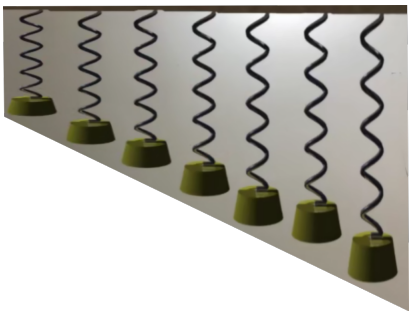
003 A Detailed Example

Ananda Biswas

Once we have the blackbox, we shall take multiple instances of the object, depending on the system we are working on.
Here we shall take multiple such springs. If the blackbox were patient, we would have taken multiple patients.
We have to make sure that the units(springs, patients) are as identical as possible. Some amount of variation will be there, that will be considered as part of random error, but we shall try to keep that as small as possible.



We are giving different weights(inputs) to the springs and for each of the cases we are measuring the length of the spring.
Here we have a covariate(as input weight w is continuous) and a continuous output length l.

Now we shall write our model, the mathematical formulation of the system.

$$l = \beta_1 + \beta_2 w + \epsilon$$

β_1 and β_2 are unknown constants, they are common properties of the identical strings.

Here we are expressing the output l as a linear function of input w and random error ϵ.

For n instances of the object, we shall have

$$l_i = \beta_1 + \beta_2 w_i + \epsilon_i, \quad \forall\, i = 1, 2, 3, \ldots n.$$

Converting our model in matrix form, we shall have,

$$\begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_n \end{bmatrix} = \begin{bmatrix} 1 & w_1 \\ 1 & w_2 \\ \vdots & \vdots \\ 1 & w_n \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}. \tag{1}$$

$\begin{bmatrix} 1 & w_1 \\ 1 & w_2 \\ \vdots & \vdots \\ 1 & w_n \end{bmatrix}$ is the **design matrix** *i.e.* it is designed by us, we exactly know the weights w_is that we gave as inputs.

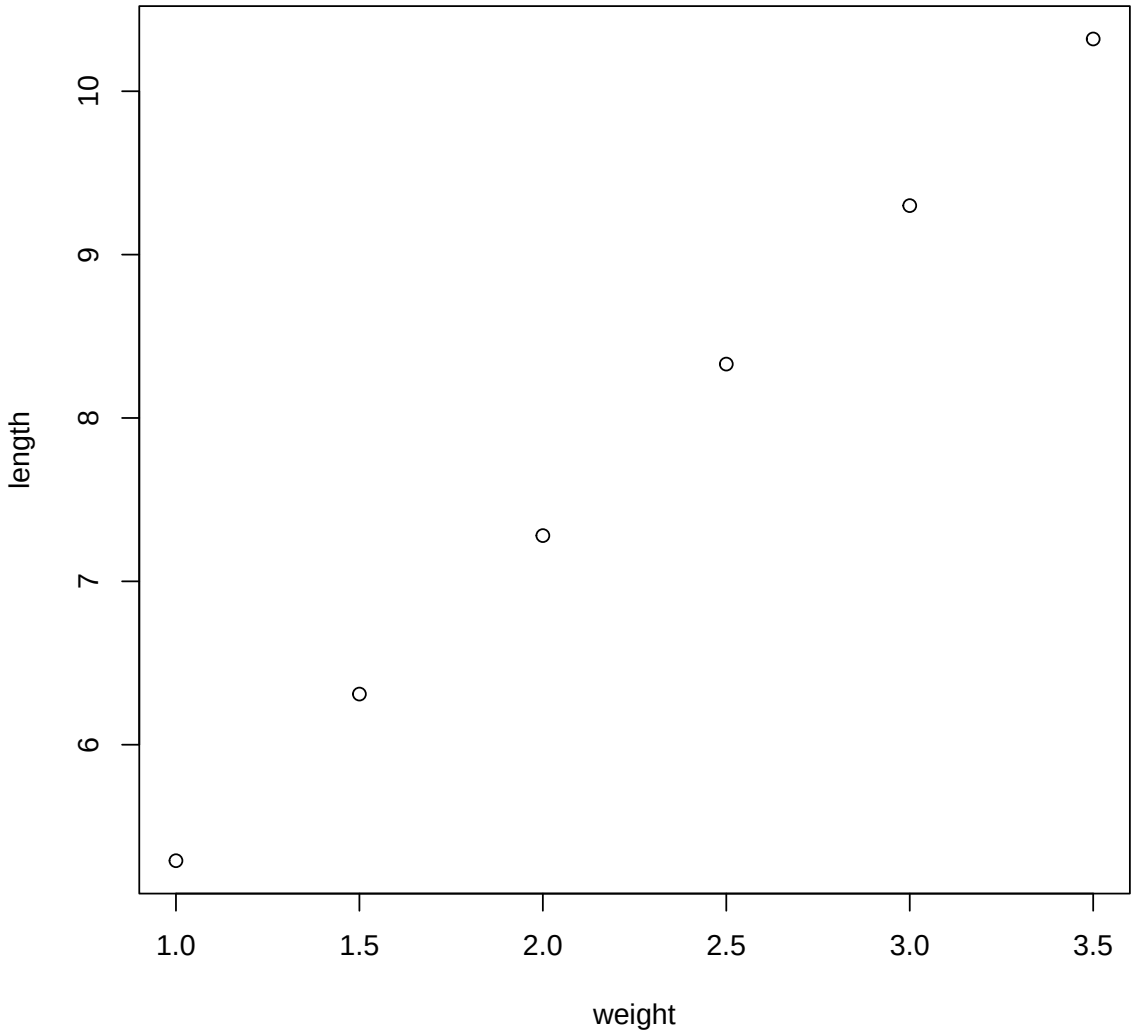
```
our_data = read.csv("springs.csv")

our_data
```

```
##   weight length
## 1    1.0   5.29
## 2    1.5   6.31
## 3    2.0   7.28
## 4    2.5   8.33
## 5    3.0   9.30
## 6    3.5  10.32
```

```
dim(our_data)
```

```
## [1] 6 2
```

```
plot(length ~ weight, data = our_data)
```




```
fit = lm(length ~ weight, data = our_data)
```

```
model.matrix(fit)

##   (Intercept) weight
## 1           1    1.0
## 2           1    1.5
## 3           1    2.0
## 4           1    2.5
## 5           1    3.0
## 6           1    3.5
## attr(,"assign")
## [1] 0 1
```

```
fit
```

```
##
## Call:
## lm(formula = length ~ weight, data = our_data)
##
## Coefficients:
## (Intercept)       weight
##       3.283        2.010
```

```
names(fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
fit$coefficients
```

```
## (Intercept)      weight
##    3.283143    2.009714
```

```
fit$residuals
```

```
##            1            2            3            4            5            6
## -0.002857143  0.012285714 -0.022571429  0.022571429 -0.012285714  0.002857143
```

