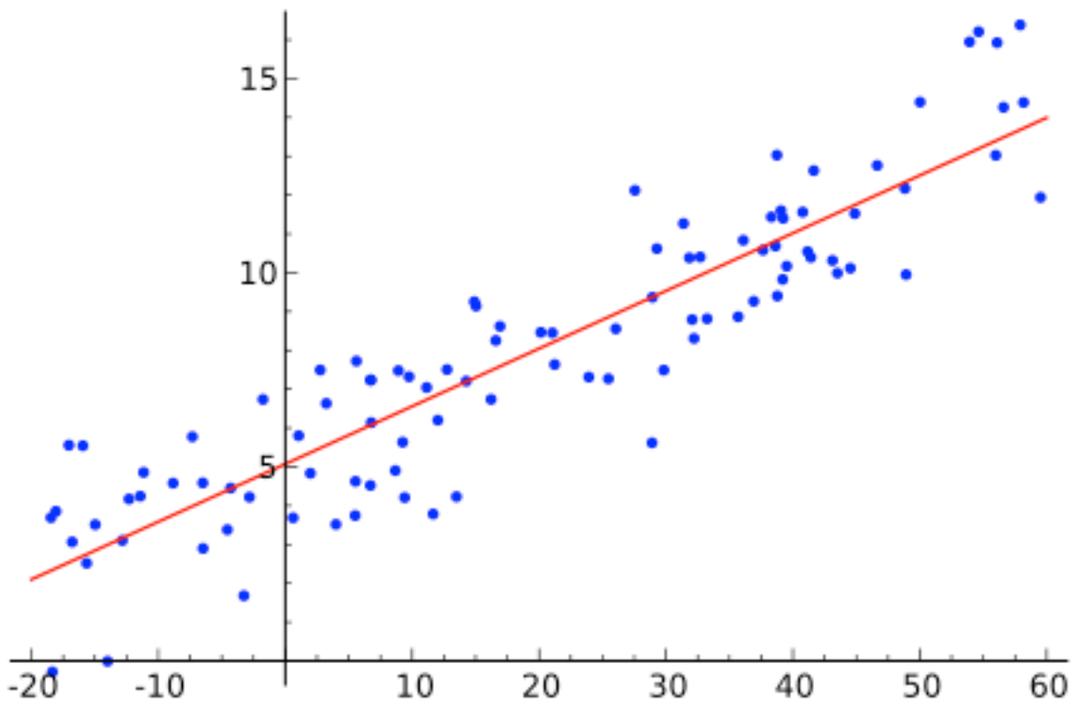
Regression and Data Analysis

Dr. Thomas Robey
October 8&9, 2016



- **Observation** - Experimental data or measurement. Expressed as a row vector **d** = [d_1, …, d_n].
- **Model** - A mathematical function.

We talk about **fitting** a model to the observations. One way is to “eyeball” the fit but generally a more automatic and rigorous approach is used.

Error

If we add an error term we have f(x) + e where then for each data point we have

$e_i = d_i - f(x_i)$

Then we have e_i, i=1, …, n where there are n data points. This is an error vector. How do we know if this error vector is “small?” First we have to be able to compare the size of vectors.

Norms

A norm $\|\mathbf{x}\|$ is a non-negative number where $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = \mathbf{0}$. A norm must also have $\|k\mathbf{x}\| = |k| \, \|\mathbf{x}\|$ and the triangle inequality $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ (the legs of a triangle cannot be shorter than the direct distance or hypotenuse).

Common norms:

$$\|\mathbf{x}\|_1 = \sum |x_i|$$
$$\|\mathbf{x}\|_2 = \left(\sum x_i^2\right)^{1/2}$$
$$\|\mathbf{x}\|_\infty = \max |x_i|$$

A data outlier is a data observation which lies much further from the model than the other data points. Which norm is least sensitive to an outlier?

If we consider a two dimensional space where **x** = [3, 4] then we have

$$\|\mathbf{x}\|_2 = (3^2 + 4^2)^{1/2} = 5$$

For Euclidean geometry this is simply the Euclidean distance.

Least Squares Method

The Least Squares Method is just choosing the parameters for the function such that $||\mathbf{e}||_2$ is minimized.

- Linear problem
- Easy to solve
- Usually adequate fit

Fits using the Method of Least Squares can be sensitive to data outliers. The Method of Least Squares is often used where there are better alternatives because it is easy to use and widely accepted.

Models

In a previous slide, the model was a line

$f(x) = mx+b$

but the model can be any function. Sometimes the problem suggests the function (e.g. elliptical orbit). There are approaches for the case when the function is unknown but this requires a lot of data and subjects the results to a lot more questioning. But sometimes the function that is found to provide the best fit can suggest a theoretical underpinning.

Consider the case where we have two data points. If we fit a linear model to the two data points what is the result? Since two points determine a line, the line exactly describes the data. In general, a model with m - 1 parameters will exactly describe n data points. Is this a good idea?

The **sum of square errors** (SSE) is $(\|\mathbf{e}\|_2)^2$. How do we choose the degree of the polynomial, m, in the model? One way is to compute

$$(\|\mathbf{e}_m\|_2)^2/(n - m - 1)$$

and continue increasing m as long as the amount decreases significantly. What is the value of this when the degree of the polynomial exactly fits the data points?

Coefficient of Determination

The **coefficient of determination** is a measure of how well the model fits the data. For a linear regression model

$$R^2 = \{ (1 / n) * \Sigma [(x_i - ave(x)) * (y_i - ave(y))] / (\sigma_x * \sigma_y) \}^2$$

- The coefficient of determination ranges from 0 to 1
- $R^2 = 0$ means the model has no predictability for the data
- $R^2 = 1$ means the model perfectly predicts the data

A common error that is made is using a high R^2 to say that the model is the right model. Some data sets are easy to get a high R^2 and others are not. It may be that a different model fit to the same data is better than the model with a high R^2.

Simple Linear Regression

The case where the model is y = b + mx has some rather simple formulas.

$$m = (\sum y_i x_i - \sum y_i \sum x_i /n)/(\sum(x_i - ave(x))^2)$$

$$b = ave(y) - m\ ave(x)$$

The SimpleLinearRegression.pdf in the examples is a worksheet for calculating a simple linear regression.

Rocket example

We are given some data pertaining to a rocket.

height = [100, 200, 300, 450, 600, 800, 1000]
distance = [253, 337, 395, 451, 495, 534, 574]

First, calculate using the formulas for simple linear regression

$$m = (\sum y_i x_i - \sum y_i \sum x_i /n)/(\sum(x_i - ave(x))^2)$$
$$m = (1712350 - 3039 * 3450/7)/642143$$
$$m = (1712350 - 1497793)/642143$$
$$m = 0.3341$$

$$b = ave(y) - m\ ave(x)$$
$$b = 434.143 - 0.3341 * 492.857$$
$$b = 269.48$$

$$f(x) = 269.48 + 0.3341*x$$

R Statistical Software

R is a free statistical analysis software package available for Windows, Mac and Linux (see Appendix). On the Mac click on R.app in the Applications folder.

