Feedback — Population Growth, Fitness, and Selection

Help Center

You submitted this quiz on **Sat 29 Jun 2013 2:17 AM PDT**. You got a score of **19.00** out of **19.00**.

Overview

In this exercise you are going to use a simple computer program for simulating the growth of a population. The simulation will be based on an exponential growth model and will involve organisms having one of two different genotypes ('A' and 'a'). By performing the simulation with a range of different model parameter values you will sharpen your intuitive understanding of the dynamics of the model.

You will also get a brief introduction to assessing how well a model describes some observed data (the "fit" of a model), and learn how model parameters can be estimated from such data (the process known as "model fitting").

Finally, you will start learning how to work in the UNIX environment that we will use in the rest of the course (and that is alo used by bioinformaticians/systems biologists in the real world).

If you are interested in learning more about UNIX you can browse this set of tutorials. If you would like more information about VirtualBox, the full documentation can be found here

Getting started

1. **Start Ubuntu Linux on VirtualBox**

   These computer exercises are best done by running your virtual Ubuntu machine, and viewing this webpage on a web browser started from there also (as opposed to having the web page open on your host operating system). There will be quite a bit of copying and pasting, and that is most easily done between windows within the guest operating system (Ubuntu).

   If you haven't done so already:

- Start the VirtualBox program
- In VirtualBox: Select your virtual machine (e.g., "molevol") and click "start"
- You may want to put your virtual machine in full screen view: In the "View" menu on VirtualBox, select "Switch to fullscreen"
- Open a web browser (click icon in upper left corner of Ubuntu desktop, and select "Web browser"
- Open this webpage and continue with the exercise

2. **Start Terminal window:**

   Click the Applications Menu icon in the upper left corner of your Ubuntu desktop and select "Terminal".

   This will give you a terminal window where you can enter commands (make sure to press "Return" after each command). The safest approach is to copy and paste commands from the instructions on this webpage to the terminal window. Note that the web browser and the terminal window use different shortcuts for copy/paste.

   - Browser - copy: Ctrl-C.
   - Browser - paste: Ctrl-V
   - Terminal - copy: Ctrl-SHIFT-C
   - Terminal - paste: Ctrl-SHIFT-V

3. **Construct today's working directory:**

   ```
   mkdir simulation
   ```

   ```
   cd simulation
   ```

   The `mkdir` (make directory) command constructs a new directory named "simulation". Subsequently the `cd` (change directory) command is used to select the newly constructed directory as "current working directory".

4. **Copy required files:**

   ```
   cp /home/student/data/growth.py ./growth.py
   ```

   ```
   cp /home/student/data/modelfit.data ./modelfit.data
   ```

```
ls -l
```

The `cp` command copies the simulation program (`growth.py`) and a file containing experimental data (`modelfit.data`), from the data directory to your current working directory. (The full path of the file was specified here).

The `ls` (list) command with option `-l` (long) shows information about the files present in the current directory.

5. **Start extra Terminal window and cd to working directory:**

   Again, click the Applications menu icon (upper left corner) and select Terminal. Now, enter the following command to `cd` to today's working directory:

   ```
   cd simulation
   ```

   In this part of the exercise, you will need two terminal windows open at the same time.

Question 1

- **Have a look at the simulation program:**

  ```
  nedit growth.py &
  ```

  This command opens the file growth.py (which is a simple text file) in the `nedit` editor. Having an ampersand ("&") at the end of the command makes `nedit` run in the background.

  The growth.py program is written in the programming language Python (warmly recommended by the way). You will probably be able to understand most of what is going on even if you have no programming experience (Note that hash signs - "#" - indicate the beginning of a comment. Text after # on a line will not be executed.)

- **Question:** what are the initial values of the parameters N0 (initial population size), fA (initial frequency of 'A'), rate_A (growth rate, or fitness, of 'A'), and rate_a (growth rate of 'a'). Report the values separated by blanks, in the following order:

```
N0 fA rate_A rate_a
```

You entered:

1000 0.3 1.2 1.2

Your Answer		Score	Explanation
1000	✓	0.25	
0.3	✓	0.25	
1.2	✓	0.25	
1.2	✓	0.25	
Total		1.00 / 1.00	

Question 2

- **Run the simulation program:**

  In one of your terminal windows, type the following:

  ```
  ./growth.py
  ```

  This command executes the program `growth.py`, which is located in your current working directory (the dot in "./growth.py" means "current directory"). For each generation the state of the population and the frequencies of the 'A' and 'a' alleles will be printed to the screen. The simulation runs for 20 generations, but you could change this by altering the `max_gen` parameter.

