ES 220 Lab 6 Chunks Example

Feb 28 2024

This example shows how to separate the code used to run the model from code that defines inputs. This is helpful in examining different scenarios, where you can change input values and then run the model code chunk using « ».

Note that there are several ways in which the specific code presented here does not fully align with your snowpack model, but the overall structure of defining and using chunks applies.

The code below creates a population model and then examines two scenarios with different growth rates.

This chunk defines one possible set of inputs for the model

```
# Initial population size
N <- 10

# Population growth rate
Lambda <- 1.02
```

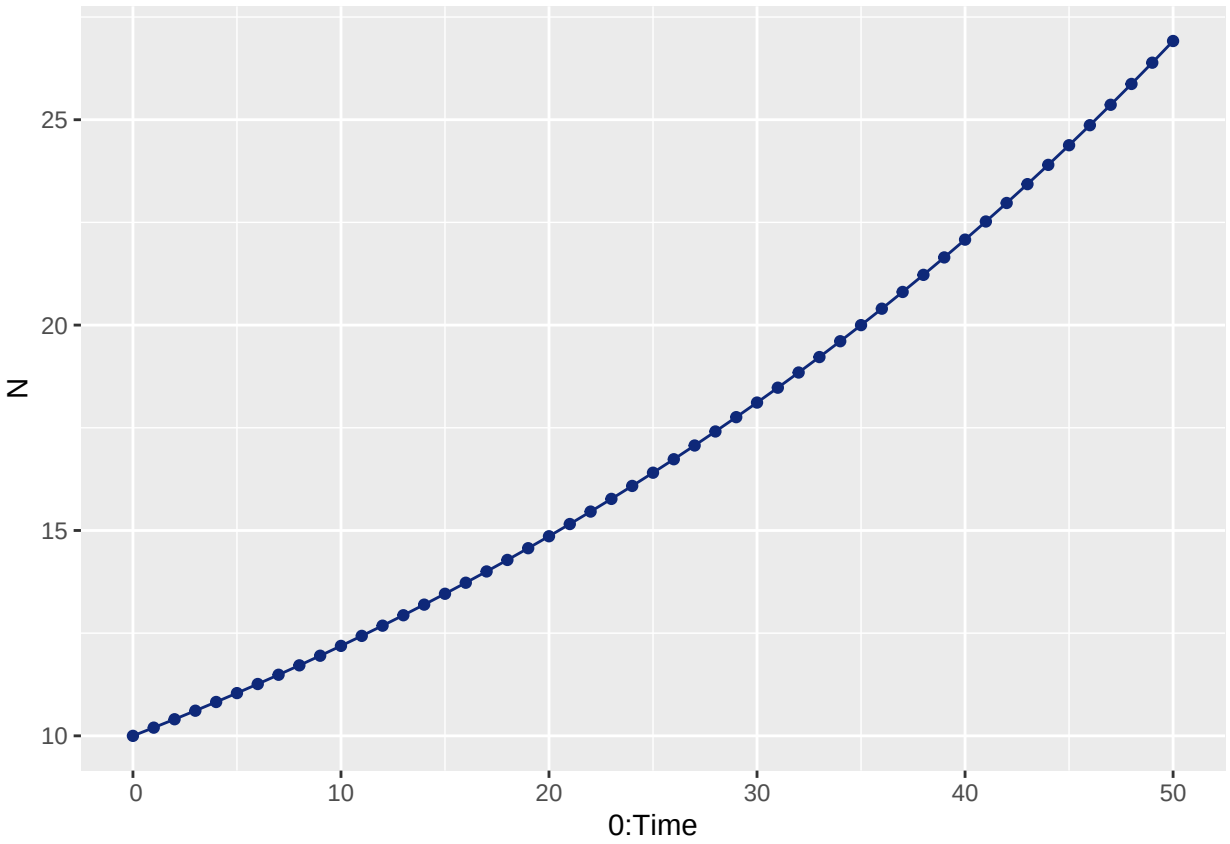
This chunk is the *model code only*, separate from the adjustable inputs

```
# Model duration; Note that by including it here, it will *always* be 50 (not an
# adjustable input outside of this chunk)
Time <- 50

# Loop that projects the population forward each time step
for (t in 1:Time) {

    N[t + 1] <- N[t] * Lambda

}
```

This chunk makes a plot of how population size (N) changes through time

Note that this chunk is separate from the one above in order to not have the model chunk *always* make a plot