Using R for the rocket example

```
> height = c(100, 200, 300, 450, 600, 800, 1000)
> distance = c(253, 337, 395, 451, 495, 534, 574)
```

Plot the data set

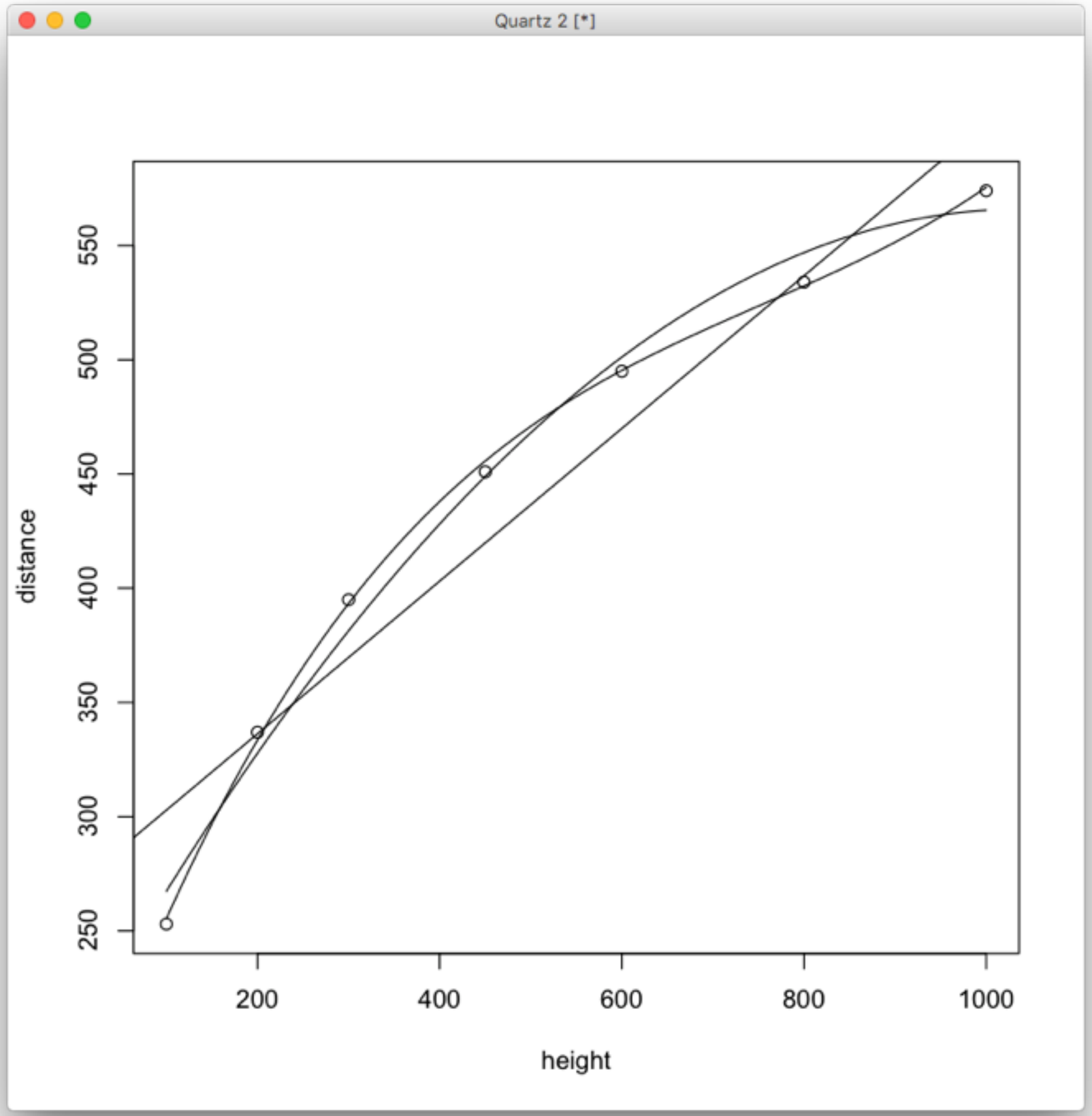
```
> plot(height, distance)
```

```
> model1 <- lm(distance ~ height); model1

Coefficients:
(Intercept)       height
   269.4661       0.3341

Try fitting a quadratic

> model2 <- lm(distance ~ height + I(height^2)); model2

Coefficients:
(Intercept)       height  I(height^2)
 200.211950     0.706182    -0.000341
```

```
> newh = seq(100, 1000, 10)
> fit2 = 200.211950 + 0.706182*newh -
0.000341*newh^2
> plot(height, distance)
> abline(model1)
> lines(newh, fit2, lty=1)
```

Repeat for a cubic

```
> model3 <- lm(distance ~ height + I(height^2) +
I(height^3)); model3
> fit3 = 1.555e+02 + 1.119*newh - 1.254e-03*newh^2 +
5.55e-07*newh^3
> lines(newh, fit3, lty=1)
```


Quartz 2 [*]
distance
250
300
350
400
450
500
550
200
400
600
800
1000
height

Would you choose the linear, quadratic or cubic model for this data? Does the information that this data comes from a rocket help in your choice?

Can you calculate the following for each of these three curves? Does this agree with your decision of the best model?