- **Run the simulation again, saving output to file:**

  ```
  ./growth.py > res.1
  ```

  The symbol ">" causes the output to be "redirected" to a file named "res.1" (you could have named it anything). Open the result file in `nedit` to verify that all output has been saved:

  ```
  nedit res.1 &
  ```

- **Question:** what is the total population size (Nt) at generation 0 and generation 20? Report the values separated by a space, in the following order:

  `Nt(0) Nt(20).`

  (Close the `nedit` window when you're done.)

You entered:

```
1000 38337
```

Your Answer		**Score**	**Explanation**
1000	✓	0.50	
38337	✓	0.50	
Total		1.00 / 1.00	

Question 3

- **Start the gnuplot program:**

  We now want to plot the results of the first simulation. In the second terminal window type the following:

  ```
  gnuplot
  ```

  gnuplot is an interactive plotting program. You write commands at a prompt much like you do when entering UNIX commands. The program can be used for plotting data sets from a file and for plotting functions given in symbolic form.

- **Quick tour of gnuplot features:**

  ```
  plot x**2-20*x-40
  ```

  This plots the function y = x^2 - 20x - 40. As you can see, the default x-range for plotting is -10:10. Let's change this so we can see the minimum point on the curve:

  ```
  plot [0:20] x**2-20*x-40
  ```

  (Note that, in gnuplot, previously entered commmands can be recalled by pressing the up-arrow. This is occasionally useful when you want to rerun, or slightly alter, previously issued plotting commands). The numbers enclosed in brackets specify the x-range (here 0 to 20). The y-range is automatically set based on the x-range, but may also be explicitly specified:

  ```
  plot [0:20][-160:0] x**2-20*x-40
  ```

  Finally, you also have the option of specifying legends and labels for the axes:

  ```
  set xlab "Branch length"
  ```

```
plot [0:20][-160:0] x**2-20*x-40 title "Parabola 1"
```

("`ylab`" would have set the label for the y-axis). There are many many other commands that we will not go into here (type "help" for online help).

- **Plot population sizes:**

  At the `gnuplot` prompt type the following:

  ```
  set xlab "Generation no."
  ```

  ```
  plot 'res.1' using 1:2 title "N" with linespoints, 'res.1' using 1:3 title "N_A" with linespoints, 'res.1' using 1:4 title "N_a" with linespoints
  ```

  The last command (which has to be entered on a single line) plots data from selected columns in the file 'res.1'. For instance, "`using 1:4`" means that we should use the numbers in column 1 (generation number) for x-values, and the numbers in column 4 (number of organisms with allele 'a') for y-values.

  Specifically, we here plot total population size ("N"), the number of organisms with allele 'A' ("N_A"), and the number of organisms with allele 'a' ("N_a") for each generation. Recall that in this run the two alleles had the same fitness (rate_A = rate_a = 1.2) but different initial population sizes (fA = 0.3, fa = 0.7).

- **Plot allele frequencies:**

  At the `gnuplot` command type the following:

  ```
  plot [][0:1]'res.1' u 1:5 tit "A freq" w linespoints, 'res.1' u 1:6 tit "a freq" w linespoints
  ```

  Here we have plotted the *frequencies* of the two alleles for each simulated generation. The leading "`[] [0:1]`" indicate that we want to use the automatic x-range and that we want the y-axis to span the interval 0:1. Also note that gnuplot commands can be abbreviated ("u" for "using", "w" for "with", "tit" for "title").

- **Question:** How do the frequencies of the two alleles (**fA** and **fa**) behave for this case where the two alleles have the same fitness?

  Symbols used:

- ↑: goes up
- ↓: goes down
- →: remains constant

Your Answer		Score	Explanation
○ **fA: ↑ fa: ↑**			
● **fA: → fa: →**	✓	1.00	
○ **fA: ↑ fa: ↓**			
○ **fA: ↓ fa: ↓**			
○ **fA: ↓ fa: ↑**			
Total		1.00 / 1.00	

Question Explanation

When the two different alleles have the same fitness, the allele frequencies are constant.