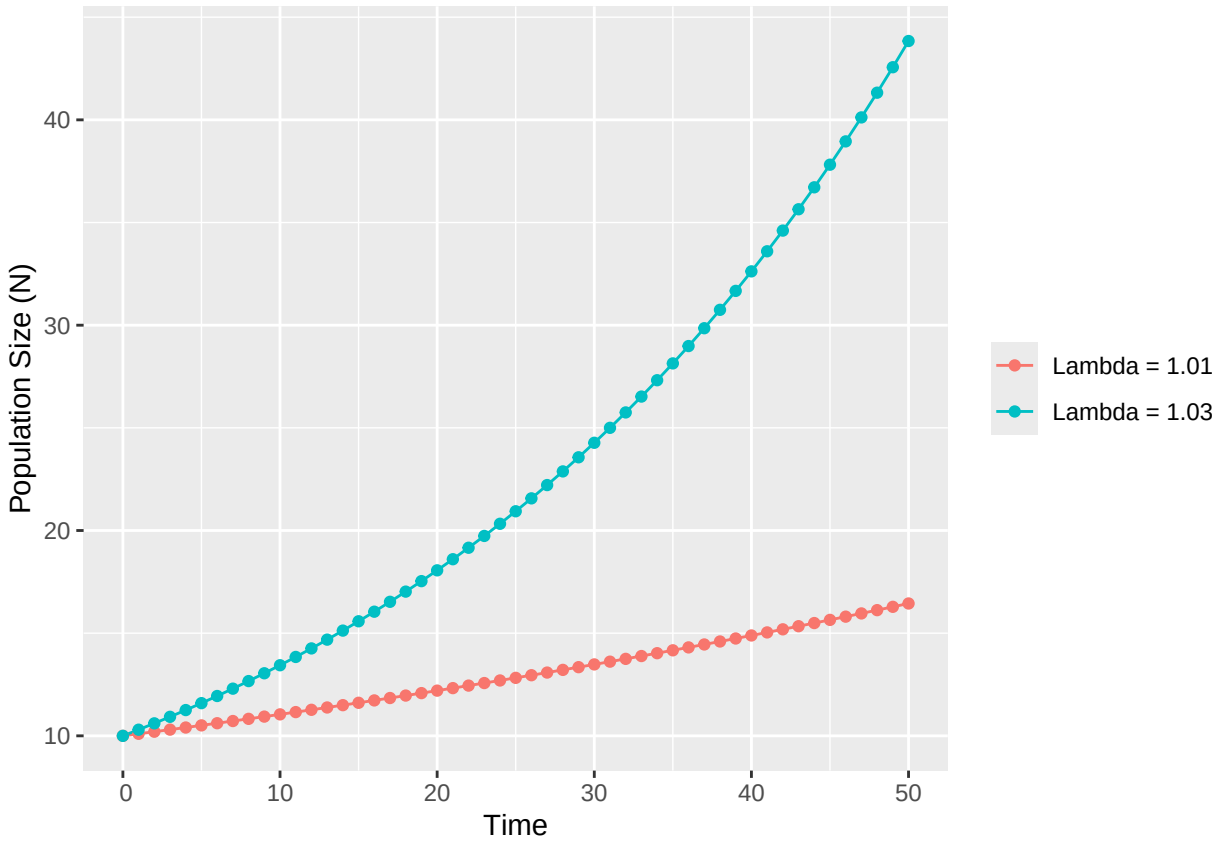
```
weplot(x = 0:Time, y = N, type = "point+line")
```