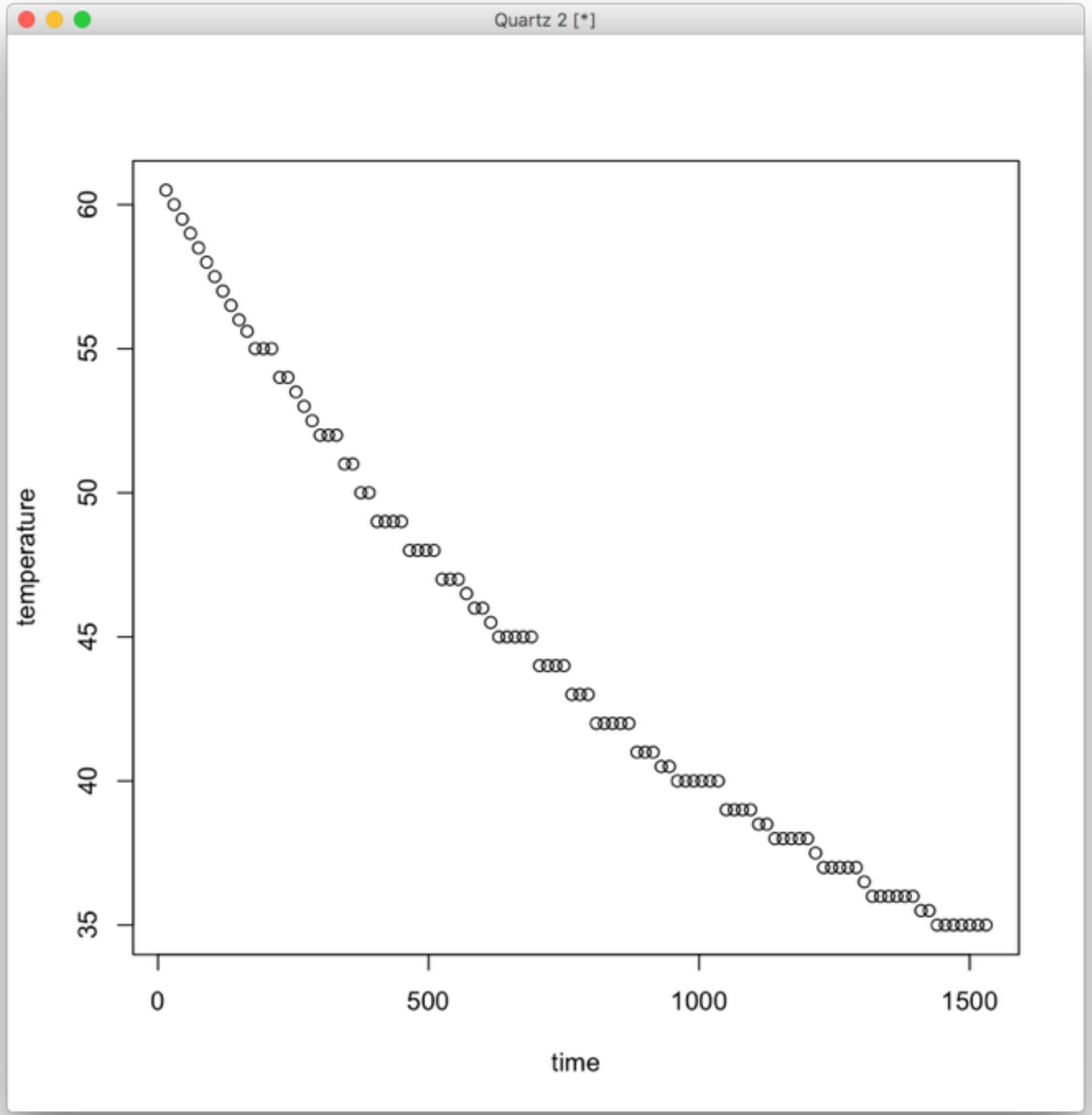
$$(\|\mathbf{e}_m\|_2)^2/(n - m - 1)$$

Netlogo example
(coffee mug cooling)

Start Netlogo and read in LinearRegression.nlogo. This model reads in the coffeMugCooling.csv data file. Press setup and then go. This shows the data and a least squares fit of a line to the data. The slope and intercept of the line are displayed. Do you think that a line is the right model for this data? If not, what would be a good model?

Using R for the coffee mug cooling example.

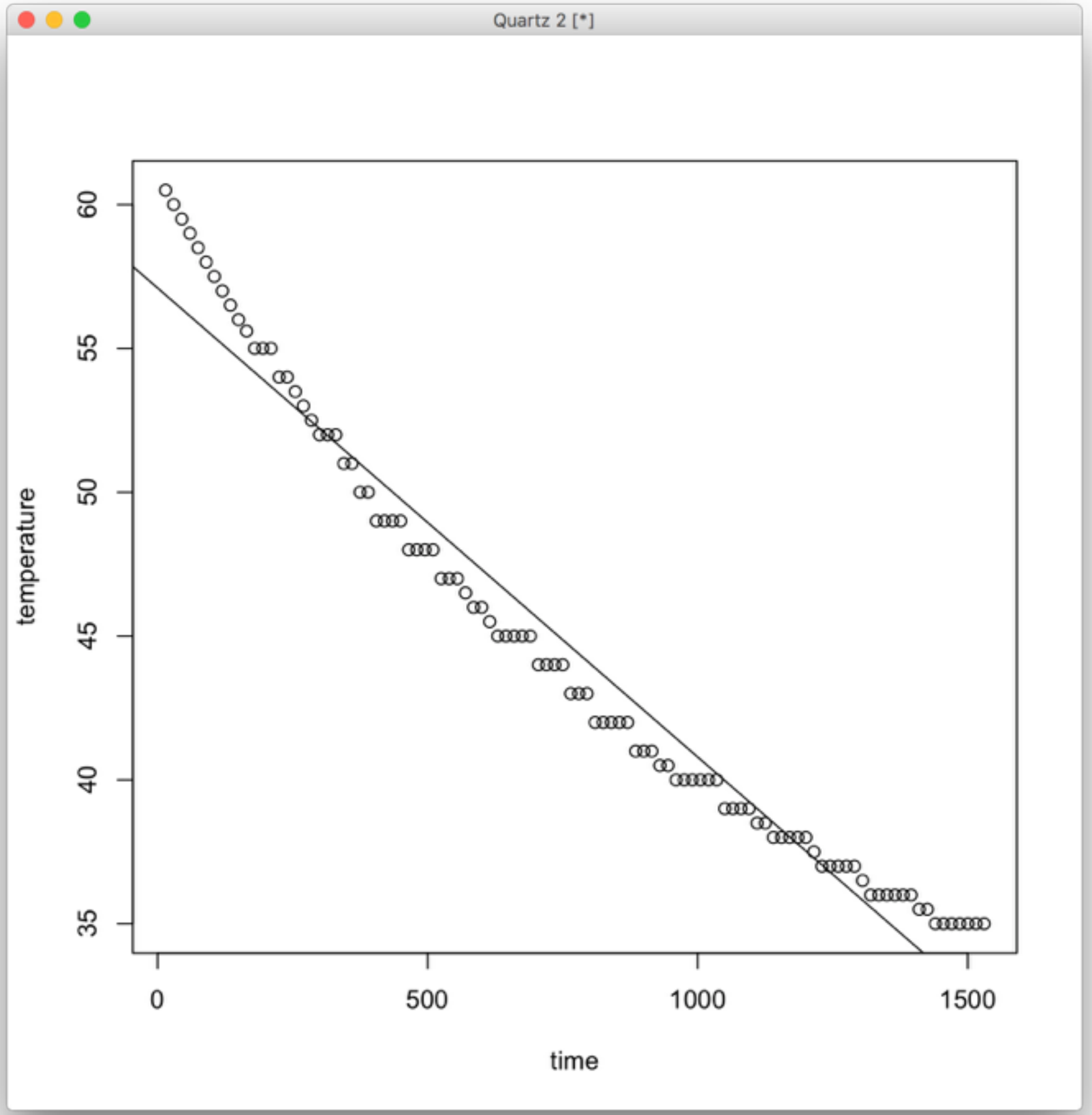
```
> mug <- read.csv("/<path>/coffeeMugCooling.csv");mug
> time = mug$X0
> temperature = mug$X61
> plot(time, temperature)
```