Question 4

- <u>**Simulation 1. Part 1**</u>
- **Run simulation:**

In the terminal window where you are not running gnuplot: Using what you learned above, run the simulation program for the parameter values listed below, and save the output to a file named `"res.1"` (without the quotes). You should use `nedit` to alter the relevant parameter values in the `growth.py` file. **Remember to save `growth.py` after making the alterations (*File -> Save*).**

```
N0 = 50     rate_A = 1.2  rate_a = 1.2  fA = 0.3
```

- **Question:** What is the relative fitness of allele 'a' (using 'A' as a reference):

You entered:

1

Your Answer		Score	Explanation
1	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

The relative fitness of a genotype, is the absolute fitness of that genotype divided by the absolute fitness of a reference genotype (typically the one with the highest fitness).

Question 5

- **Simulation 1. Part 2**
- **Plot population sizes:**

  At the `gnuplot` prompt (in the other terminal window) type (paste) the following:

```
set xlab "Generation no."
```

```
plot 'res.1' using 1:2 title "N" with linespoints, 'res.1' using 1:3 title "N_A" with linespoints, 'res.1' using
```

```
1:4 title "N_a" with linespoints
```

- **Plot allele frequencies:**

  At the `gnuplot` prompt type (paste) the following:

  ```
  plot [][0:1] 'res.1' u 1:5 tit "A freq" w linespoints, 'res.1' u 1:6 tit "a freq" w linespoints
  ```

- **Question:** What is the behavior of population sizes (**NA** and **Na**) and allele frequencies (**fA** and **fa**) for this set of parameter values?

Your Answer		Score	Explanation
○ **NA: ↓ Na: ↑ fA: ↓ fa: ↑**			
○ **NA: ↑ Na: ↑ fA: ↓ fa: ↑**			
○ **NA: ↓ Na: ↓ fA: ↑ fa: ↓**			
● **NA: ↑ Na: ↑ fA: → fa: →**	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

When the two genotypes have the same fitness, allele frequencies will remain at their starting value.

Question 6

- **Simulation 2. Part 1**
- **Run simulation:**

  Run the simulation program for the parameter values listed below, and save the output to a file named "`res.2`" (without the quotes).

```
N0 = 1000   rate_A = 0.7  rate_a = 0.7  fA = 0.3
```

- **Question:** What is the relative fitness of allele 'a' (using 'A' as a reference):

You entered:

1

Your Answer		Score	Explanation
1	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

The relative fitness of a genotype, is the absolute fitness of that genotype divided by the absolute fitness of a reference genotype (typically the one with the highest fitness).

Question 7

- **Simulation 2. Part 2**
- **Plot population sizes:**

  At the `gnuplot` prompt type (paste) the following:

  ```
  plot 'res.2' using 1:2 title "N" with linespoints, 'res.2' using 1:3 title "N_A" with linespoints, 'res.2' using 1:4 title "N_a" with linespoints
  ```

- **Plot allele frequencies:**

At the gnuplot prompt type (paste) the following:

```
plot [][0:1] 'res.2' u 1:5 tit "A freq" w linespoints, 'res.2' u 1:6 tit "a freq" w linespoints
```

- **Question:** What is the behavior of population sizes (**NA** and **Na**) and allele frequencies (**fA** and **fa**) for this set of parameter values?

Your Answer		Score	Explanation
● **NA: ↓ Na: ↓ fA: → fa: →**	✓	1.00	
○ **NA: ↑ Na: ↑ fA: ↑ fa: ↓**			
○ **NA: ↓ Na: ↓ fA: ↑ fa: ↓**			
○ **NA: ↑ Na: ↑ fA: ↓ fa: ↑**			
Total		1.00 / 1.00	

Question Explanation

When the two genotypes have the same fitness, allele frequencies will remain at their starting value. This is true even when growth rates are less than 1 (corresponding to decreasing population size).

Question 8

- **Simulation 3. Part 1**
- **Run simulation:**

Run the simulation program for the parameter values listed below, and save the output to a file named "res.3".

```
N0 = 1000   rate_A = 2.0  rate_a = 1.5  fA = 0.02
```

- **Question:** What is the relative fitness of allele 'a' (using 'A' as a reference):

You entered:

0.75

Your Answer		Score	Explanation
0.75	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

The relative fitness of a genotype, is the absolute fitness of that genotype divided by the absolute fitness of a reference genotype (typically the one with the highest fitness).

