbers approaches $(\sqrt{5}+1)/2$. We'll be using

```
fibonacci <- NULL
ratio.fibonacci <- NULL
fibonacci[1] <- 1
fibonacci[2] <- 1
for (i.fibonacci in 3:1000) {
  fibonacci[i.fibonacci] <- fibonacci[i.fibonacci - 1] + fibonacci[i.fibonacci - 2]
  ratio.fibonacci[i.fibonacci] <- fibonacci[i.fibonacci] / fibonacci[i.fibonacci - 1]
}
```

Let's check:

```
(5^.5 + 1)/2
```

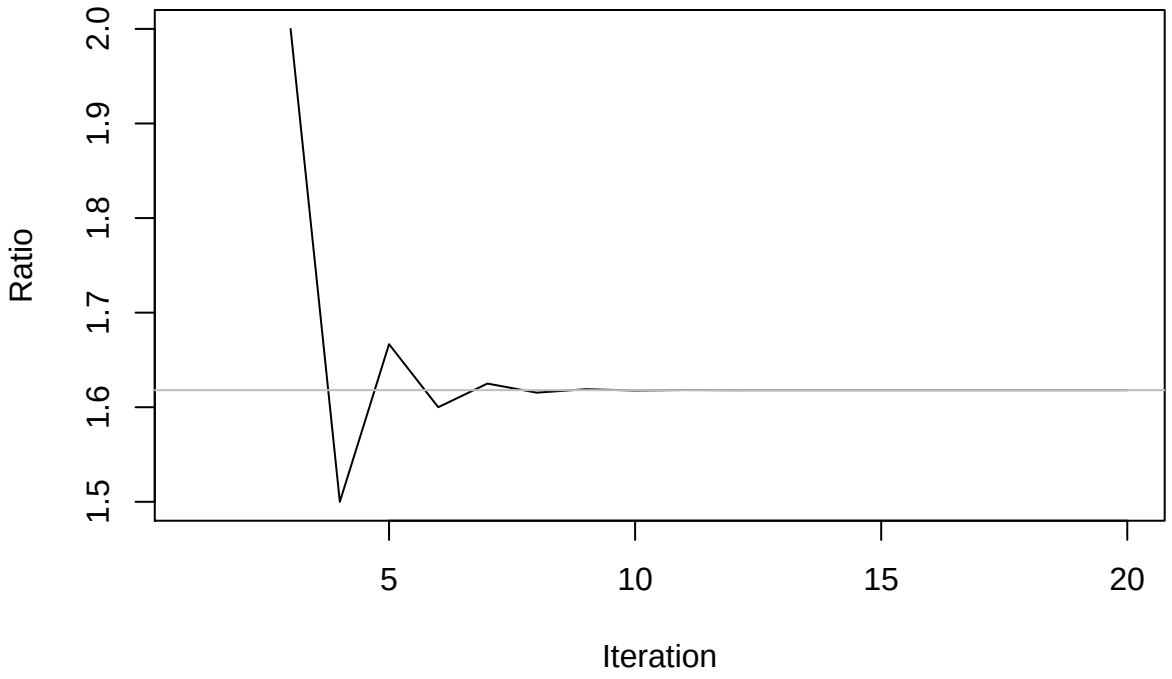
```
[1] 1.618034
```

```
ratio.fibonacci[1:20]
```

```
 [1]       NA       NA 2.000000 1.500000 1.666667 1.600000 1.625000
 [8] 1.615385 1.619048 1.617647 1.618182 1.617978 1.618056 1.618026
[15] 1.618037 1.618033 1.618034 1.618034 1.618034 1.618034
```

And watch it converge:

```
plot(1:20, ratio.fibonacci[1:20], type = "l",
    xlab = "Iteration", ylab = "Ratio")
abline(h = (5^.5 + 1)/2, col = "gray")
```



Writing a Loop, in Practice

In this section, we are going to focus on a subset of the American National Election Study (ANES) cumulative data file. The ANES survey is run on a roughly biennial basis, providing a reliable estimate of public opinion, partisanship, and public knowledge from the mid 1950's through today.

The subset we will be exploring contains all years that have information on how respondents assess the statement

> This country would be better off if we just stayed home and did not concern ourselves with problems in other parts of the world.

This variable, `isolate`, is coded as -1, 0, and 1, depending on whether the respondent disagrees, has no opinion, or agrees with this statement. So, a score of 1 for `isolate` indicates the respondent is more isolationist; a score of 0 indicates they are more interventionist.

We analyze 23,287 responses, with observations in the years 1956, 1958, 1960, 1968, 1972, 1976, 1980, 1984, 1986, 1988, 1990, 1992, 1994, 1996, 1998, 2000, and 2004. We will focus on two covariates: `partyid`, which is party identification on a scale from 1-7 (Strong Democrat, Weak Democrat, Independent-Democrat, Independent, Independent-Republican, Weak Republican, Strong Republican), and `year`, for the year the survey was administered.