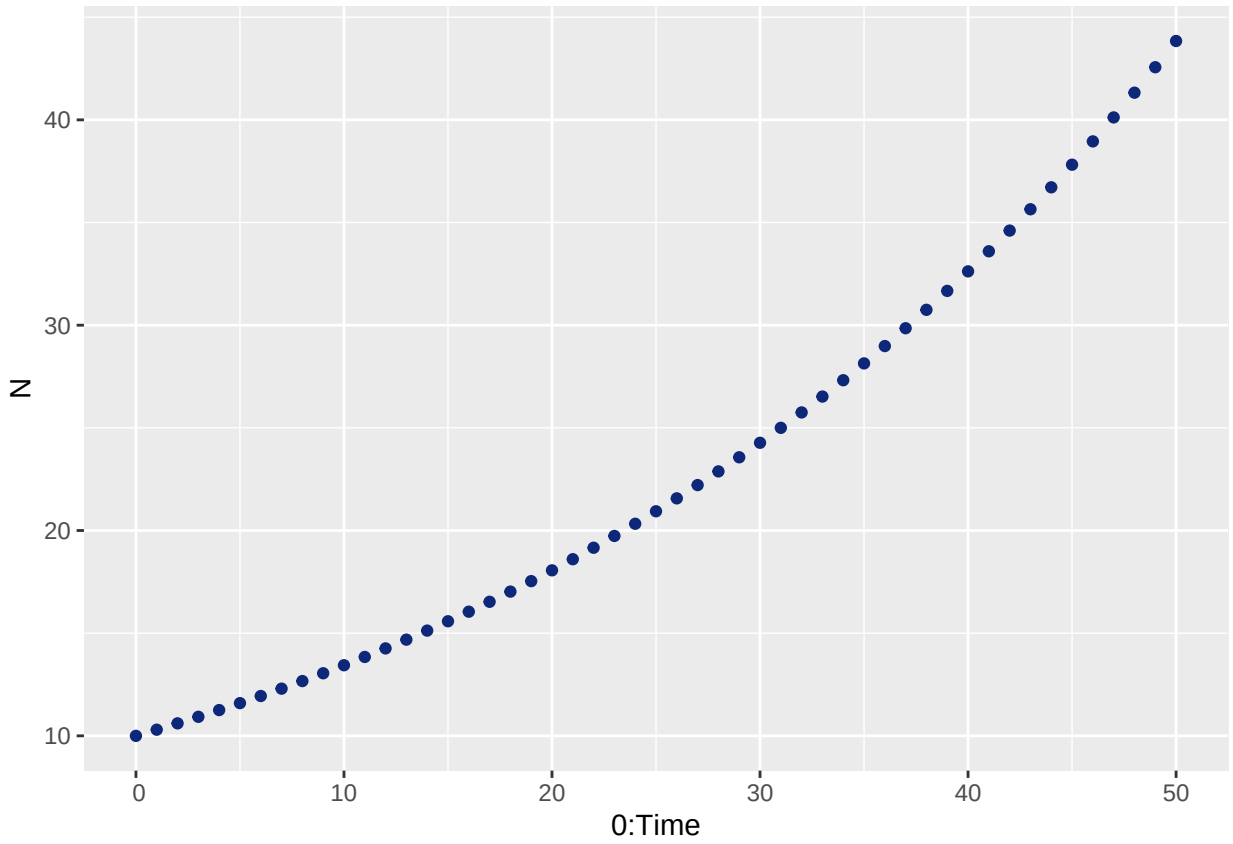
This chunk shows examples of how to use previous "Model" chunk to run scenarios with different population growth rates

```
# # Scenario 1: Lambda = 1.00

# Set Lambda to 1.00 and reset N to 10
Lambda <- 1.01
N <- 10

# Run the 'Model' code chunk

# Model duration; Note that by including it here, it will *always* be 50 (not an
# adjustable input outside of this chunk)
Time <- 50

# Loop that projects the population forward each time step
for (t in 1:Time) {

    N[t + 1] <- N[t] * Lambda

}



# Store the model output (N) in a unique object ('N_1.01')
N_1.01 <- N
```

```r
# # Scenario 2: Lambda = 1.03

# Set Lambda to 1.03 and reset N to 10
Lambda <- 1.03
N <- 10

# Run the 'Model' code chunk

# Model duration; Note that by including it here, it will *always* be 50 (not an
# adjustable input outside of this chunk)
Time <- 50

# Loop that projects the population forward each time step
for (t in 1:Time) {

    N[t + 1] <- N[t] * Lambda

}



# Store the model output (N) in a unique object ('N_1.03')
N_1.03 <- N


# # Make an overlay plot of both scenarios

weplot(x = 0:Time, y = list(N_1.01, N_1.03), type = "point+line", xlab = "Time", ylab = "Population Siz
    group.names = c("Lambda = 1.01", "Lambda = 1.03"))
```




```
# # Can we change the model duration to 100 and then run it?

# Set Lambda to 1.03, reset N to 10, and set Time to 100
Lambda <- 1.03
N <- 10
Time <- 100

# Run the 'Model' code chunk

# Model duration; Note that by including it here, it will *always* be 50 (not an
# adjustable input outside of this chunk)
Time <- 50

# Loop that projects the population forward each time step
for (t in 1:Time) {

    N[t + 1] <- N[t] * Lambda

}



# Plot results
weplot(x = 0:Time, y = N)
```




```
# No! The object 'Time' is 'hard-coded' into the 'Model' chunk and will therefore
# always be reset to 50
```