Quartz 2 [*]
temperature
60
55
50
45
40
35
0
500
1000
1500
time

```
> model1 = lm(temperature~time); model1

Coefficients:
(Intercept)         time
   57.10177     -0.01631
```

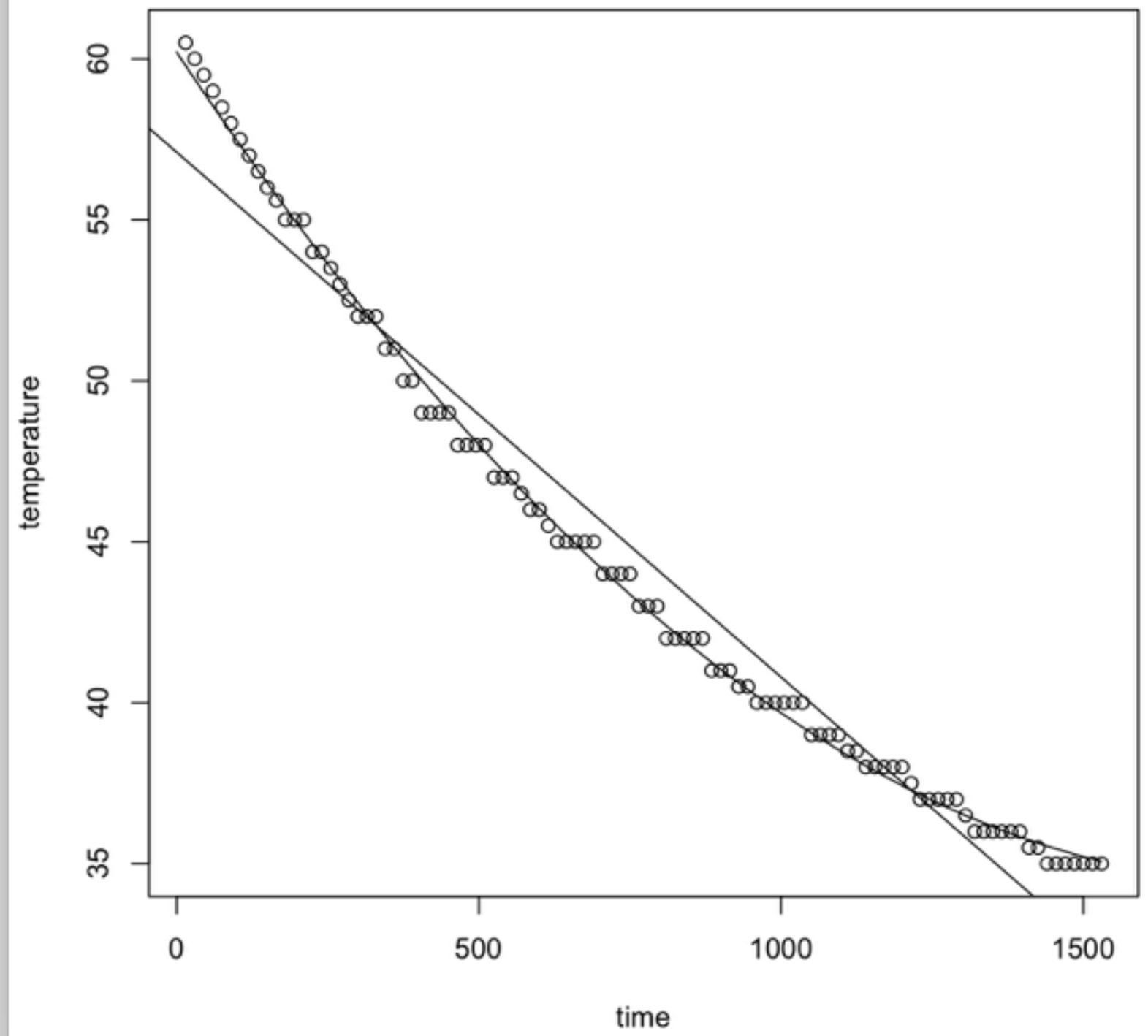
How do these compare to the Netlogo model?

```
>abline(model1)
```


Quartz 2 [*]
temperature
60
55
50
45
40
35
0
500
1000
1500
time

```
> model2 = lm(temperature~time+I(time^2)); model2

Coefficients:
(Intercept)         time    I(time^2)
  6.021e+01   -2.826e-02    7.733e-06

> t = seq(0, 1530, 10)
> temp = 60.21 - 0.02826*t + 7.733e-06*t^2
> lines(t, temp, lty=1)
```

Quartz 2 [*]



Data from Computer Models

So far we have considered the case of data that is given to us and creating a model of the data. This is useful for as data input into a computer model. But computer models also generate data. When generating data the question is how much data is enough. Running experiments that general data can post the same questions about the amount of data although budget or resources may limit the data.

Rules of thumb*

- Expectation, average, mean - 7 or 8 runs
- Standard deviation, linear model, expected range - 13 runs
- 1% (rare) events - hundreds of runs

*For data that is normally distributed. Other distributions will require more.

Mexican Wolves Model

For this example we will construct a pseudo-model of the Mexican Wolf population. Our model produces an estimate of the wolf population sometime in the future. To construct our model we randomly select a mean and standard deviation. The mean is somewhere between 30 and 150 wolves.

```
> mymean = runif(1, 30, 150)
> mystddev = runif(1, .7, 2.3) * mymean
> samples = round(rnorm(1000, mymean, mystddev))
```

The array samples contains data generated by the model. While the model does not capture much about the physical environment of the wolves but we know a lot about what data the model should produce.