Question 9

- **Simulation 3. Part 2**
- **Plot population sizes:**

  At the `gnuplot` prompt type (paste) the following:

  ```
  plot 'res.3' using 1:2 title "N" with linespoints, 'res.3' using 1:3 title "N_A" with linespoints, 'res.3' using 1:4 title "N_a" with linespoints
  ```

- **Plot allele frequencies:**

  At the `gnuplot` prompt type (paste) the following:

  ```
  plot [][0:1] 'res.3' u 1:5 tit "A freq" w linespoints, 'res.3' u 1:6 tit "a freq" w linespoints
  ```

- **Question:** What is the behavior of population sizes (**NA** and **Na**) and allele frequencies (**fA** and **fa**) for this set of parameter values?

Your Answer		Score	Explanation
○ **NA: ↑ Na: ↑ fA: ↓ fa: ↑**			
● **NA: ↑ Na: ↑ fA: ↑ fa: ↓**	✓	1.00	
○ **NA: ↓ Na: ↑ fA: ↓ fa: ↑**			
○ **NA: ↓ Na: ↓ fA: ↑ fa: ↓**			
Total		1.00 / 1.00	

Question Explanation

Both genotypes have growth rates larger than 1 and the population sizes will therefore increase. The fitness of genotype a is smaller than that of genotype A (its relative fitness is less than 1), and fa will therefore decrease.

Question 10

- **Simulation 4. Part 1**
- **Run simulation:**

  Run the simulation program for the parameter values listed below, and save the output to a file named "`res.4`".

```
      N0 = 1000   rate_A = 1.2  rate_a = 0.9  fA = 0.02
```

- **Question:** What is the relative fitness of allele 'a' (using 'A' as a reference):

You entered:

```
0.75
```

Your Answer		Score	Explanation
0.75	✓	1.00	
Total		1.00 / 1.00	

Question Explanation

The relative fitness of a genotype, is the absolute fitness of that genotype divided by the absolute fitness of a reference genotype (typically the one with the highest fitness).

Question 11

- **Simulation 4. Part 2**
- **Plot population sizes:**

  At the `gnuplot` prompt type (paste) the following:

  ```
  plot 'res.4' using 1:2 title "N" with linespoints, 'res.4' using 1:3 title "N_A" with linespoints, 'res.4' using 1:4 title "N_a" with linespoints
  ```

- **Plot allele frequencies:**

  At the `gnuplot` prompt type (paste) the following:

  ```
  plot [][0:1] 'res.4' u 1:5 tit "A freq" w linespoints, 'res.4' u 1:6 tit "a freq" w linespoints
  ```

- **Question:** What is the behavior of population sizes (**NA** and **Na**) and allele frequencies (**fA** and **fa**) for this set of parameter values?

Your Answer		Score	Explanation
○ NA: ↓ Na: ↑ fA: → fa: →			
● NA: ↑ Na: ↓ fA: ↑ fa: ↓	✓	1.00	
○ NA: → Na: → fA: → fa: →			
○ NA: ↓ Na: ↑ fA: ↓ fa: ↑			
Total		1.00 / 1.00	

Question Explanation

The absolute fitness of A is greater than 1, and the population of A will therefore increase. The population of a will decrease, because it has an absolute fitness that is less than 1 (every individual on average leaves less than 1 offspring). The relative fitness of a is less than 1, and fa will therefore decrease (while fA will go up).

Question 12

- **Simulation 5. Part 1**
- **Run simulation:**

  Run the simulation program for the parameter values listed below, and save the output to a file named "`res.5`".

  ```
              N0 = 10000  rate_A = 0.8  rate_a = 0.6  fA = 0.02
  ```

- **Question:** What is the relative fitness of allele 'a' (using 'A' as a reference):

You entered:

0.75

Your Answer		Score	Explanation
0.75	✓	1.00	
Total		1.00 / 1.00	

Question 13

- **Simulation 5. Part 2**
- **Plot population sizes:**

  At the `gnuplot` prompt type (paste) the following:

  ```
  plot 'res.5' using 1:2 title "N" with linespoints, 'res.5' using 1:3 title "N_A" with linespoints, 'res.5' using 1:4 title "N_a" with linespoints
  ```

- **Plot allele frequencies:**

  At the `gnuplot` prompt type (paste) the following:

  ```
  plot [][0:1] 'res.5' u 1:5 tit "A freq" w linespoints, 'res.5' u 1:6 tit "a freq" w linespoints
  ```

- **Question:** What is the behavior of population sizes (**NA** and **Na**) and allele frequencies (**fA** and **fa**) for this set of parameter values?

