Augmented Coefficient Matrices

Learning Outcomes

At the end of this session, the students should be able to:

1. convert linear systems to matrices;
2. perform functions for string manipulation in R; and
3. create R scripts in converting linear systems to augmented coefficient matrices.

Content

Linear Systems

A system of linear equations of n variables is a collection of linear equations involving the same set of n variables. A solution to such system is the set of values for each of the n variables where the system is simultaneously satisfied. Therefore, the equations in the system is to be treated collectively, not individually.

The general form of a system of linear equation is of the following:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \quad & \quad \vdots \quad \quad \vdots \quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m. \end{aligned}$$

Where a = constant coefficients, b =constants, n = number of variables and m = number of equations.

This general form may be transformed into a matrix form, collecting all n variables and coefficients to be the following:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

As observed, the system will generate a vector x of variables, a vector b of right-hand sides and a square matrix A of rank n or dimensions $n \times n$. An augmented coefficient matrix $(A \mid B)$ is an $n \times n+1$ matrix which has the vector B added as the last column of matrix A. As such:

$$(A \mid B) = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right].$$

Example

We are given the following system of linear equations:

$8x_1 - 3x_2 + 0.9x_3 = 12$

$0.6x_1 - 0.3x_2 + 0.1x_3 = -45$

$20x_1 + x_3 = 46$

We can construct the augmented coefficient matrix to be:

$$\left[\begin{array}{cccc} 8 & -3 & 0.9 & 12 \\ 0.6 & -0.3 & 0.1 & -45 \\ 20 & 0 & 1 & 46 \end{array}\right]$$

String Manipulation in R

Concatenating Strings

Strings in R are combined using the **paste()** function. It can combined any number of arguments.

```
paste(..., sep = "  ", collapse = NULL)
                … represents any number of arguments to be combined.
                sep represents any separator between the arguments. It is optional
                collapse is used to eliminate the space in between two strings
```

Example :

```
print(paste("Hello","World",sep=""))          #The output is "HelloWorld"
print(paste("Hello","World",sep="-"))         #The output is "Hello-World"
```

Formatting numbers & strings

Numbers and strings can be formatted to a specific style using **format()** function.

```
format(x, digits, nsmall, scientific, width, justify = c("left", "right", "centre", "none"))
                x is the vector input
                digits is the number of digits displayed
                nsmall is the minimum number of digits to the right of the decimal point
                scientific is set to TRUE to display scientific notation.
                width indicates the minimum width to be displayed by padding blanks in the
                beginning
                justify is the display of the string to left, right or center.
```

Example:

```
print (format (34.56, nsmall = 10))          # 34.56 is formatted to “34.5600000000”
```

Changing the case

toupper(x)
tolower(x)
x is the vector input

Example:

```
print(toupper("changing to upper"))          # The result is "CHANGING TO UPPER"
print(tolower("CHANGING TO LOWER"))          # The result is "changing to lower"
```

Extracting parts of a string

substring(x,first,last)
x is the character vector input
first is the position of the first character to be extracted.
last is the position of the last character to be extracted

Example:

```
print(substring("Substring", 5, 7))          # The result is "tri"
print(substring("Substring", 5))             # The result is “tring”
```

Parsing

To convert a function into a string, used the **deparse()** function.

deparse(expr, width.cutoff = 60L, backtick = mode(expr) %in% c("call", "expression", "(", "function"), control = c("keepNA", "keepInteger", "niceNames", "showAttributes"), nlines = -1L)
expr any R expression.
width.cutoff integer in [20, 500] determining the cutoff (in bytes) at which line-breaking is tried.
backtick logical indicating whether symbolic names should be enclosed in backticks if they do not follow the standard syntax.
control character vector (or NULL) of deparsing options.
nlines
nlines the maximum number of lines to produce. Negative values indicate no limit.

Example:

```
EQ <- function (x1, x2) 4 * x1 - 6 * x2 + 4   #create a function
deparse(EQ)                                   #result to vector with 2 elements
                                    "function (x1, x2) ";"4 * x1 - 6 * x2 + 4"
deparse(EQ)[2]                                # prints "4 * x1 - 6 * x2 + 4"
```

Split Elements

```
strsplit(x, split)
                x character vector, each element of which is to be split. Other inputs,
                including a factor, will give an error.
                split character vector (or object which can be coerced to such) containing
                regular expression(s) (unless fixed = TRUE) to use for splitting. If empty
                matches occur, in particular if split has length 0, x is split into single
                characters. If split has length greater than 1, it is re-cycled along x.
```

Example:

```
m="Hello/World/!"              #create a string
strsplit(m,"/")                #create a vector with words splitted by '/'
```

Labelled Lists

A labelled list is advantageous if you want the elements of a list to be accessed by a certain name, and not by index. To create a labelled list, refer to the following example:

```
> coordinates = list(x = c(1, 2, 3, 4, 5, 6), y = c(1, 4, 9, 16, 25, 36))
> coordinates$x
[1] 1 2 3 4 5 6
> coordinates$y
[1] 1 4 9 16 25 36
> coordinates
$x
[1] 1 2 3 4 5 6

$y
[1] 1 4 9 16 25 36
```

Learning Experiences

1. Students will have hands-on experience in ***manipulating matrices*** in R.
2. Students will try the different ***built-in functions in string manipulation*** in R.
3. Students will create functions in creating ***augmented coefficient matrix*** in R.
4. Students will attempt to accomplish ***sample exercises for self-learning*** provided below.

Sample Exercises for Self-Learning

1. *Required Competencies*: vector, function, string manipulation commands.
   Create a function that will count the number of variables involved in a given mathematical functions.
2. Required Competencies: vector, function, string manipulation commands.
   Create a function that will count the frequency of the same letters in a given statement. The function must return the matrix with the frequency of letters.