Each student will now have their own model of the Mexican wolf population. The instructor may want to write down the values of mymean and mystddev for each student and then remove them

```
> rm(mymean)
> rm(mystddev)
```

The student then runs his/her model one time

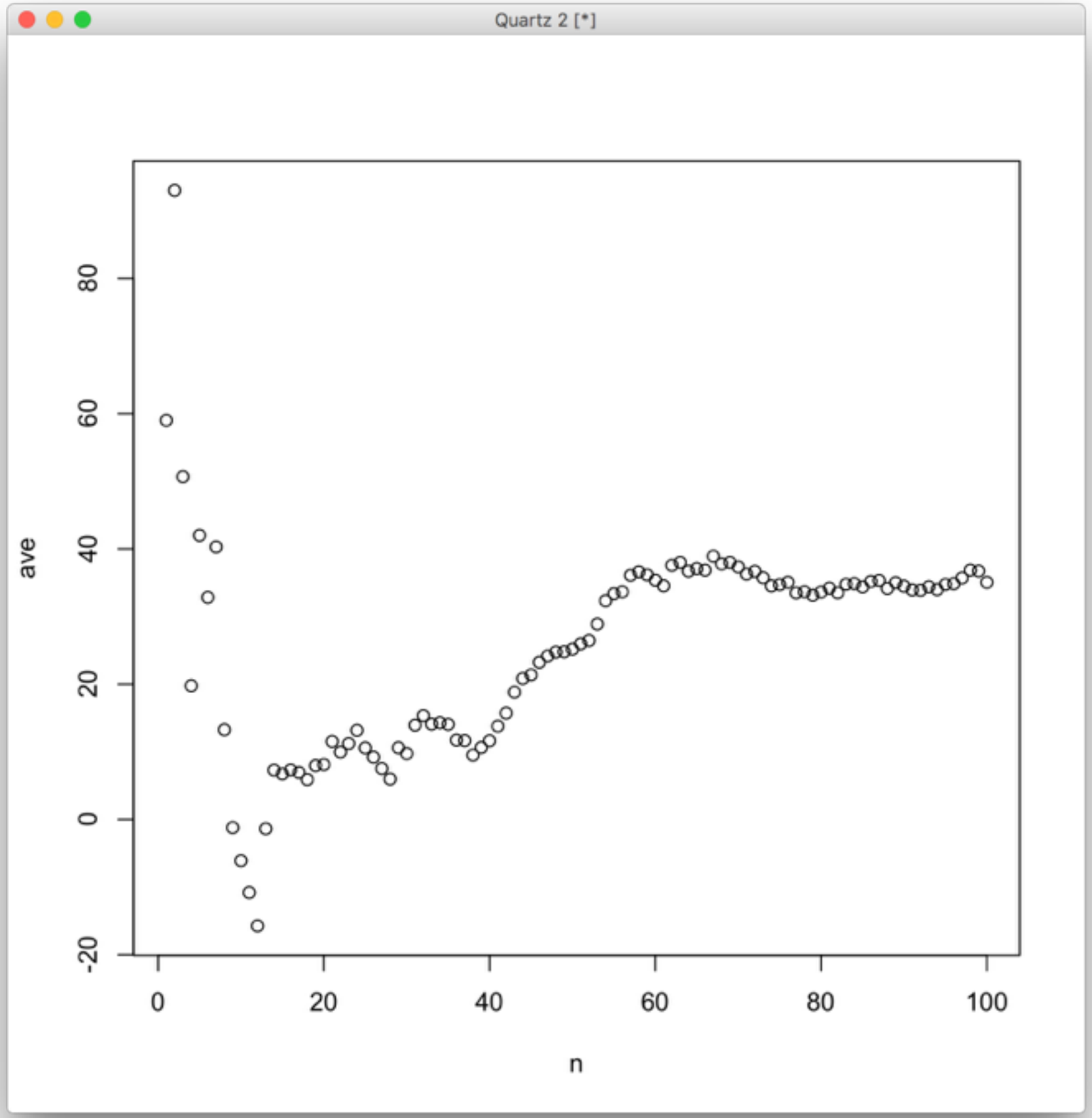
```
> samples[1]
```

This gives the number of wolves. The student should now write down their analysis of the results of their model. How likely is it that the wolves will survive?

Now run the model multiple times. samples[2] gives the second run. To get the results of ten runs type samples[1:10]. Does the interpretation of the model change as data is added?

```
> n = seq(1, 50, 1)
> ave = c()
> for (i in 1:50) {ave[i] = mean(samples[1:i])}
> plot(n, ave)
```

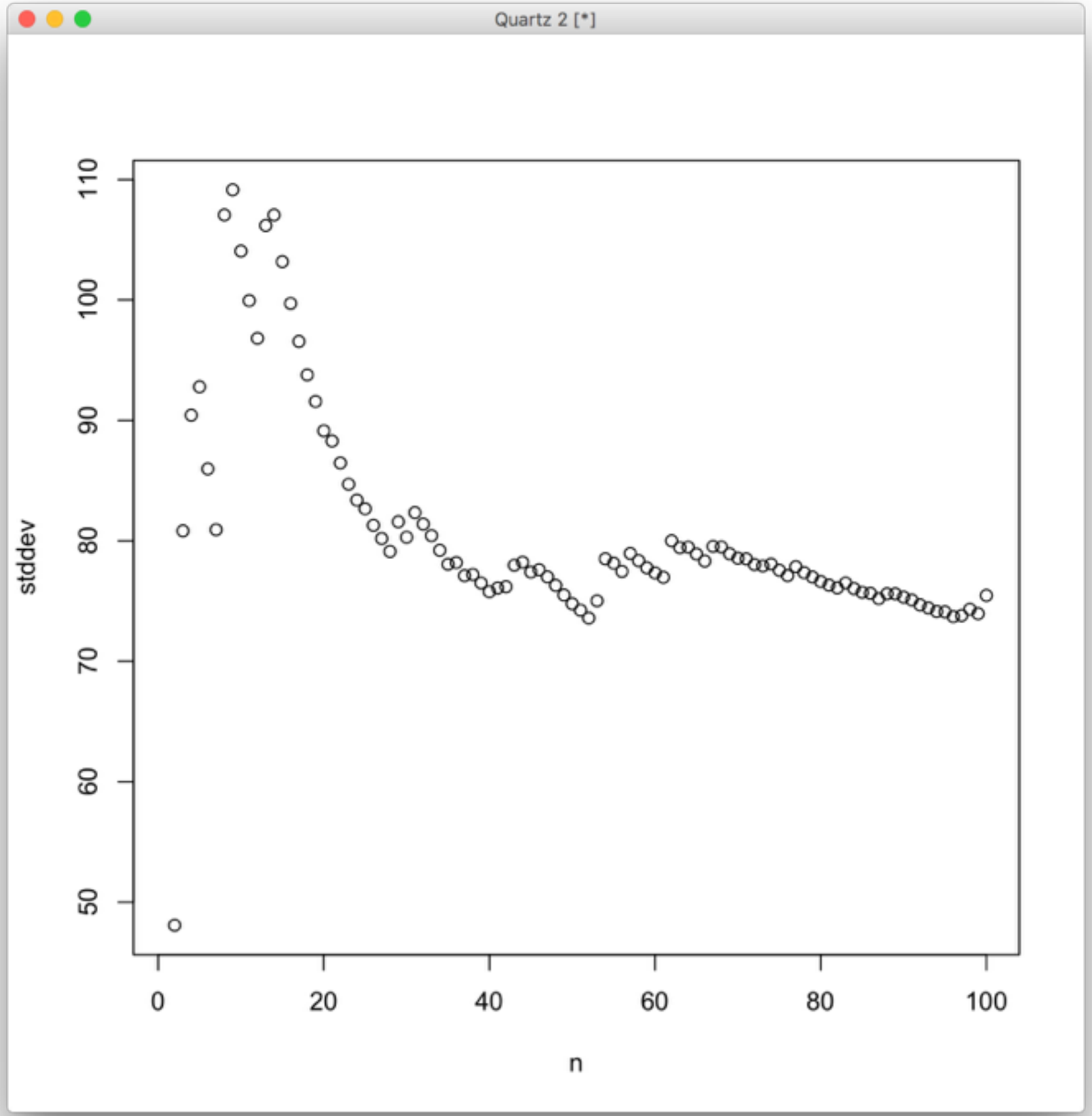
What is your estimate for the average? How many samples does it take to get a good estimate?