Your Answer		Score	Explanation
● **NA: ↓ Na: ↓ fA: ↑ fa: ↓**	✓	1.00	

○ **NA: ↑ Na: ↓ fA: ↑ fa: ↓**		
○ **NA: → Na: → fA: → fa: →**		
○ **NA: ↓ Na: ↑ fA: → fa: →**		
Total	1.00 / 1.00	

Question Explanation

Both genotypes have growth rates less than 1 and the population sizes will therefore decrease. The fitness of genotype a is smaller than that of genotype A (its relative fitness is less than 1), and fa will therefore decrease (while fA goes up).

Question 14

- **Compare simulation 3 and 5**
  Simulation 3 had `rate_A = 2.0` and `rate_a = 1.5`, while simulation 5 had `rate_A = 0.8` and `rate_a = 0.6`. Using what you have learned: Plot fA and fa for both simulations in a single plot.
- **Question:** How does the behavior of fA and fa compare in the two simulations?

Your Answer		Score	Explanation
○ fA increased more slowly in simulation 3 than in simulation 5.			
○ fA increased in simulation 3 and decreased in simulation 5.			
○ fA increased more rapidly in simulation 3 than in simulation 5.			
● The frequencies change in exactly the same manner in the two simulations (the graphs are perfectly overlapping).	✓	1.00	

Total 1.00 / 1.00

Question Explanation

The relative fitness of genotype a is 0.75 in both simulations (and also in simulation 4 in fact). The frequency of the genotypes therefore behave in exactly the same manner, regardless of the fact that in the first case the populations are increasing in size, while they are decreasing in the second case.

Question 15

Model Fit, Parameter Estimation

In this part of the exercise, we will briefly consider some aspects of the fit between models and reality.

- **Have a look at the data file:**

  ```
  nedit modelfit.data &
  ```

  This file contains a set of (pseudo) empirical data: the size of a population (column labeled "N") for a number of generations (column labeled t).

- **Plot data points:**

  In the terminal window where you have gnuplot running, type the following:

  ```
  plot [-1:13] 'modelfit.data'
  ```

- **Estimating parameters of exponential growth model from data:**

  We will now assume that the population is growing exponentially according to this model:

```
Nt = N0 * exp(r*t)
```

where `N0` is the initial population size, `r` is the instantaneous rate of increase, and `t` is the generation number. Our model thus has two "free parameters": `N0` and `r`. You will now attempt to find a "good" set of values for the `N0` and `r` parameters. We will take "good" values to be those that cause the theoretical curve to lie as close as possible to the observed data. This process is called model-fitting.

There are actually several different ways of defining "as close as possible". One measure that turns out to be convenient is the "sum of squared residuals" (SSR; the word "error" is sometimes used instead of "residual"). The approach is the following: for each x,y point in the data set, the difference between the observed y and the y-value predicted by the model is computed. This difference (the "error" or "residual") is then squared, and the sum of all the squared residual terms is then taken to be an indication of how well the model fits the data. The best fitting model thus has the smallest possible SSR, and this approach is therefore referred to as "least squares model fitting". Another measure of model fit that we will return to later in the course is the model likelihood.

- **Find a good set of parameter values:**

  In order to fit a model to our data, we first have to define the model in the form of a function in gnuplot. We will then use gnuplot's built-in "fit" command to find the best (here meaning "least squares") estimates of the two parameters.

  First, define the model function by entering the following at your gnuplot prompt:

  ```
  f1(x) = N0 * exp(r * x)
  ```

  For the purpose of this run we will also provide initial guesses at the parameter values. (This is not strictly necessary - the gnuplot program will choose starting values if you don't provide them - but it is sometimes helpful if the fitting procedure gets stuck in a local optimum).

  ```
  N0 = 500
  ```

  ```
  r = 0.1
  ```

  ```
  plot [-1:13] 'modelfit.data', f1(x)
  ```

  The last command above plots the data along with our (not particularly well-fitting) first guess at a model that describes the data. You can try out a few different values of the "`N0`" and "`r`" parameters if you want to get a feeling for where the best values are.

Now, fit the model to the data, and plot the resulting best-fit line, by issuing the following commands at your gnuplot prompt:

```
fit f1(x) 'modelfit.data' via N0,r
```

```
plot [-1:13] 'modelfit.data', f1(x)
```

If you inspect the output you can see how the gnuplot program uses an iterative method to find the best parameter values (i.e., the parameter values that result in the smallest sum of squared errors). Notice how the line now is much closer (on average) to the data points.