Other variables in the dataset include `age`, in years; `attend`, church attendance on an ordinal scale of `1-5`, in terms of decreasing attendance; `south` as a regional indicator; `income`, which takes on five values; `female`, an indicator for gender; and `minority`, an indicator variable for whether the respondent is white or not.

Below, we are going to make a more involved loop. The loop is going to calculate the mean level of isolationist tendency (-1, 0, 1) for each level of party ID (1, 2, 3, ... 7) by year.

Before beginning, first read in the data:

```
Data.isolate <- read.table("data/ANES_isolate.tsv")
head(Data.isolate)
```

```
     isolate year partyid age attend south income female minority
3701      -1 1956       4  43      4     0      2      1        1
3702      -1 1956       2  35      4     0      3      0        0
3703       1 1956       1  55      1     1      2      1        0
3704       0 1956       3  23      4     0      3      0        0
3705      -1 1956       1  23      1     0      4      0        0
3706      -1 1956       2  24      4     0      3      1        0
```

```
summary(Data.isolate)
```

```
    isolate             year         partyid          age       
 Min.   :-1.0000   Min.   :1956   Min.   :1.000   Min.   :17.00  
 1st Qu.:-1.0000   1st Qu.:1972   1st Qu.:2.000   1st Qu.:31.00  
 Median :-1.0000   Median :1986   Median :3.000   Median :43.00  
 Mean   :-0.4776   Mean   :1982   Mean   :3.667   Mean   :45.23  
 3rd Qu.: 0.0000   3rd Qu.:1994   3rd Qu.:6.000   3rd Qu.:58.00  
 Max.   : 1.0000   Max.   :2004   Max.   :7.000   Max.   :99.00  
     attend          south            income          female      
 Min.   :1.000   Min.   :0.0000   Min.   :1.000   Min.   :0.0000  
 1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:0.0000  
 Median :3.000   Median :0.0000   Median :3.000   Median :1.0000  
 Mean   :2.964   Mean   :0.3376   Mean   :2.896   Mean   :0.5471  
 3rd Qu.:4.000   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:1.0000  
 Max.   :5.000   Max.   :1.0000   Max.   :5.000   Max.   :1.0000  
    minority
```

```
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.1792
 3rd Qu.:0.0000
 Max.   :1.0000
```

Writing a Loop: Get the Inner Part Correct

The first step to successfully writing a loop is producing code that works when the counter is set to 1. So, we are going to create a counter, $i.year$, and set it equal to 1

```
i.year <- 1
```

Next, we are going to create a vector that contains a sorted list of all unique years in the data.

```
all.years <- sort(unique(Data.isolate$year))
all.years
```

```
 [1] 1956 1958 1960 1968 1972 1976 1980 1984 1986 1988 1990 1992 1994 1996
[15] 1998 2000 2004
```

This variable, `all.years` will tell us which year to look at.

Next, we are going to subset the data, and then use `tapply` to calculate the relevant means:

```
Data.sub <- Data.isolate[Data.isolate$year == all.years[i.year], ]
tapply(Data.sub$isolate, INDEX = Data.sub$partyid, FUN = mean)
```

```
         1          2          3          4          5          6
-0.2674419 -0.3509235 -0.3557692 -0.3541667 -0.4511278 -0.3956522
         7
-0.3483607
```

It appears that, in 1956, Strong Democrats (1's) were the most interventionist and Independent-Republicans were the least interventionist (5's).

Writing a Loop: Place the Inner Part in a Loop

Now, we are going to place the code from above in a loop. The loop is going to use the counter `i.year`, which will go from 1 to 'length(all.years):

```
all.years <- sort(unique(Data.isolate$year))
n.years <- length(all.years) # For length of counter

for (i.year in 1:n.years) {
    Data.sub <- Data.isolate[Data.isolate$year == all.years[i.year], ]
    tapply(Data.sub$isolate, INDEX = Data.sub$partyid, FUN = mean)
}
```

At this point, if you want to see if your loop is working, you can place the `tapply` within a `plot`.

Also, note that we are creating a temporary data frame, `Data.sub`, that gets written over at each iteration of the loop.

