Intro to R Programming: Lab 1

More on computers' mistakes

1. You are probably aware that there are two formulae for computing the variance of a sample:

$$s_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \bar{x}^2$$

   When performing calculations by hand you would probably use the right hand one. Which one should we use on a computer? We will simulate data with a large mean and a very small variance.

```
n <- 1000                              # Set sample size.
mu <- 1e7                              # Set mean to something very large.
sigma <- 1e-1                          # Set standard deviation.
x <- rnorm(n,mu,sigma)                 # Simulate the data.
```

The following code implements the two formulae above

```
sum((x-mean(x))^2)/n                   # Formula on left hand side.
sum(x^2)/n - mean(x)^2                 # Formula on right hand side.
```

Are the results equal? Compare these with the R built-in function `var`.

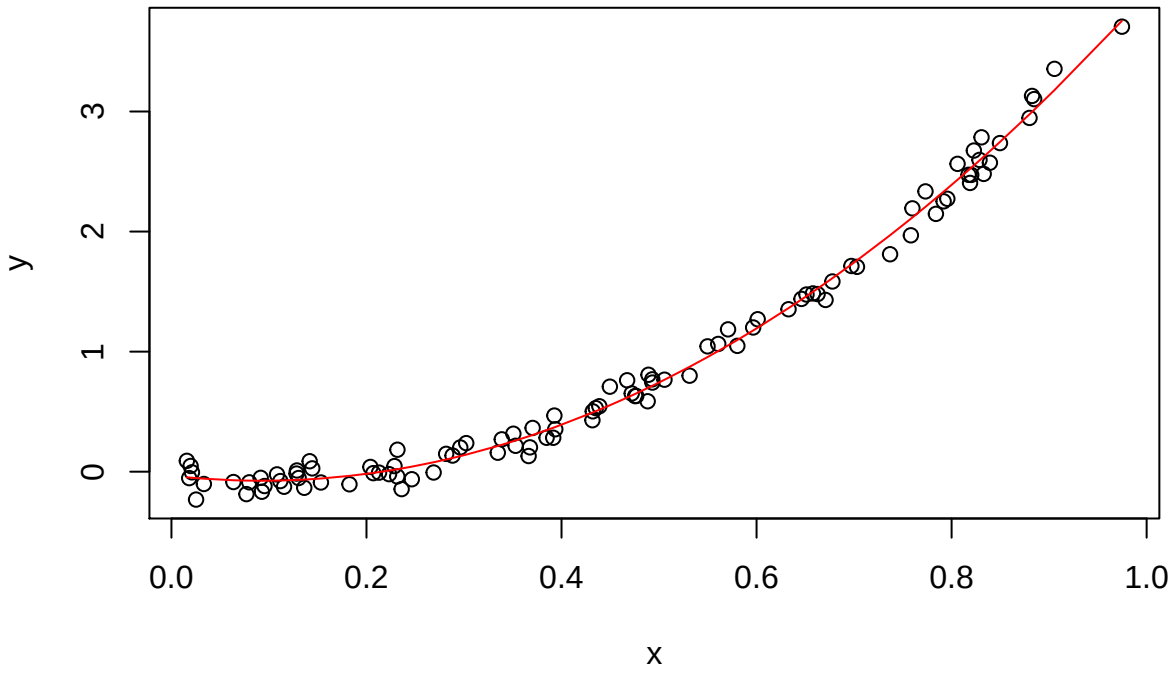
```
var(x)                                 # Use the built-in function.
```