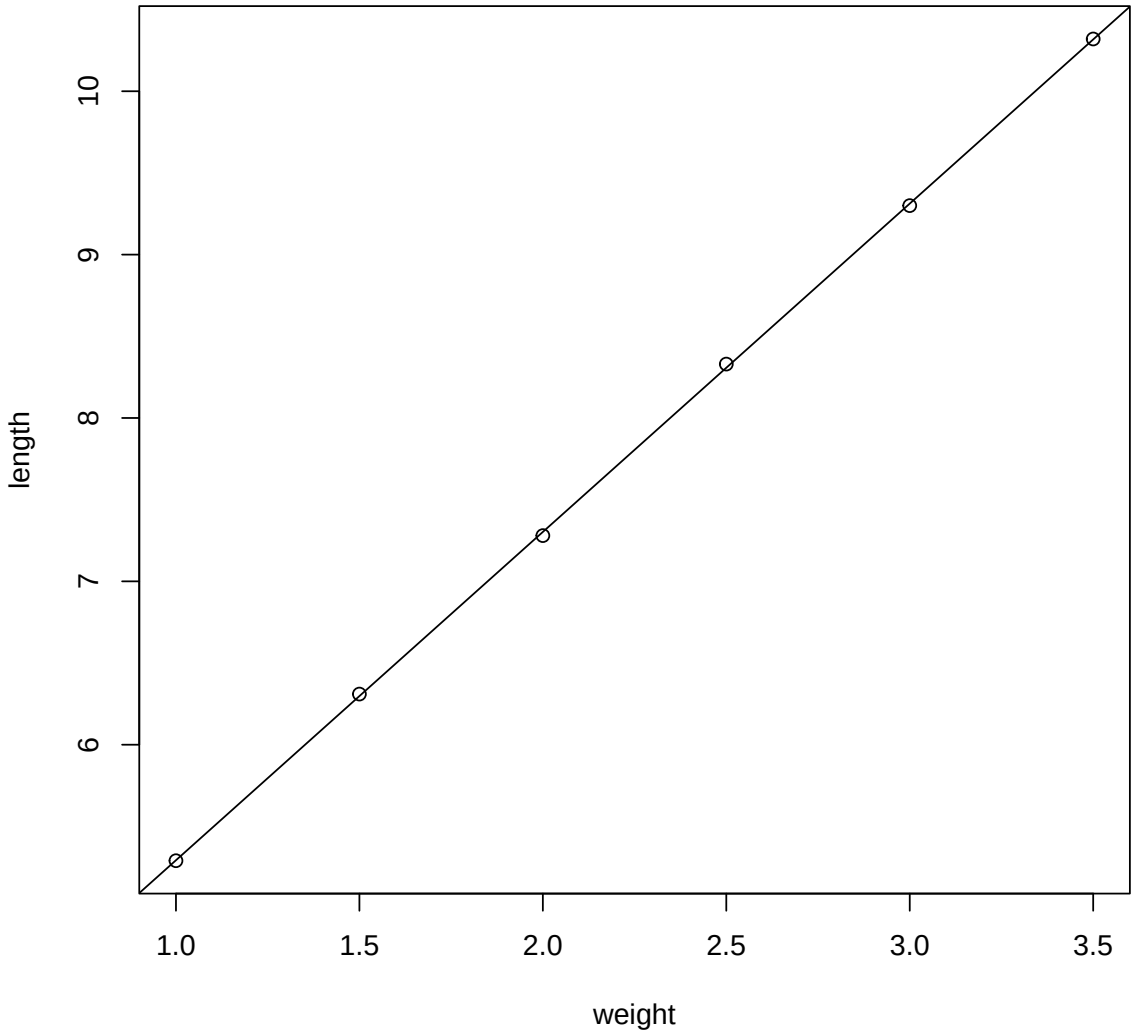
```
fit$fitted.values
```

```
##         1         2         3         4         5         6
##  5.292857  6.297714  7.302571  8.307429  9.312286 10.317143
```

```
fit$rank
```

```
## [1] 2
```

```
plot(length ~ weight, data = our_data)
abline(fit$coefficients)
```