- **Question:** Find the "final set of parameters" (N0 and r) and the "final sum of squares of residuals" (SSR) of the fitted model near the end of the gnuplot output. Report the values separated by blanks, in the following order:

  `N0 r SSR`

You entered:

626.18 0.209828 1.77449e+06

Your Answer		Score	Explanation
626.18	✓	0.33	
0.209828	✓	0.33	
1.77449e+06	✓	0.33	
Total		1.00 / 1.00	

Question 16

- **Fit polynomial model:**

  So far we have assumed that the exponential model was the best choice for describing the growth data. This model has two free parameters: "`N0`" and "`r`". Let us now consider an alternative model with 7 free parameters, namely the 6'th order polynomial

  ```
  f2(x)=a+bx+cx^2+dx^3+ex^4+fx^5+gx^6
  ```

  The free parameters of this model are: `a, b, c, d, e, f,` and `g`. Note that we now have as many free parameters as data points.

  Using the same approach as we did above for an exponential model, you should now fit this polynomial model to the data (you may want to call this "f2" so it wont get mixed up with the exponential model, which we called f1). **Note that in gnuplot you have to explicitly write mutiplication using "*", and exponentiation as "**"**; for instance cx^2 would be written: `c*x**2`. As mentioned above, you do not have to specify initial guesses for the 7 parameter values, but you are allowed to if you want.

  If you want to visually compare the fit of the two models, you can use the following command:

  ```
  plot 'modelfit.data', f1(x), f2(x)
  ```

- **Question:** Report the "final set of parameters" (a, b, c, d, e, f, g) and the "final sum of squares of residuals" (SSR) of the fitted model. Report the values separated by blanks, in the following order:

  ```
  a b c d e f g SSR
  ```

You entered:

233 3614.18 -3094.67 1053.35 -167.97 12.6974 -0.364149 1.22128e-21

Your Answer		Score	Explanation
233	✓	0.12	
3614.18	✓	0.12	
-3094.67	✓	0.12	

1053.35	✓	0.12
-167.97	✓	0.12
12.6974	✓	0.12
-0.364149	✓	0.12
1.22128e-21	✓	0.12
Total		1.00 / 1.00

Question 17

- **Question:** How is the fit of the polynomial compared to that of the exponential model?

Your Answer		Score	Explanation
● Polynomial model has better fit	✓	1.00	
○ Polynomial model has worse fit			
○ The models have the same fit			
Total		1.00 / 1.00	

Question Explanation

The final sum of squares of residuals (SSR, also called “sum of squared errors”) of the exponential model is much larger than the SSR of the sixth order polynomial. This means that the sixth order polynomial fits the data better than the exponential model. This can also be seen on the plot (where the polynomial goes through all data points).

Question 18

- **Assess predictive performance of polynomial and exponential models:**

  Imagine that we have now collected one additional data point, and this shows us that the real observed population size at generation t=14 is N=11825. Now, use (1) the exponential model, f1, and (2) the polynomial model, f2, to predict the population size at generation 14. This is done by entering the value 14 for x in the model equations that you found by fitting. You can do this in gnuplot as follows (assuming that the model parameters have not been set to new values after fitting the two models):

  ```
  print f1(14)
  ```

  ```
  print f2(14)
  ```

- **Question:** Report the predicted values separated by a space: `f1(14) f2(14)`

You entered:

```
11816.1939351818 -30952.0000000005
```

Your Answer		Score	Explanation
11816.1939351818	✓	0.50	
-30952.0000000005	✓	0.50	
Total		1.00 / 1.00	

Question 19

- **Question:** Based on the data point for t=14 which model do you now think has captured the biological reality best?

Your Answer	Score	Explanation
○ The models are equally good		
● The exponential model is best ✓	1.00	
○ The polynomial model is best		
Total	1.00 / 1.00	

Question Explanation

The exponential model is able to make a much more precise prediction of the extra data point, and we therefore think it has captured the biological reality best. The polynomial model has a very good fit to the first set of data points, but is terrible at predicting the extra point. This is essentially because the exponential model captures the important trend in the data (exponential growth) while ignoring the noise in the data (points are scattered randomly around the exponential curve). The polynomial model, on the other hand, is essentially fitted to the noise, but has not at all captured the important exponential trend. We say such a model is overfitted. SSR is a useful way of finding the best estimate of a set of parameter values for a given model, but it does not allow you to choose between two different types of model. Later in the course we will deal with stringent ways of choosing a good model.