Quartz 2 [*]
ave
80
60
40
20
0
-20
0
20
40
60
80
100
n

Now repeat this using the standard deviation. The standard deviation is a measure of the variability of the output.

```
> n = seq(1, 100, 1)
> stddev = c()
> for (i in 1:100) {stddev[i] = sd(samples[1:i])}
> plot(n, stddev)
```

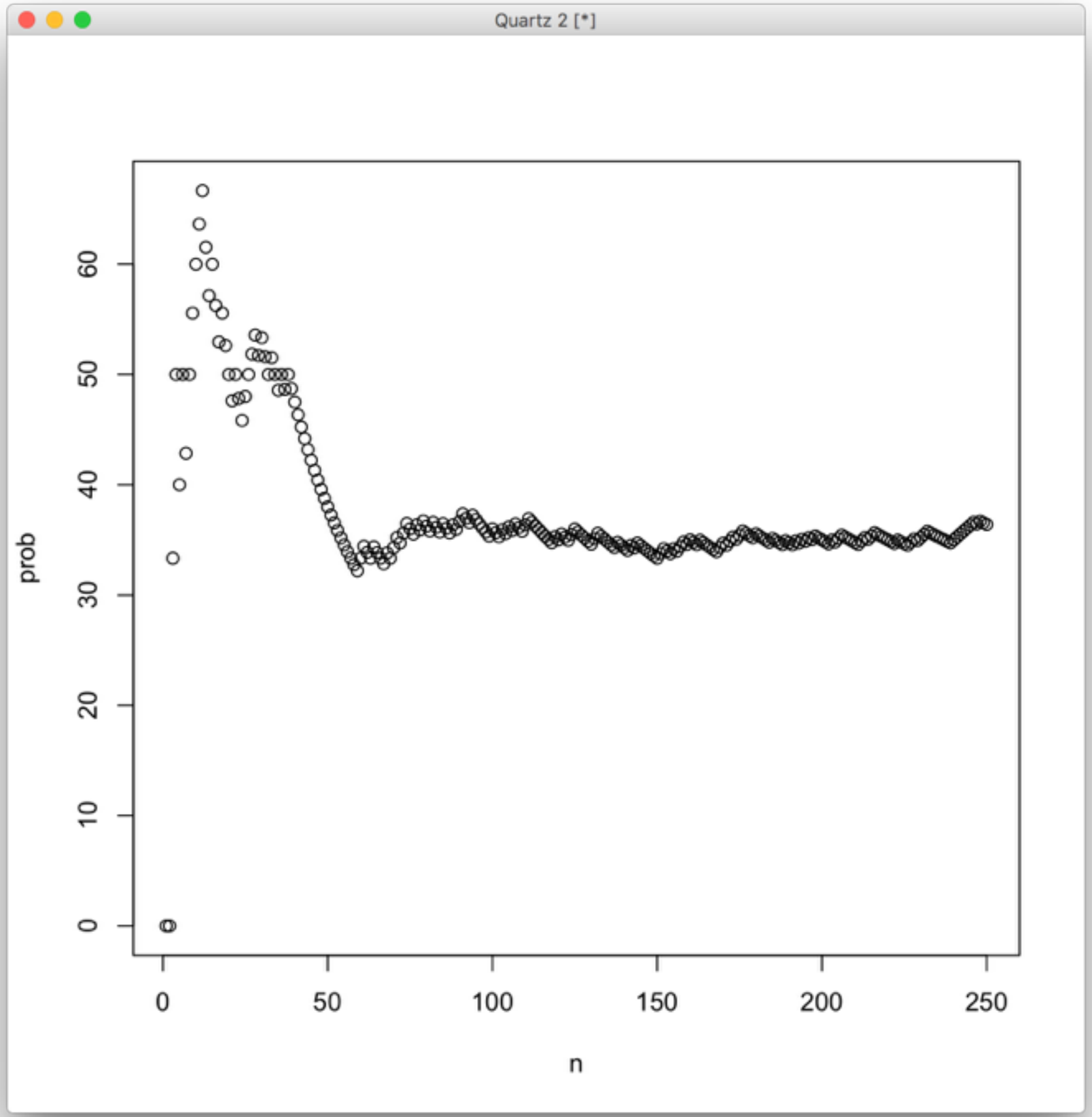
What is your estimate of the standard deviation? How many samples did it take to get a realistic estimate?


Quartz 2 [*]
stddev
110
100
90
80
70
60
50
0
20
40
60
80
100
n

Now calculate the probability that the wolf population goes extinct.

```
> n = seq(1, 250, 1)
> prob = c()
> for (i in 1:250) {prob[i] = 100.0 * sum(samples[1:i] < 0)/i}
> plot(n, prob)
```

What is your estimate of the probability that the wolf population goes extinct? How many samples did it take to get a reliable estimate of the probability? Lower probabilities will generally take more data. Why is this?


Quartz 2 [*]
prob
0
10
20
30
40
50
60
0
50
100
150
200
250
n

Now compare your estimates with mymean, mystddev and for the probability

> 100.0*pnorm(0, mymean, mystddev)