The Important Steps

(1) Draw the blackbox

(2) Make n sets of measurements

(3) Create a CSV file. No. of columns = no. of inputs + 1. In the columns, we shall have the inputs and the output. No. of rows will be n.

(4) Load the file in R.

(5) Do some sanity check on the data set.

(6) Plot and explore.

(7) fit = lm(output $\sim$ ..., name of data set)

(8) Explore fit.

(9) Assess goodness. Simplest way to do so is to plot the fitted line.

Quadratic Regression

In the same example with springs, let our model be

$$l_i = \beta_1 + \beta_2 w_i + \beta_3 {w_i}^2 + \epsilon_i, \quad \forall\, i = 1, 2, 3, \ldots n.$$

In matrix form,

$$\begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_n \end{bmatrix} = \begin{bmatrix} 1 & w_1 & {w_1}^2 \\ 1 & w_2 & {w_2}^2 \\ \vdots & \vdots & \vdots \\ 1 & w_n & {w_n}^2 \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}. \tag{2}$$

```
our_data = read.csv("springs.csv")
```

```
fit_quad = lm(length ~ weight + I(weight^2), our_data)
```

```
fit_quad

##
## Call:
## lm(formula = length ~ weight + I(weight^2), data = our_data)
##
## Coefficients:
## (Intercept)       weight  I(weight^2)
##   3.283e+00    2.010e+00    7.245e-16
```

See that, the coefficient of $weight^2$ is close to 0; and the scatterplot of the data set also justifies this.

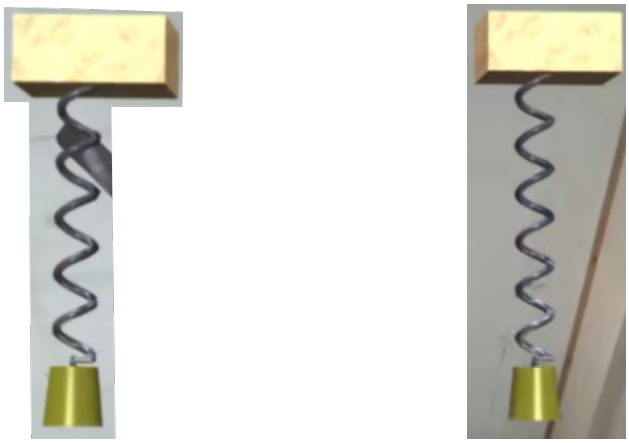
```
model.matrix(fit_quad)

##   (Intercept) weight I(weight^2)
## 1           1    1.0        1.00
## 2           1    1.5        2.25
## 3           1    2.0        4.00
## 4           1    2.5        6.25
## 5           1    3.0        9.00
## 6           1    3.5       12.25
## attr(,"assign")
## [1] 0 1 2
```

- In the lm() function, **"I(weight $\wedge$ 2)"** is important to form the model matrix properly.