Writing a Loop: Adding a Container

So far, we have gotten the internal component of a loop working. We have also successfully placed it within a `for` loop. Finally, we are going to generate a container in order to calculate the calculations that we need

from the loop. Let's call the container `isolate.byyear`

```
isolate.byyear <- NULL
all.years <- sort(unique(Data.isolate$year))
n.years <- length(all.years) # For length of counter

for (i.year in 1:n.years) {
    Data.sub <- Data.isolate[Data.isolate$year == all.years[i.year], ]
    isolate.byyear <- cbind(isolate.byyear,
        tapply(Data.sub$isolate, INDEX = Data.sub$partyid, FUN = mean))
}
```

To see what `cbind` does, take a look at the output:

```
round(isolate.byyear, 3)
```

```
    [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]   [,9]  [,10]
1 -0.267 -0.334 -0.418 -0.496 -0.578 -0.194 -0.628 -0.380 -0.199 -0.231
2 -0.351 -0.319 -0.422 -0.472 -0.575 -0.361 -0.622 -0.509 -0.401 -0.335
3 -0.356 -0.494 -0.548 -0.699 -0.509 -0.407 -0.551 -0.321 -0.309 -0.235
4 -0.354 -0.344 -0.561 -0.462 -0.455 -0.469 -0.471 -0.532 -0.328 -0.392
5 -0.451 -0.556 -0.603 -0.681 -0.740 -0.623 -0.638 -0.613 -0.524 -0.360
6 -0.396 -0.335 -0.443 -0.455 -0.605 -0.438 -0.719 -0.570 -0.466 -0.504
7 -0.348 -0.425 -0.561 -0.547 -0.705 -0.679 -0.699 -0.682 -0.642 -0.694
   [,11]  [,12]  [,13]  [,14]  [,15]  [,16]  [,17]
1 -0.281 -0.391 -0.381 -0.513 -0.727 -0.385 -0.481
2 -0.276 -0.384 -0.318 -0.448 -0.622 -0.512 -0.512
3 -0.202 -0.409 -0.379 -0.514 -0.556 -0.577 -0.635
4 -0.337 -0.285 -0.327 -0.346 -0.508 -0.276 -0.473
5 -0.618 -0.656 -0.409 -0.449 -0.533 -0.473 -0.538
6 -0.492 -0.616 -0.330 -0.485 -0.720 -0.490 -0.729
7 -0.687 -0.735 -0.579 -0.565 -0.719 -0.546 -0.868
```

Each row corresponds with a level of party ID, and each column is a year. `cbind` takes two arguments, and adds the second as a vector to the right of the first. `rbind` does the same, except it combines by adding a row to the bottom of the matrix.

Coding tip

`cbind(A, B)` returns the matrix `[A | B]`. `cbind(B, A)` returns the matrix `[B | A]`. These are not the same! Be careful as to how you are combining objects with `cbind` and `rbind`.

Finally, let's add informative column names to `isolate.byyear`

```
colnames(isolate.byyear) <- all.years
round(isolate.byyear, 3)
```

```
    1956   1958   1960   1968   1972   1976   1980   1984   1986   1988
1 -0.267 -0.334 -0.418 -0.496 -0.578 -0.194 -0.628 -0.380 -0.199 -0.231
2 -0.351 -0.319 -0.422 -0.472 -0.575 -0.361 -0.622 -0.509 -0.401 -0.335
3 -0.356 -0.494 -0.548 -0.699 -0.509 -0.407 -0.551 -0.321 -0.309 -0.235
4 -0.354 -0.344 -0.561 -0.462 -0.455 -0.469 -0.471 -0.532 -0.328 -0.392
5 -0.451 -0.556 -0.603 -0.681 -0.740 -0.623 -0.638 -0.613 -0.524 -0.360
6 -0.396 -0.335 -0.443 -0.455 -0.605 -0.438 -0.719 -0.570 -0.466 -0.504
7 -0.348 -0.425 -0.561 -0.547 -0.705 -0.679 -0.699 -0.682 -0.642 -0.694
    1990   1992   1994   1996   1998   2000   2004
1 -0.281 -0.391 -0.381 -0.513 -0.727 -0.385 -0.481
```

```
2 -0.276 -0.384 -0.318 -0.448 -0.622 -0.512 -0.512
3 -0.202 -0.409 -0.379 -0.514 -0.556 -0.577 -0.635
4 -0.337 -0.285 -0.327 -0.346 -0.508 -0.276 -0.473
5 -0.618 -0.656 -0.409 -0.449 -0.533 -0.473 -0.538
6 -0.492 -0.616 -0.330 -0.485 -0.720 -0.490 -0.729
7 -0.687 -0.735 -0.579 -0.565 -0.719 -0.546 -0.868
```

In order to do an initial analysis of this data, let's make three more vectors. The first two are the movement in Democrats and Republicans, each versus Independents. Next, we are going to create a variable that takes on a value of `TRUE` if a Democrat is President and `FALSE` otherwise. To do so, we will use the command `%in%`. The command is used as follows:

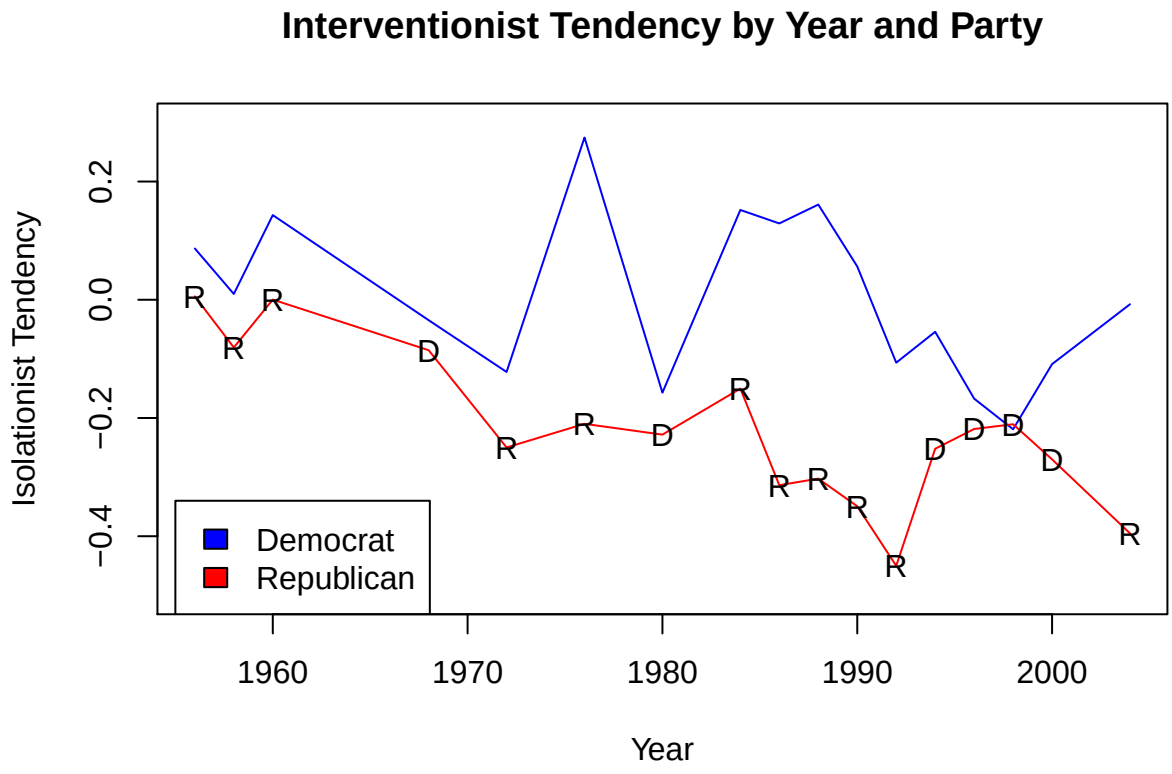
```
A %in% B
```

and returns a list of values `TRUE` or `FALSE` equal to the length of `A`, representing whether each element of `A` appears in `B`.

The code for the figure we are looking at is below:

```
demsvsindep <- isolate.byyear[1, ] - isolate.byyear[4, ]
repsvsindep <- isolate.byyear[7, ] - isolate.byyear[4, ]
dempres <- all.years %in% c(1968, 1980, 1994, 1996, 1998, 2000 )

plot(all.years,  demsvsindep,
     type = "l", ylim = c(-.5,.3), col = "blue",
     xlab = "Year", ylab = "Isolationist Tendency",
     main = "Interventionist Tendency by Year and Party")
lines(all.years, repsvsindep, col = "red")
points(all.years, repsvsindep, pch = ifelse(dempres, "D", "R"))
legend(x = 1955, y = -.34, legend = c("Democrat","Republican"),
       fill = c("blue","red"))
```



Though we cannot make conclusive claims from the data, it appears that since approximately 1970, Repub-

licans may be more interventionist when a co-partisan President is in office. We see no similar effect for Democrats, at least in this simple analysis. It is difficult to say whether this is just noise, though.

Precept Questions

Please note: loops commonly cause problems to students encountering them for the first time. If you are really struggling on these problems, put in a genuine effort, then talk to your preceptor. We can be flexible for this week.

Question 1

Repeat the figure above, but use four lines: a dashed red line for Republican men, a dashed blue line for Democratic men, a solid red line for Republican women, and a solid blue line for Democratic women. Label one of the lines with `D` and `R` for a Democratic and Republican President.

Make sure the figure is properly formatted (legend, labels, title, etc.)

Question 2

Repeat, but use Southerners and non-Southerners to split the data. Use dashed and solid red and blue lines to differentiate Southern and non-Southern Democrats and Republicans. Make sure the figure is properly formatted.

Question 3

Does splitting by gender or region give any insight?