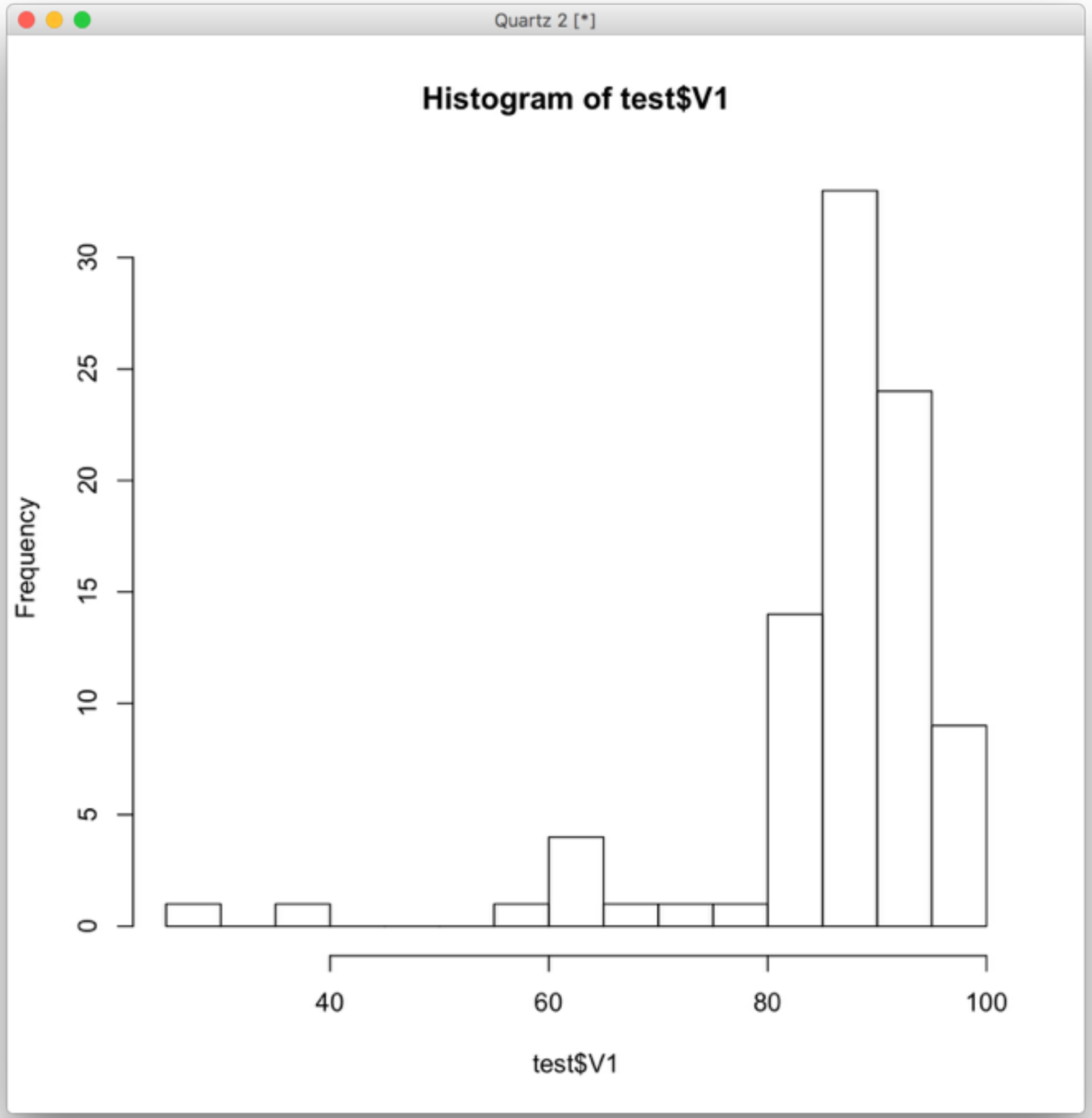
How close were your estimates?

Advanced example

A probability density function (pdf) describes the likelihood of a value. The first step is to plot the histogram and compare the shape to types of pdfs. If all values are equally likely then it is a uniform distribution. Test scores usually form a normal distribution (bell curve). Often looking at the domain gives a clue. If the domain is from [0, inf] then the log normal distribution may be a better choice than the normal distribution which has a range of [-inf, inf]. The data for this example is a set of test scores.

> scores <- read.table("/<path>/scores.txt")
> hist(scores$V1, 20)

Quartz 2 [*]