Another Example

Suppose we have 2 springs from 2 labs. We want to study the elastic property of the springs. The material of the two springs are same and their shapes are also same.

The lab from which a spring come may not be a very important factor, but its importance is too little to neglect. So we shall consider the lab as an input to our system.



Observe that, here "lab" is a control factor.

For the springs from Lab 1 and Lab 2, we have the following weights as inputs and lengths as outputs respectively.

W	L
1.0	5.29
1.5	6.31
2.0	7.28
2.5	8.33
3.0	9.30
3.5	10.32

W	L
1.2	7.60
1.5	8.11
1.8	8.88
2.1	9.40
2.1	9.39

Now we shall load the two data sets in R.

```
data_lab_1 = read.csv("springs_lab_1.csv")

data_lab_1
```

```
##   weight length
## 1    1.0   5.29
## 2    1.5   6.31
## 3    2.0   7.28
## 4    2.5   8.33
## 5    3.0   9.30
## 6    3.5  10.32
```

```
dim(data_lab_1)
```

```
## [1] 6 2
```

```
data_lab_2 = read.csv("springs_lab_2.csv")
```

```
data_lab_2
```

```
##   weight length
## 1    1.2   7.60
## 2    1.5   8.11
## 3    1.8   8.88
## 4    2.1   9.40
## 5    2.1   9.39
```

```
dim(data_lab_2)
```

```
## [1] 5 2
```

We have to merge them to do operations.

```
temp_1 = data.frame(data_lab_1, lab = 1)
```

```
temp_1
```

```
##   weight length lab
## 1    1.0   5.29   1
## 2    1.5   6.31   1
## 3    2.0   7.28   1
## 4    2.5   8.33   1
## 5    3.0   9.30   1
## 6    3.5  10.32   1
```

```
temp_2 = data.frame(data_lab_2, lab = 2)
```

```
temp_2
```

```
##   weight length lab
## 1    1.2   7.60   2
## 2    1.5   8.11   2
## 3    1.8   8.88   2
## 4    2.1   9.40   2
## 5    2.1   9.39   2
```

```r
all_data = rbind(temp_1, temp_2)

all_data
```

```
##    weight length lab
## 1     1.0   5.29   1
## 2     1.5   6.31   1
## 3     2.0   7.28   1
## 4     2.5   8.33   1
## 5     3.0   9.30   1
## 6     3.5  10.32   1
## 7     1.2   7.60   2
## 8     1.5   8.11   2
## 9     1.8   8.88   2
## 10    2.1   9.40   2
## 11    2.1   9.39   2
```

Here, our model will be

$$l_{ij} = \alpha_i + \beta w_{ij} + \epsilon_{ij}$$

where w_{ij} is the $j-th$ weight in $i-th$ lab.

For lab 1, we have a constant α_1 and for lab 2, we have α_2.

As the springs are almost identical, we have a constant β, same for both the springs.

Here, for $i = 1$, $j = 1(1)6$ and for $i = 2$, $j = 1(1)5$.

In matrix form,

$$\begin{bmatrix} l_{11} \\ l_{12} \\ \vdots \\ l_{16} \\ l_{21} \\ \vdots \\ l_{25} \end{bmatrix} = \begin{bmatrix} 1 & 0 & w_{11} \\ 1 & 0 & w_{12} \\ \vdots & \vdots & \vdots \\ 1 & 0 & w_{16} \\ 0 & 1 & w_{21} \\ \vdots & \vdots & \vdots \\ 0 & 1 & w_{25} \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta \end{bmatrix} + \begin{bmatrix} \epsilon_{11} \\ \epsilon_{12} \\ \vdots \\ \epsilon_{16} \\ \epsilon_{21} \\ \vdots \\ \epsilon_{25} \end{bmatrix}. \tag{3}$$

We have to change the "lab" to a factor.

```r
all_data$lab = factor(all_data$lab)
```

```r
class(all_data)
```

```
## [1] "data.frame"
```

