Today

- Questions about video lecture?
- Code: moving *Paramecium*
- Repetition structures:
  - counter control using for
- Data structures: vectors & indexing

Stochastic processes

- Substitute for all the stuff we don’t know
- Uncertainty about finer-scale processes
- Is the world deterministic or stochastic?
  - my view: depends on scale
  - individual scale is stochastic
  - individuals perceive the world as uncertain

Paramecium movement continued

R: `for` repetition structure

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

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

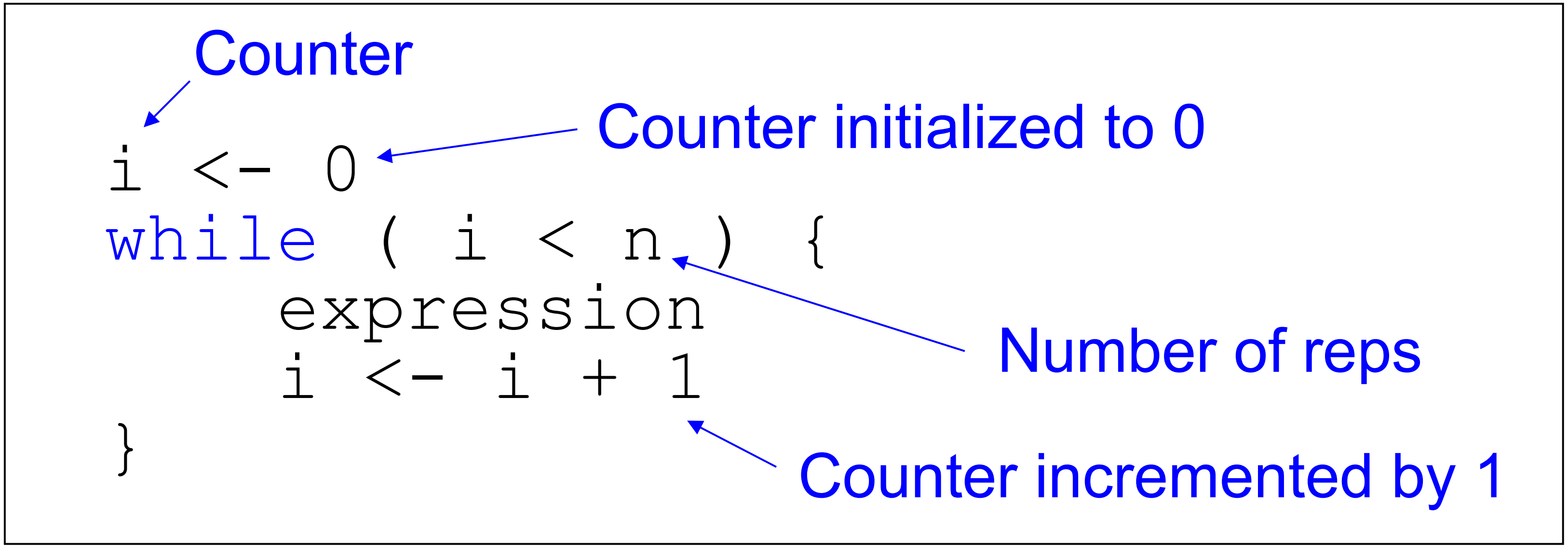
R: `for` repetition structure

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



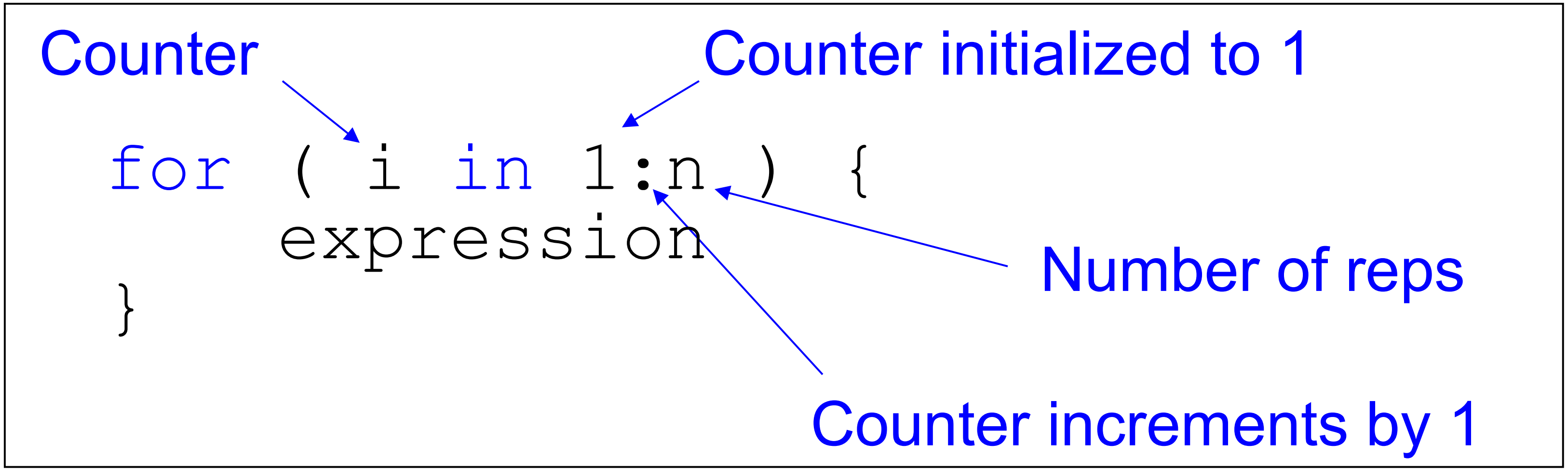
Counter

Counter initialized to 0

```
i <- 0
while ( i < n ) {
        expression
        i <- i + 1
}
```

Number of reps

Counter incremented by 1



Counter

Counter initialized to 1

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

Number of reps

Counter increments by 1

R

When the expression involves i

Counter initialized to 1

"or equal to"

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

option 1

Counter initialized to 1

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

R

When the expression involves i

Counter initialized to 0

```
i <- 0
while ( i < n ) {
    i <- i + 1
    expression using i
}
```

increment first

option 2

Counter initialized to 1

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

R

When the expression involves i

Counter initialized to 0

```
i <- 0
while ( i < n ) {
    expression using i
    i <- i + 1
}
```

Counter initialized to 0

```
for ( i in 0:(n-1) ) {
    expression using i
}
```

