Today

- Questions from homework (brief)?
- Linear regression reflection
- Regression workflow algorithm
- Repetition structures:
  - counter control using for

Linear regression reflection

- What was the goal of your analysis?
  - what question were you asking?
- How did the analysis answer the question?
  - what output (e.g. numerical, graphical) answered the question?

Workflow algorithm

- Combine your efforts to make a workflow algorithm for linear regression
  - diagram it, e.g. flowchart
- What did you learn as a group?
  - things that not everyone had

Workflow algorithms in DS

NSF master data science algorithm

```
Plan for data
Acquire data
Manage data
Analyze data
Infer from data
Report about data
```

R: `for` repetition structure

Most programming languages have a specialized structure for counter-controlled repetition (usually called "for")

```
for ( i in starti:endi ) {
    expression
}
```

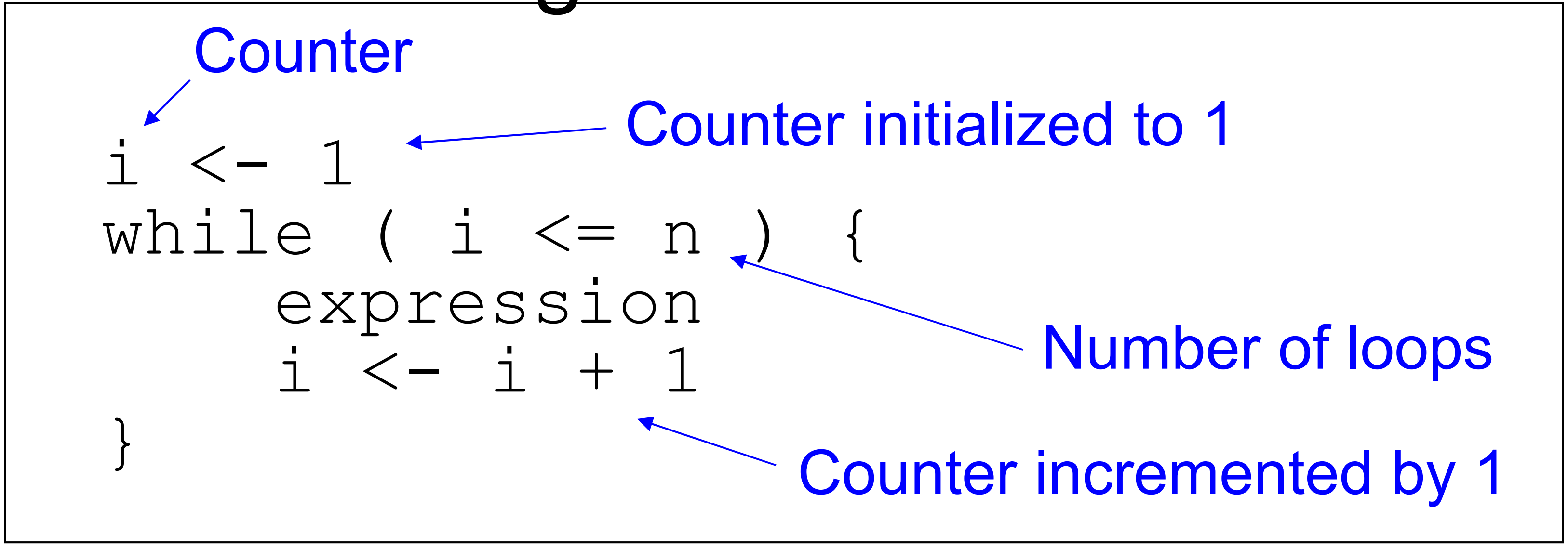
R: `for` repetition structure

Example

```
for ( i in 1:10 ) {
        j <- i * 2
        print(j)
}
```

What does this do?