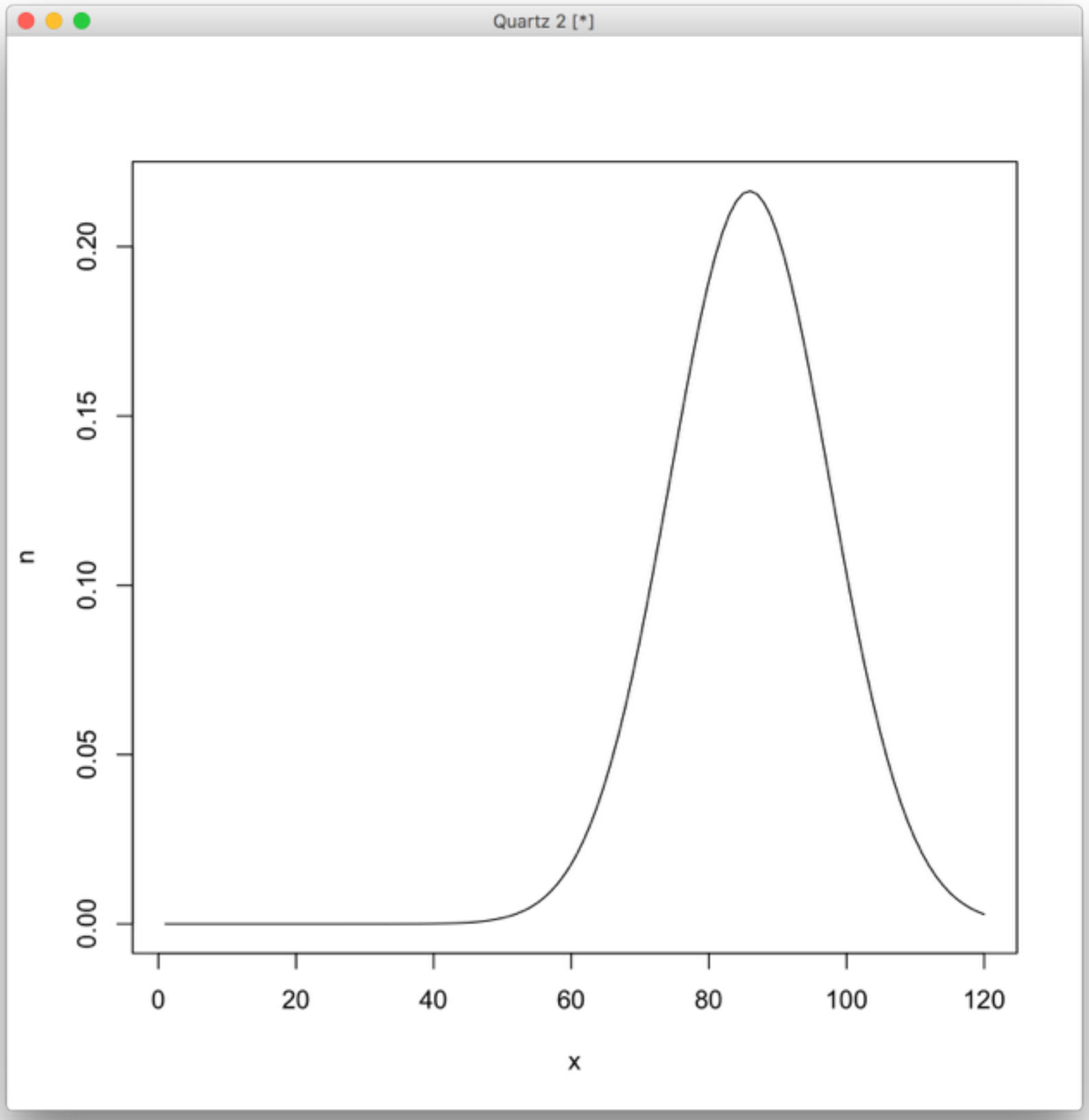

Histogram of test$V1
Frequency
0
5
10
15
20
25
30
40
60
80
100
test$V1

Least squares is not a good measure of fit for fitting probability distribution functions to data. If you have a f_i data point from the tail of the data then the fitted pdf has $p_i = 0$; this says there is no possibility of that value ever happening but it happened. $e_i = p_i - f_i$ is a small part of the overall error vector and the 2-norm so it is very possible the fitted pdf using least squares will have this happen. It may not be important if the problem does not care about the tails of the pdfs. But there are measures instead of using norms that can give better results. The MASS module for R uses a maximum likelihood estimation to fit a probability density function to a set of data.

The normal distribution is

$$y=\frac{1}{\sigma sqrt(2\pi)}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

```
> library("MASS")
> fitdistr(scores$V1, 'normal')

      mean          sd
  85.9200000   11.5811744
 ( 1.2207630) ( 0.8632098)

> x = seq(1, 120, 1)
> n = (1/11.5811744*sqrt(2*3.1415))*exp(-
((x-85.92)^2)/(2*11.5811744^2))
> plot(x, n, "l")
```


Quartz 2 [*]
n
0.00
0.05
0.10
0.15
0.20
0
20
40
60
80
100
120
x

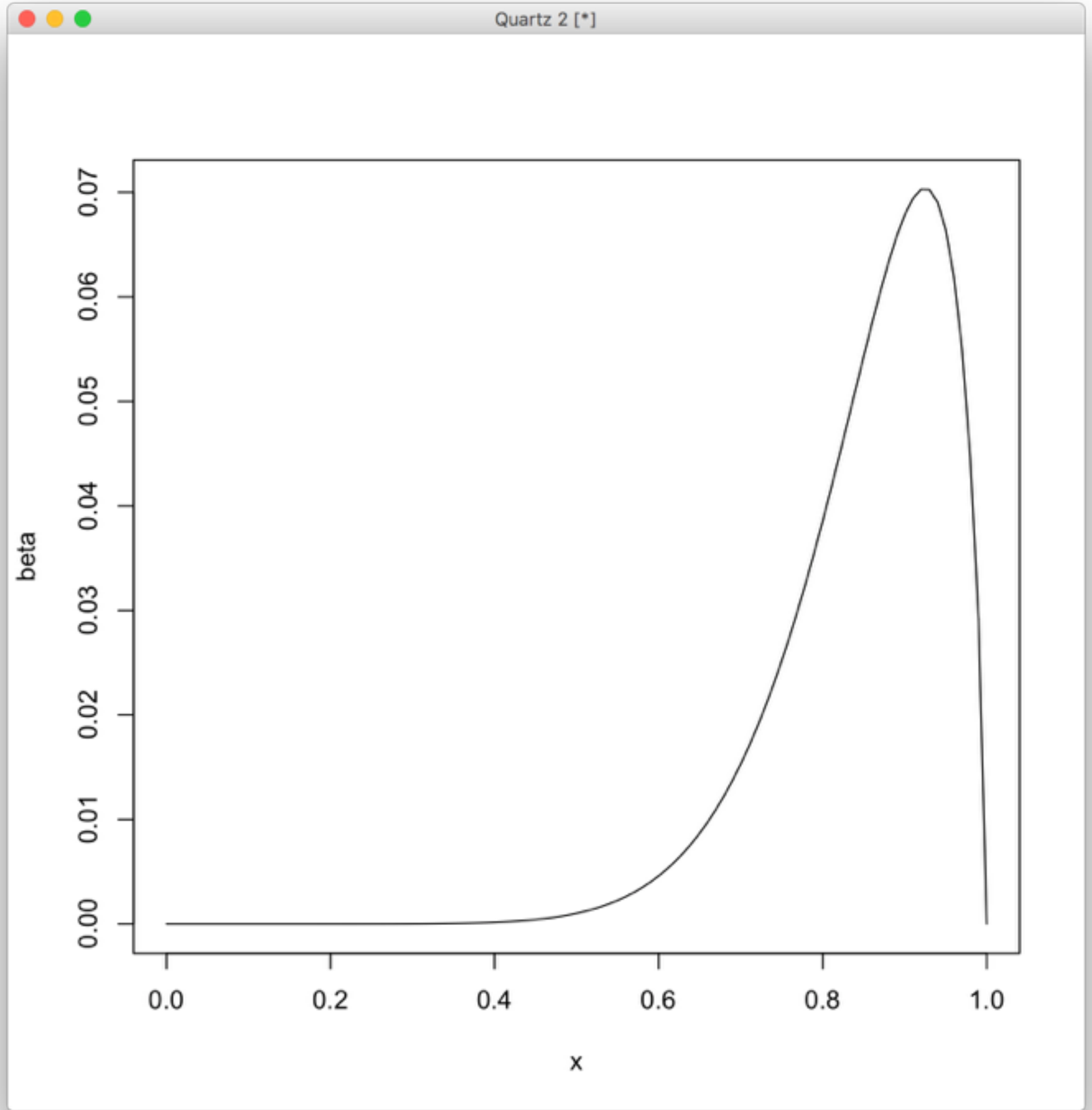
The domain of the normal distribution goes from [-inf, inf]. Is this realistic? Note that the plot of the normal distribution shows values greater than 100 are common. The beta distribution has the domain [0, 1] and can mimic the normal distribution and many other distributions.

$$y=Cx^{\alpha-1}(1-x)^{\beta-1}$$

To fit this we need to supply starting guesses for alpha (shape1) and beta (shape2). We also need to divide our test scores by 100 to get them into the domain [0, 1].

```
> starter = list('shape1'=10, 'shape2'=10)
> fitdistr(scores$V1/100, 'beta', starter)
```

```
> x = seq(0, 1, 0.01)
> beta = (x^(10.1983895-1))*((1-x)^(1.7475886-1))
> plot(x, beta, “l”)
```


Quartz 2 [*]
beta
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.0
0.2
0.4
0.6
0.8
1.0
x

Do you think that the normal distribution or the beta distribution is a better fit for the test scores? Why?

Appendix: Download R

Binary distributions are available for Windows and Mac OS X at https://cran.r-project.org/. Many Linux distributions have R available in the package management systems or check the link above.

On the Mac click on R.app in the Applications folder to start R.