option 3

R

R: `for` repetition structure

Correct

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

Incorrect

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

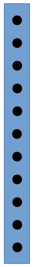
R

Paramecium movement
Replicate simulations with for

Data structures

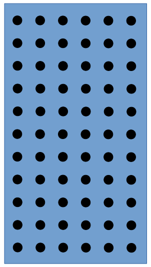
Scalar

single
element



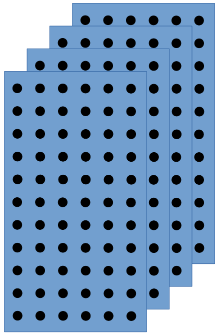
Vector

multiple
elements
1D



Matrix

multiple
elements
2D

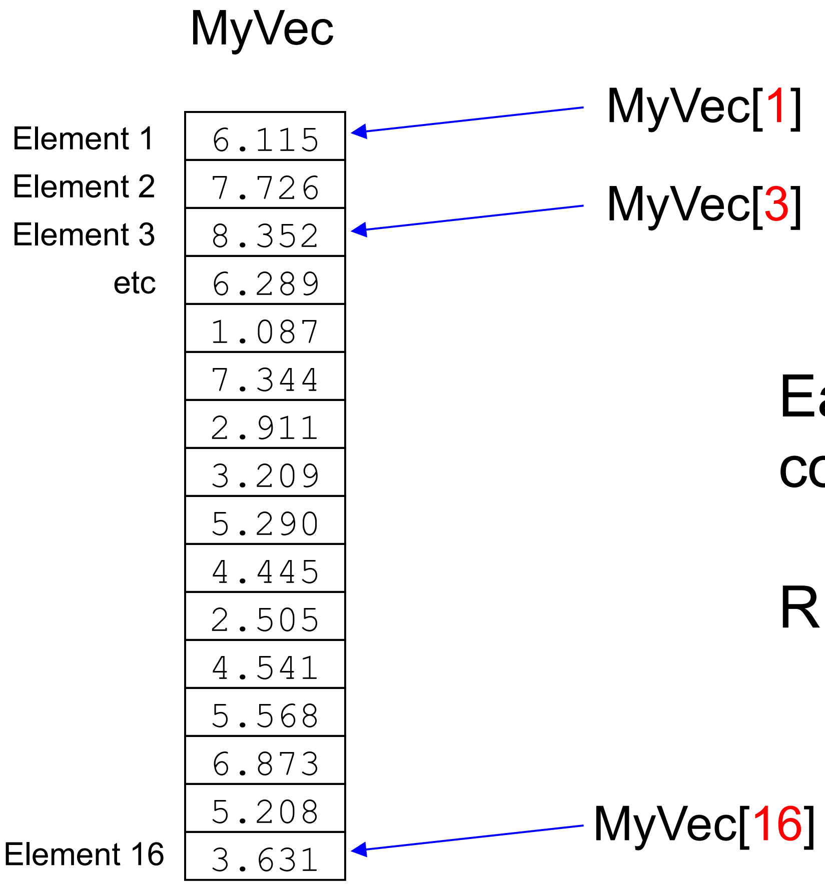


Array

multiple
elements
3D+

Generally, elements are the same type

Vectors: indexing



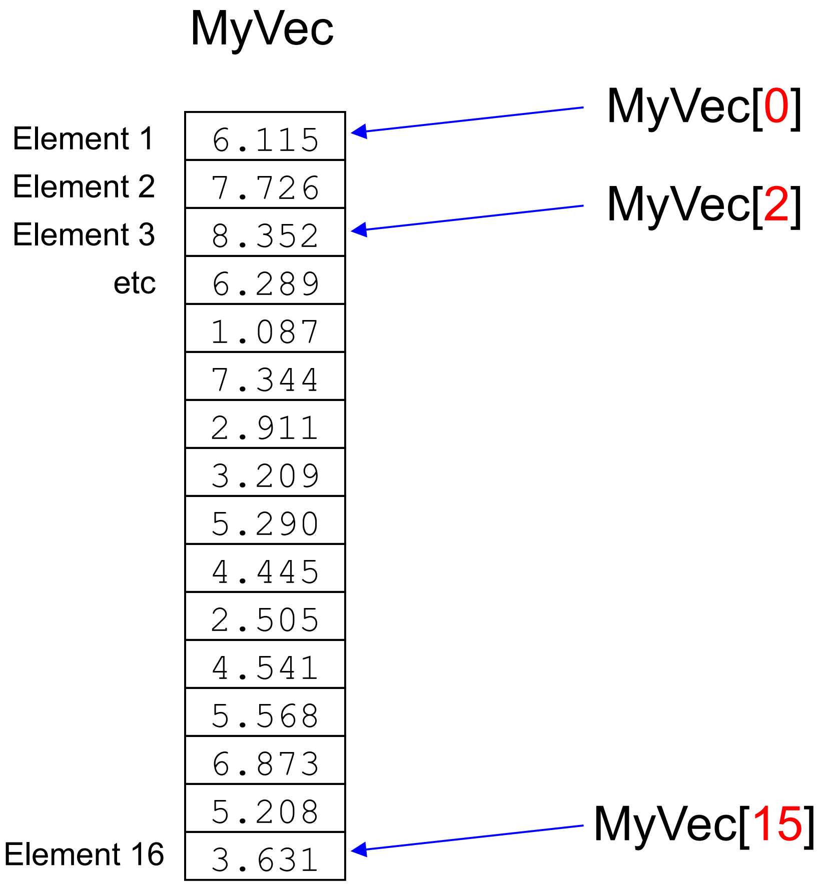
MyVec

Element 1 6.115 ← MyVec[1]
Element 2 7.726
Element 3 8.352 ← MyVec[3]
etc 6.289
1.087
7.344
2.911
3.209
5.290
4.445
2.505
4.541
5.568
6.873
5.208
Element 16 3.631 ← MyVec[16]

Each element is a slot in the computer's memory (RAM)

R uses math indexing

Vectors: indexing



MyVec

MyVec[0]

MyVec[2]

MyVec[15]

Element 1	6.115
Element 2	7.726
Element 3	8.352
etc	6.289
	1.087
	7.344
	2.911
	3.209
	5.290
	4.445
	2.505
	4.541
	5.568
	6.873
	5.208
Element 16	3.631