The 4 components of counter control using `while` or `for`



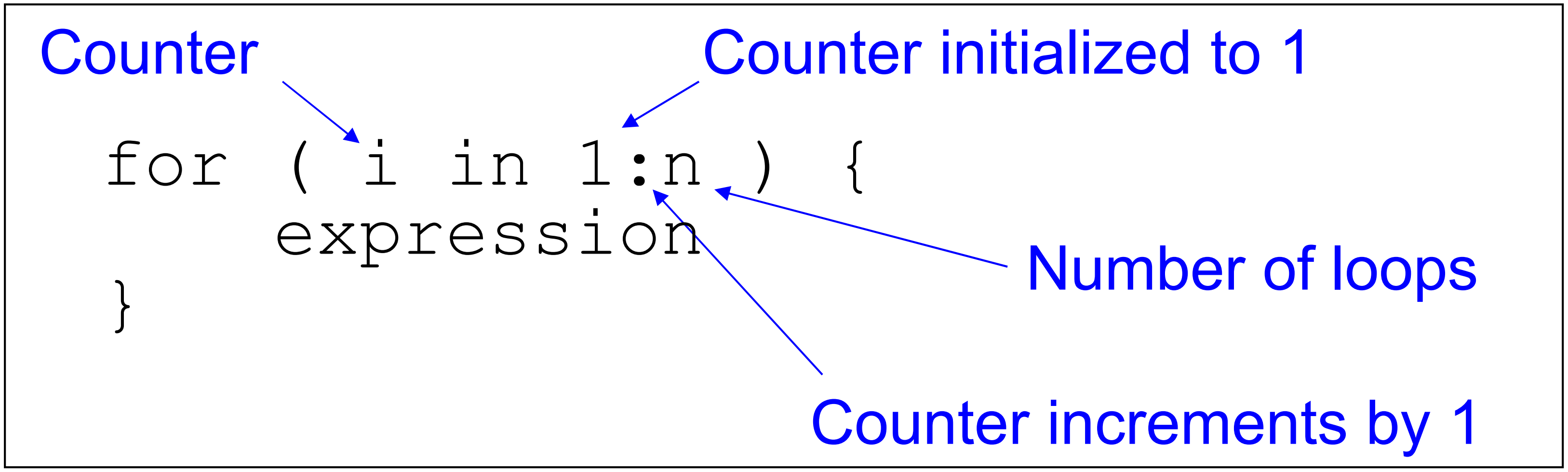
```
i <- 1
while ( i <= n ) {
    expression
    i <- i + 1
}
```

Counter

Counter initialized to 1

Number of loops

Counter incremented by 1



```
for ( i in 1:n ) {
    expression
}
```

Counter

Counter initialized to 1

Number of loops

Counter increments by 1

R: `for` repetition structure

Correct

```
for ( i in 1:n ) {
        expression
}
```

Incorrect

```
i <- 1
for ( i in 1:n ) {
        expression
        i <- i + 1
}
```

```
# Finds the number (y) that is the zth power of x

# Initialize parameters
x <- 3.2    #Any real number
z <- 2      #Any integer >= 0

# Initialize working variables
y <- 1
counter <- 1

# Processing phase
while ( counter <= z ) {
    y <- y * x
    counter <- counter + 1
}

# Termination phase
y
```

This code uses a while structure to do counter controlled repetition. Modify it to use a for counter-control structure instead.

Does it work for integer z = 0? Try it.

If not, fix it so integer z = 0 will work correctly with a for structure.

