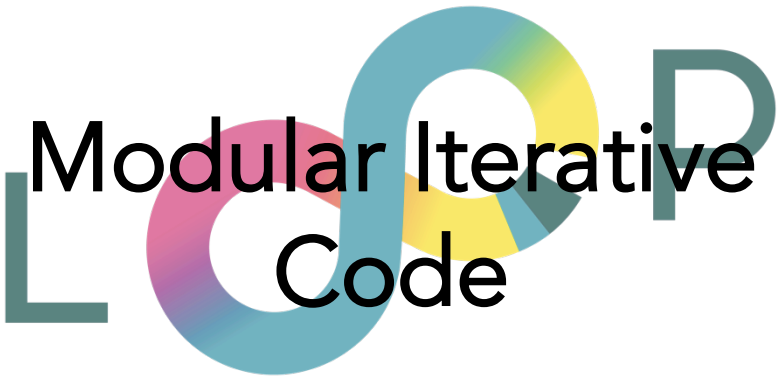

Modularity

- Many of you wrote two functions
  - `read_buoy()`
  - `format_buoy()`

```
buoy_1986 <- format_buoy(read_buoy(1986))
```

- This is the beginning of *MODULAR CODE*

Why write in Modules?

- Each function should do one thing, and do it well
  - Easier to debug
- Modular code means that individual functions can be used in many places
  - Flexible
- Modular code allows you to organize your thought process

Think Before You Code: Modularity Edition

1. Ask yourself, what do I want to do
2. Start at the highest level
3. This is your 'wrapper function'
4. Then break it down into sub-functions

Think Through Your Plan

```
# Read in a file based on year

# Standardize file format
# for between year differences

# Take that data frame and format it

# Return formatted data frame
```

Turn Your Plan Into a Function

```
make_buoy_data <- function(a_year){
  # Read in a file based on year

  # Standardize file format
  # for between year differences

  # Take that data frame and format it

  # Return formatted data frame
  return(buoy_data)
}
```

Write Functions for Each Step

```
make_buoy_data <- function(a_year){
   # Read in a file based on year
   buoy_data <- read_buoy(a_year)

   # Standardize file format
   # for between year differences
   buoy_data <- standardize_buoy(buoy_data )

   # Take that data frame and format it
   buoy_data <- format_buoy(buoy_data )

   # Return formatted data frame
   return(buoy_data)
}
```

read_buoy

```
read_buoy <- function(a_year){

  #make a filename
  filename <- paste0("./buoydata/44013_",
                     a_year, ".csv")

  #read in the file with that name
  a_buoy <- read.csv(filename,
              na.strings = c("99", "999"))

  #return the resulting data frame
  return(a_buoy)

}
```

standardize_buoy

- If year < 1999
  - Column name changes from YYYY or X.YY to YY
  - need to add 1900 to year
- If year > 2005, we need to eliminate row 2, and make sure our columns of interest are numeric

```
standardize_buoy <- function(buoydata){

  #make sure the year column is called YY
  names(buoydata) <- gsub("X.YY", "YY", names(buoydata))
  names(buoydata) <- gsub("YYYY", "YY", names(buoydata))

  #If the year > 2005, remove the second row
  if(as.numeric(buoydata$YY[1])>2005) buoydata <- buoydata[-1,]

  #Make sure our columns of interest are numeric
  buoydata <- buoydata %>%
    mutate(YY = as.numeric(YY),
           MM = as.numeric(MM),
           DD = as.numeric(DD),
           WTMP = as.numeric(WTMP),
           WVHT = as.numeric(WVHT))

  #make sure the years all are in the thousands
  if(buoydata$YY[1]<1999) buoydata$YY <- buoydata$YY + 1900

  return(buoydata)
}
```

format_buoy

- You've done this…

```
format_buoy <- function(buoydata){
  buoydata <- buoydata %>%
    select(YY, MM, DD, WVHT, WTMP) %>%
    rename(Year = YY,
         Month = MM,
         Day = DD,
         Wave_Height = WVHT,
         Temperature_c = WTMP) %>%
    group_by(Year, Month, Day) %>%
    summarise(Wave_Height = mean(Wave_Height, na.rm=T),
            Temperature_c = mean(Temperature_c, na.rm=T)) %>%
    ungroup()

  return(buoydata)
}
```

Your Wrapper Function!

```
make_buoy_data <- function(a_year){
   # Read in a file based on year
   buoy_data <- read_buoy(a_year)

   # Standardize file format
   # for between year differences
   buoy_data <- standardize_buoy(buoy_data )

   # Take that data frame and format it
   buoy_data <- format_buoy(buoy_data )

   # Return formatted data frame
   return(buoy_data)
}
```

Putting it All together

- OK, now I have a set of functions
- But I have a LOT of years!
- What now?

The for loop

```
for(i in 1:100){

    print(i)

}
```

The for loop

```
a_vec <- 201:500

for(i in 1:100){

    print(a_vec[i])

}
```

The for loop

```
a_vec <- rnorm(100)

for(i in 1:100){

    print(a_vec[i])

}
```

The for loop

```
a_vec <- 201:500
i_vec <- round(runif(50, min = 1, max = 300))

for(i in i_vec){

    print(a_vec[i])

}
```

Fibonacci

```
fib_vec <- c(1,1)

for(i in 3:20){
 fib_vec[i] <- fib_vec[i-1] + fib_vec[i-2]
}

fib_vec
```

The For Loop

- For loops let us iterate
- We have flexibility in values we use by creating a vector to iterate over

For Loops and Creating Data

1. Create a basic data frame from the first file
2. Use a loop to iterate over all other files
3. For each file, read it in and rbind it to the growing data frame

Example

```
my_data <- read.csv("100.csv")

for(a_num in 101:200){

  my_data <- rbind(my_data,
    read.csv(paste0(a_num, ".csv"))

}
```

Try it with the buoy data!

```
buoys <- make_buoy_data(1986)

for(one_year in 1987:2013){
    buoys <- rbind(buoys,
        make_buoy_data(one_year))

}
```

Now plot change over time!