

Doing things over and over again: Functions and **Loops**

For Loops

For each thing in some group of things, do something.

For Loops

For each thing in some group of things, do something.

```
for( <THING> in <GROUP OF THINGS> ){  
  <DO SOMETHING>  
}
```

For Loops

For each thing in some group of things, do something.

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

For Loops

For each thing in some group of things, do something.

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

i is a counter

For Loops

For each thing in some group of things, do something.

```
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

For each thing in some group of things, do something.

```
fruits <- c("apple", "banana", "cantaloupe")
```

```
for(i in 1:length(fruits)) {  
  print(fruits[i])  
}
```

square brackets to subset

We don't want to print *i* itself – we want to print the *i*th member of our vector.

For Loops

For each thing in some group of things, do something.

```
fruits <- c("apple", "banana", "canteloupe")
```

```
for(i in 1:length(fruits)) {  
  print(fruits[i])  
}
```


For Loops

For each thing in some group of things, do something.

```
fruits <- c("apple", "banana", "cantaloupe")
```

```
for(i in 1:length(fruits)){  
  print(fruits[i])  
}
```

```
[1] "apple"  
[1] "banana"  
[1] "cantaloupe"
```

For Loops

For each thing in some group of things, do something.

- Counting (indexing) starts at 1
- To save results, you have to pre-allocate the output – i.e., make an empty vector before your loop

Safer indexing

For each thing in some group of things, do something.

```
for(i in 1:length(fruits)) {  
  print(fruits[i])  
}
```

```
for(i in seq_along(fruits)) {  
  print(fruits[i])  
}
```

```
[1] "apple"  
[1] "banana"  
[1] "cantaloupe"
```

A few tips

1. Get code working outside a loop
2. Manually set “i” and make sure you get what you expect from the body of the loop

```
i <- 1  
print(fruits[i])
```

"apple"

3. Make sure it works on a different one too

```
i <- 2  
print(fruits[i])
```

"banana"

Pre-allocating memory for output

When you want to do more than just print things.

Of course, we'll start by just printing.

```
my_stns <- unique(wq_trimmed$station_code)

for(i in seq_along(my_stns)) {
  wq_sub <- wq_trimmed %>%
    filter(station_code == my_stns[i])

  print(my_stns[i])
  print(mean(wq_sub$sal, na.rm = TRUE))
}
```

Pre-allocating memory for output

When you want to do more than just print things.

Of course, we'll start by just printing.

```
my_stns <- unique(wq_trimmed$station_code)

for(i in seq_along(my_stns)) {
  wq_sub <- wq_trimmed %>%
    filter(station_code == my_stns[i])

  print(my_stns[i])
  print(mean(wq_sub$sal, na.rm = TRUE))
}
```

Pre-allocating memory for output

When you want to do more than just print things.

Of course, we'll start by just printing.

```
my_stns <- unique(wq_trimmed$station_code)

for(i in seq_along(my_stns)) {
  wq_sub <- wq_trimmed %>%
    filter(station_code == my_stns[i])

  print(my_stns[i])
  print(mean(wq_sub$sal, na.rm = TRUE))
}
```

[1]	"gndblwwq"
[1]	19.68066
[1]	"gtmpcwq"
[1]	20.20109
[1]	"kachdwq"
[1]	30.12615
[1]	"niwolwwq"
[1]	30.91247
[1]	"rkblhwq"
[1]	30.40225

Pre-allocating memory for output

What we *want* is probably a data frame that looks like this:

Station	Mean salinity
gndblwq	19.68
gtmpcwq	20.20
kachdwq	30.12
niwolwq	30.91
rkblhwq	30.40

Pre-allocating memory for output

Data frames make loops very, veeeeeeery slow.

So we're going to store output in vectors, then bind them together at the end.

```
stns_out <- rep("dummy_value", length(my_stns))  
mean_sal_out <- rep(0, length(my_stns))
```

```
stns_out; mean_sal_out
```

```
[1] "dummy_value" "dummy_value" "dummy_value"  
[4] "dummy_value" "dummy_value"
```

```
[1] 0 0 0 0 0
```

Pre-allocating memory for output

Now, to loop!

```
for(i in seq_along(my_stns)) {  
  wq_sub <- wq_trimmed %>%  
    filter(station_code == my_stns[i])  
  
  stns_out[i] <- my_stns[i]  
  mean_sal_out[i] <- mean(wq_sub$sal, na.rm = TRUE)  
}  
  
stns_out; mean_sal_out
```

Pre-allocating memory for output

Now, to loop!

```
for(i in seq_along(my_stns)) {  
  wq_sub <- wq_trimmed %>%  
    filter(station_code == my_stns[i])  
  
  stns_out[i] <- my_stns[i]  
  mean_sal_out[i] <- mean(wq_sub$sal, na.rm = TRUE)  
}  
  
stns_out; mean_sal_out
```

```
[1] "gndblwq" "gtmpcwq"  
[3] "kachdwq" "niwolwq"  
[5] "rkblhwq"  
  
[1] 19.68066 20.20109 30.12615  
[4] 30.91247 30.40225
```

Pre-allocating memory for output

```
for(i in seq_along(my_stns)) {  
  wq_sub <- wq_trimmed %>%  
    filter(station_code == my_stns[i])  
  
  stns_out[i] <- my_stns[i]  
  mean_sal_out[i] <- mean(wq_sub$sal, na.rm = TRUE)  
}
```

```
cbind(stns_out, mean_sal_out)
```

```
      stns_out  mean_sal_out  
[1,] "gndblwq" "19.6806614086868"  
[2,] "gtmpcwq" "20.2010917213249"  
[3,] "kachdwq" "30.1261500701366"  
[4,] "niwolwq" "30.91246666667409"  
[5,] "rkblhwq" "30.4022544404481"
```

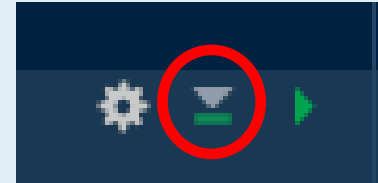
Pre-allocating memory for output

What did we do?

1. Created vectors to hold our output as it was generated
2. Replaced `print()` with a command to store our value in our desired output vector
3. Used “i” to specify *where* in the output vector our value needed to go for each iteration of the loop
4. Used `cbind()` to glue our output vectors together.

Your Turn

1. Navigate to “Your Turn 2” in the .Rmd file
2. Use this symbol to run all the prior code in the file:
3. Write a loop, in which you:
 - a. calculate mean temperature for each station
 - b. calculate mean depth for each station
 - c. bind these together, along with an identifying vector for “station_code”
4. Remember to set up output vectors *first*!



Your Turn: Answers

```
my_stns <- unique(wq_trimmed$station_code)
stns_out <- rep("stn", length(my_stns))
sal_out <- rep(0, length(my_stns))
depth_out <- rep(0, length(my_stns))

for(i in seq_along(my_stns)){
  stns_out[i] <- my_stns[i]

  wq_sub <- wq_trimmed %>%
    filter(station_code == my_stns[i])

  sal_out[i] <- mean(wq_sub$sal, na.rm = TRUE)
  depth_out[i] <- mean(wq_sub$depth, na.rm = TRUE)
}

cbind(stns_out, sal_out, depth_out)
```