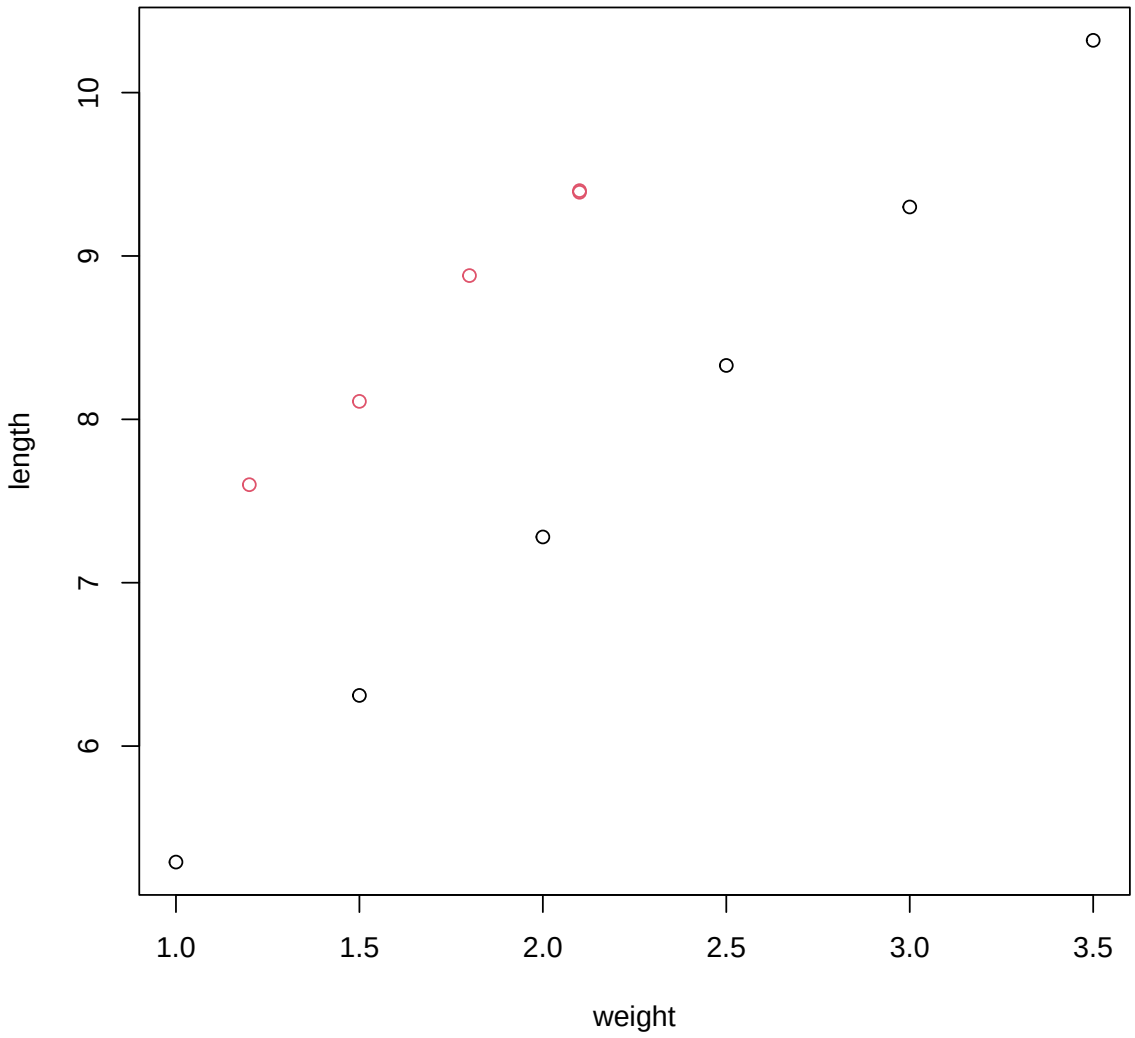
```r
class(all_data$weight)
```

```
## [1] "numeric"
```

```r
class(all_data$length)
```

```
## [1] "numeric"

class(all_data$lab)

## [1] "factor"
```

```
plot(length ~ weight, all_data, col = lab)
```



Now we shall fit our linear model.

```
fit_2 = lm(length ~ lab + weight - 1, all_data)
```

```
fit_2

##
```

```
## Call:
## lm(formula = length ~ lab + weight - 1, data = all_data)
##
## Coefficients:
##   lab1    lab2  weight
##  3.276   5.173   2.013
```

```
model.matrix(fit_2)

##    lab1 lab2 weight
## 1     1    0    1.0
## 2     1    0    1.5
## 3     1    0    2.0
## 4     1    0    2.5
## 5     1    0    3.0
## 6     1    0    3.5
## 7     0    1    1.2
## 8     0    1    1.5
## 9     0    1    1.8
## 10    0    1    2.1
## 11    0    1    2.1
## attr(,"assign")
## [1] 1 1 2
## attr(,"contrasts")
## attr(,"contrasts")$lab
## [1] "contr.treatment"
```

2 Problems

1. How to estimate β when α_1 and α_2 are known ?
2. How to estimate β and α_1 when α_2 is known ?