301107 - Analytics Programming

Nick Tothill

Practical class 02, week 03 - R data structures

In this class, you will familiarise yourselves some basic R data structures and learn a few ways to manipulate them to get the results you need.

1. Generate a vector of length 20 sampled from a normal distribution with mean 0 and standard deviation (sd) 1 (the function you need is `rnorm()`. You can access the help page with `help(rnorm)`. Here is one such vector:

```
 [1]  1.69734022 -1.76946135 -0.32203377 -1.17504365 -0.65729986 -2.24762667  0.48651801
 [8] -0.91165682 -1.52732050  0.91338578  0.13556346 -1.63381026 -0.26077803  0.03595732
[15] -0.63526510  0.21294543  1.66424042 -0.04615740 -0.55399795 -0.68341268
```

Every time you run or re-run `rnorm()`, you'll get different numbers in the vector.

2. Print the number of positive values and negative values and flip the negative values to positive values (the functions you need are `sum()`,`print()`,`paste()` or simply `cat()`)

```
There are 7 positive ones and 13 negative ones in the vector
Now the vector looks like after flipping negative values:

 [1] 1.69734022 1.76946135 0.32203377 1.17504365 0.65729986 2.24762667 0.48651801
 [8] 0.91165682 1.52732050 0.91338578 0.13556346 1.63381026 0.26077803 0.03595732
[15] 0.63526510 0.21294543 1.66424042 0.04615740 0.55399795 0.68341268
```

Hint: There are various ways of doing this correctly. A particular useful one is to use filtering and data type conversion.

3. Divide the values in the vector into 3 equal-sized bins and count how many values fall into each bin (the functions you may need are `seq()`,`max()`,`min()`, `range()` …)

```
Borders of the bins:

[1] 0.03595731 0.77318043 1.51040356 2.24762668

bin 1 bin 2 bin 3
   11     3     6
```

Hint: There is a very simple implementation of this task which is using the *factors* and related function `cut()`.

4. Fill this vector into a matrix with 3 rows and 6 columns. Observe how the matrix is filled and excessive values are discarded.

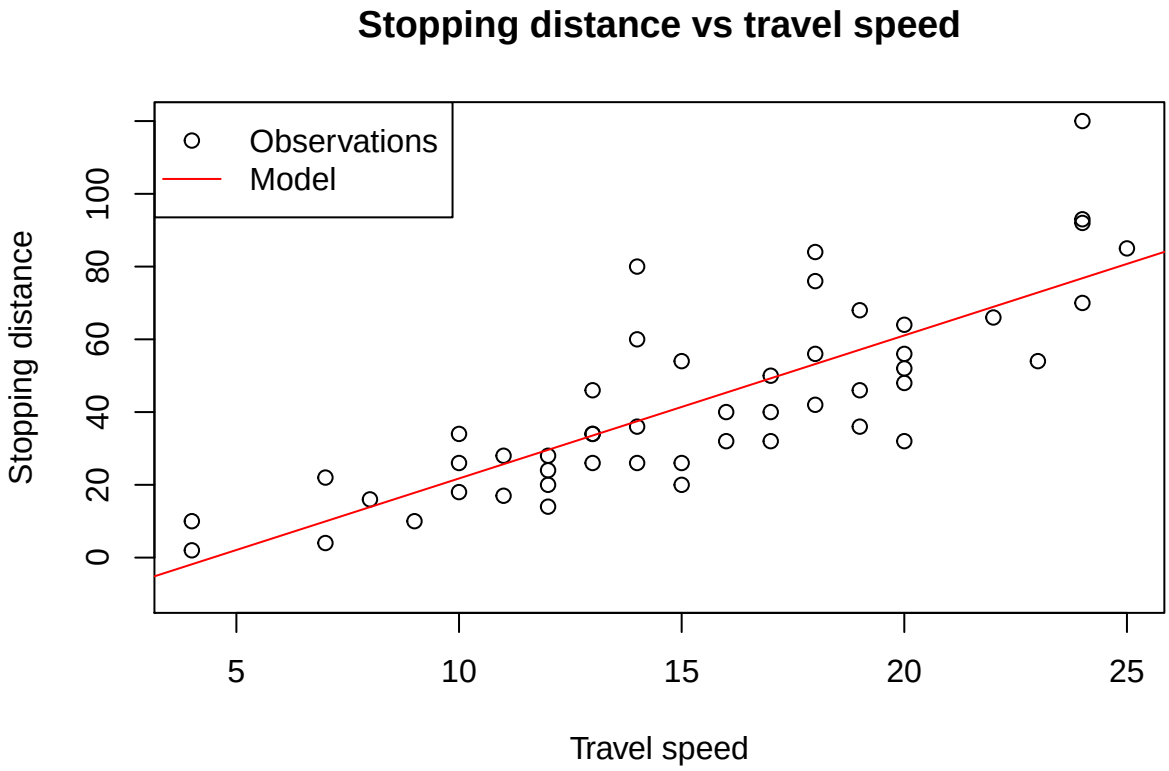
```
          [,1]      [,2]      [,3]      [,4]       [,5]      [,6]
[1,] 1.6973402 1.1750437 0.4865180 0.9133858 0.26077803 0.2129454
[2,] 1.7694614 0.6572999 0.9116568 0.1355635 0.03595732 1.6642404
[3,] 0.3220338 2.2476267 1.5273205 1.6338103 0.63526510 0.0461574
```

Fill in the matrix by rows instead like the following (the function you may need is `t()`):

```
         [,1]       [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 1.697340 1.76946135 0.3220338 1.1750437 0.6572999 2.2476267
[2,] 0.486518 0.91165682 1.5273205 0.9133858 0.1355635 1.6338103
[3,] 0.260778 0.03595732 0.6352651 0.2129454 1.6642404 0.0461574
```

5. Linear regression. Use data in `cars` data set, build a linear regression model. Here is some `cars` data

```
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```



From the above speed against stopping distance plot, we see that the stopping distance is almost linearly related to travel speed. As such we try to regress distance onto speed to see how this linear model performs.

The model is the following

$$Y = wX + b + E$$

where $Y,\ X,\ E$ are random variables for stopping distance, travel speed, and error, $w,\ b$ are model parameter. In statistics, Y is called *response variable* and X is called *explanatory variable*. Sometimes there may be more than one explanatory variable, in which case the model becomes

$$Y = w_1X_1 + w_2X_2 + \ldots + w_dX_d + b + E$$

where X_i for $i = 1, \ldots d$ is the ith explanatory variable and d is the number of total explanatory variables. We need to obtain the values for w and b using our observations. There are many ways to do this. One widely used is *the least squares (LS)*. The LS solution is given by

$$[w,\ b] = \mathbf{X}^{+}\mathbf{y}$$

where

$$\mathbf{X} = \begin{bmatrix} x_1, 1 \\ x_2, 1 \\ x_3, 1 \\ \vdots, \vdots \\ x_n, 1 \end{bmatrix}$$

$\mathbf{y} = [y_1, y_2, \ldots, y_n]$ and $\mathbf{X}^+ = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$ is the pseudo inverse of matrix $\mathbf{X}$. $x_1, \ldots, x_n$ are the observaions of speed and likewise $y_1, \ldots, y_n$ are observations of distance in `cars` data set. In this task, you have to estimate w and b by using LS and plot the figure as shown above. You may use `plot()`, `abline()`, `legend()` to produce the figure.

There is a very easy and simple way to do this by using `lm()` which uses *data frame* and *formula* as input. Try to use `lm()` to get the same results.