C and Python use
offset indexing

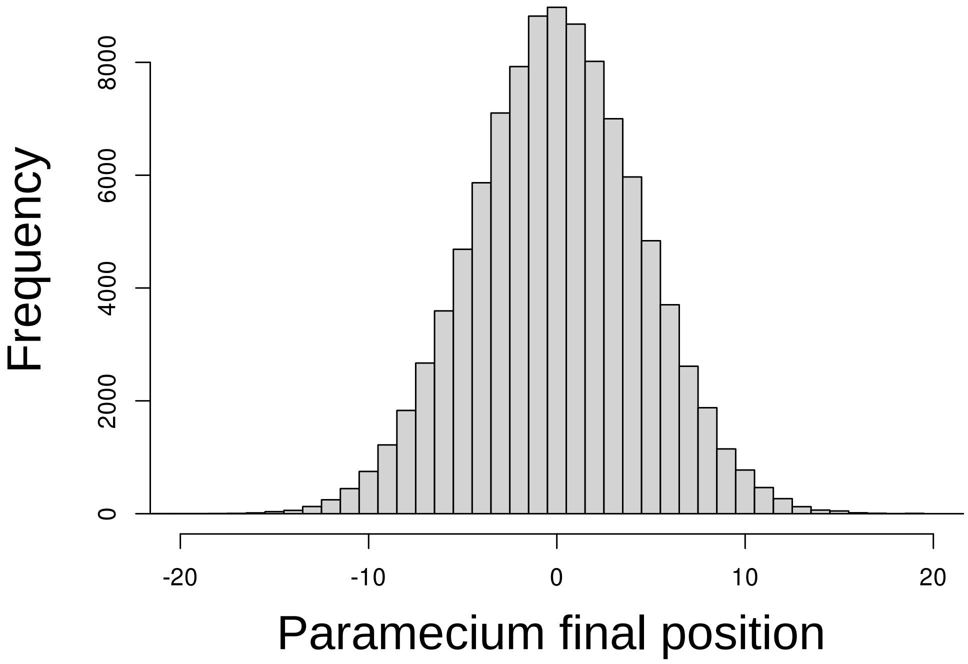
0 is first element

C, Py

Paramecium movement
Vector indexing to store results

Data generating process

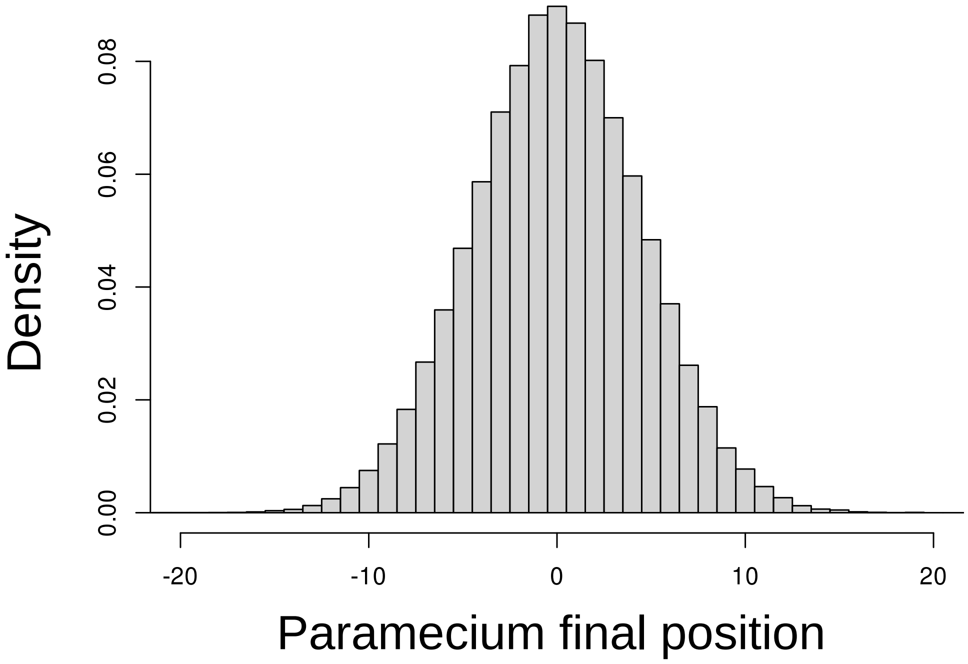
- Histogram of the data simulation is the distribution of the DGP



More exact as
n_reps -> Inf

Data generating process

- Histogram of the data simulation is the distribution of the DGP



For a proper distribution: divide by area under the curve to give probability density

In fact, in this case, since final position is an integer, this is a discrete distribution and the histogram shows the probability mass rather than density