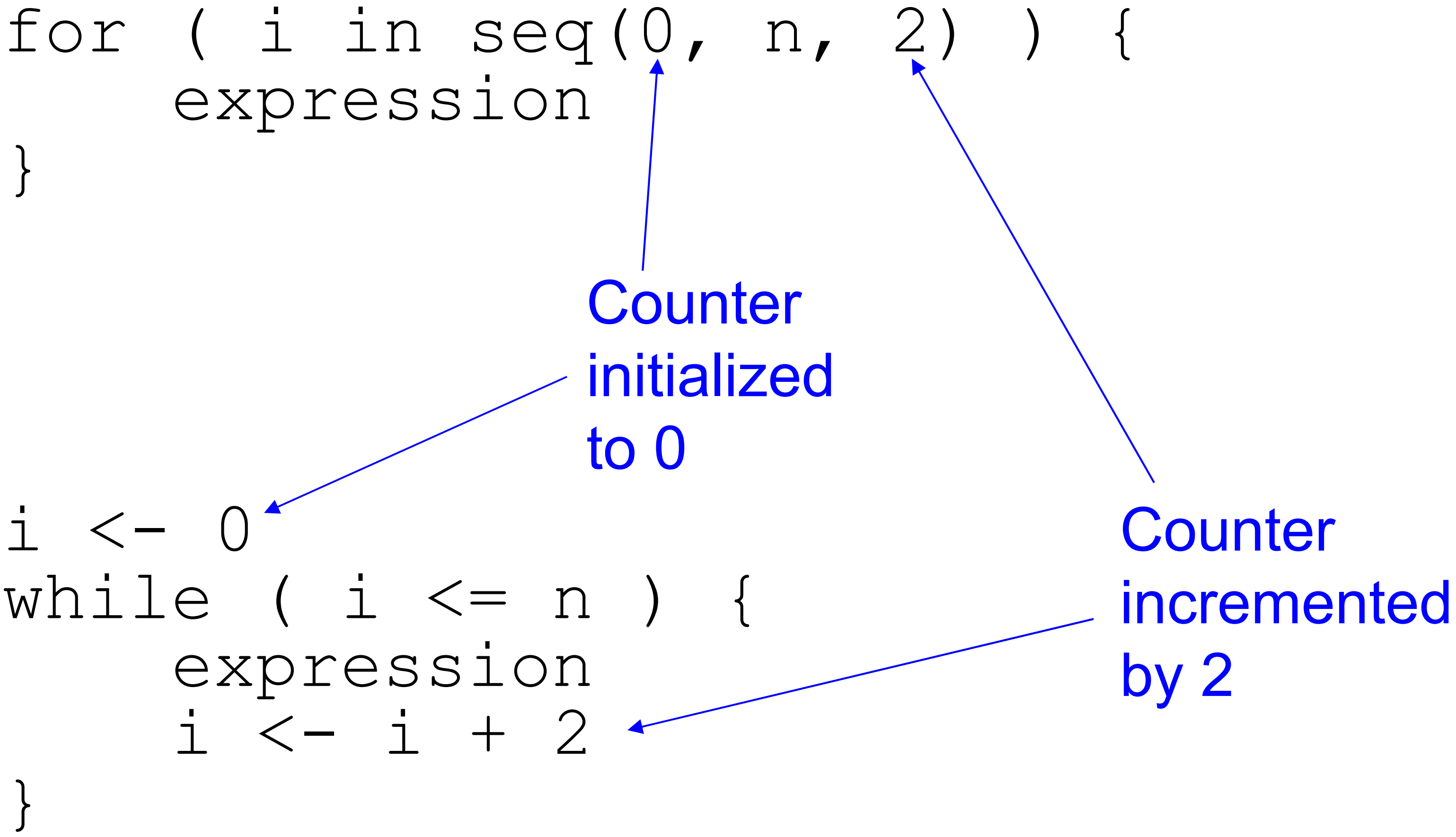
Hints:

1) One possible solution would be to use a selection structure.

2) What do each of the following lines of code return?

```
1:10
1:3
1:1
1:0
```

Increment variation



```
for ( i in seq(0, n, 2) ) {
    expression
}
```

Counter initialized to 0

```
i <- 0
while ( i <= n ) {
    expression
    i <- i + 2
}
```

Counter incremented by 2

R: `for` is vector controlled

R's for structure is actually vector controlled repetition, a special case of counter controlled repetition

seq is an expression that evaluates to a vector

```
for ( var in seq ) {
    expression
}
```

var will in turn be assigned the value of each element in the vector

Any vector will do!

R: `for` is vector controlled

Example

```
a <- c(0.51,0.57,0.09,1.02,1.10)
for ( number in a ) {
    print(number * 2)
}
```

What does this do?

Vector controlled repetition

Vector controlled repetition is a special case of counter controlled repetition

```
v #a vector
n <- length(v)
i <- 1
while ( i <= n ) {
    expression on v[i]
    i <- i + 1
}
```

Vector controlled repetition

- Many languages have convenience structures for vector (or object) controlled repetition
- Often called foreach or similar
- General pseudocode:

```
for each item in container
    do something
```

R: vector control with lists

List

- a special type of vector
- a container for multiple objects

creates a list

```
mylist <- list(obj1,obj2,obj3,...)
for ( object in mylist ) {
     ... do something
}
```

it doesn't have
to do something
TO the object
(but it can)

R: vector control with lists

Example

a, b, c, d
are numerical
vectors

```
datasets <- list(a,b,c,d)
for (x in datasets) {
    hist(x)
}
```

What does this do?