2. Consider the following quadratic regression problem. We wish to use a model of the type

$$E(y_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2.$$

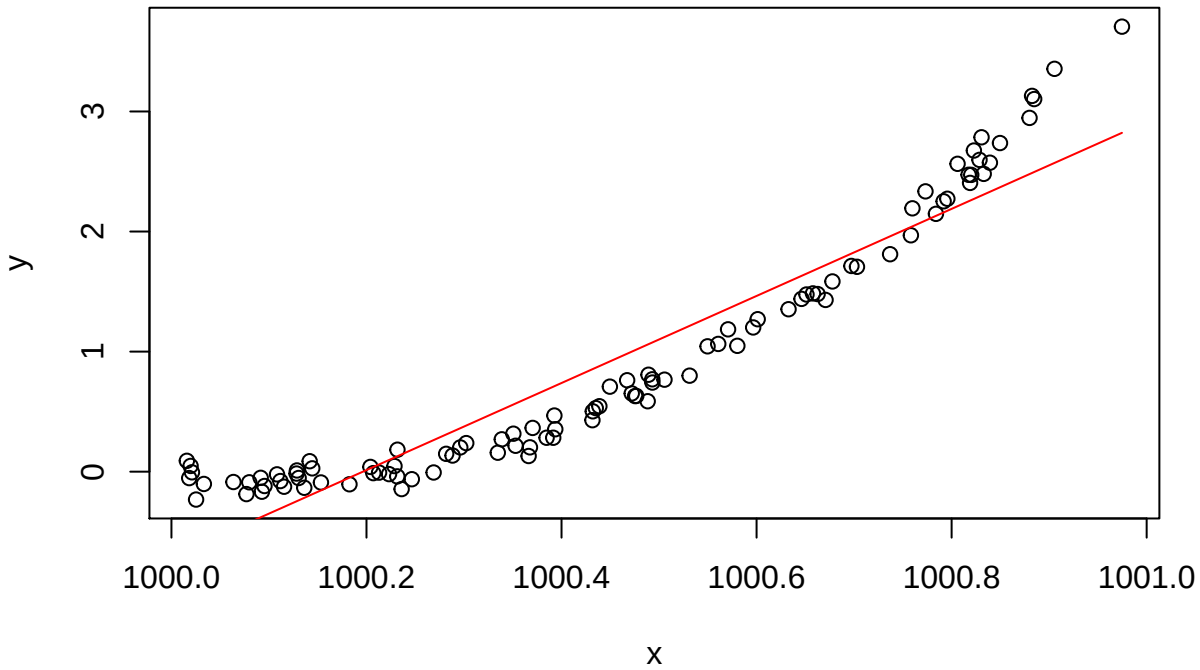
   Use the following code to fit a quadratic regression to the simulated data.

```
n <- 100
x <- sort(runif(n))                    # Create 100 values between 0 and 1.
y <- 5*x^2 - x + 0.1*rnorm(n)          # Create simulated response.
plot(x,y)                              # Plot the data.
reg.model <- lm(y~x+I(x^2))            # Fit the regression model.
lines(x,fitted(reg.model),col="red")   # Plot the fitted function.
```



What if we change the range of x from $[0, 1]$ to $[1000, 1001]$? According to the theory of the linear model the fitted values should be exactly the same. Try using the following code.

```
x <- x+1000                          # Add 1000 to each x.
plot(x,y)                            # Plot the data.
reg.model <- lm(y~x+I(x^2))          # Fit the regression model.
lines(x,fitted(reg.model),col="red")  # Plot the fitted function.
```



You can see that we run into numerical difficulties, as $X^{\mathrm{T}}X$ is almost a singular matrix, so computing the estimated regression coefficients $\hat{\beta}$ fails.

R as a calculator

3. Use R to compute $5 - \frac{7}{8}$, $\frac{5-7}{8}$, $\pi - \frac{333}{123}$, $256^{1/4}$, $\exp(1)$, $\left(1 + \frac{1}{1000}\right)^{1000}$, $\left(\frac{1}{56}\right)^{\frac{1}{4}}$.

4. The "Babylonian method" provides a way of approximating $\sqrt{2}$. The sequence defined recursively by

$$x_n = \frac{x_{n-1}}{2} + \frac{1}{x_{n-1}}$$

tends to $\sqrt{2}$ as $n \longrightarrow \infty$ (provided $x_0 > 0$).

- Define a new variable `x` taking the value 1.
- Update `x` to take the value

$$\frac{x}{2} + \frac{1}{x}.$$

- Repeat the update from the previous part. You should see that x tends to $\sqrt{2}$. (You will learn later on in this course how to use loops to do this more efficiently. If you already know how to use loops, feel free to use them in this question.)

Logical Variables

5. Consider two logical variables `a` and `b` generated using the code below

```
a <- sample(c(TRUE, FALSE), 1)
b <- sample(c(TRUE, FALSE), 1)
```

which randomly sets each of them to either `TRUE` or `FALSE`. Use the operators `&`, `|`, `!` and/or `==` to define a new variable `c` in terms of `a` and `b`, which is `TRUE` if `a` and `b` are either both `TRUE` or both `FALSE`. Otherwise `c` should be `FALSE`.

Can you think of more than one way of defining `c`?