‘From Statistics to Data Mining’ Computer Lab Session n$^\circ$ 4: Linear Algebra (1/2)

Master 1 CPS^2 / COSI Saint-Étienne, France

Fabrice Muhlenbach

Laboratoire Hubert Curien, UMR CNRS 5516
Université Jean Monnet de Saint-Étienne
18 rue du Professeur Benoît Lauras
42000 SAINT-ÉTIENNE, FRANCE
https://perso.univ-st-etienne.fr/muhlfabr/

Outcome

The objective of this lab is to become familiar with R functions for working with linear algebra, especially with basic functions and instructions for handling matrices in R, and for being able to compute matrix multiplication and inversion.

1 Introduction to Computational Linear Algebra

One of the most important applications of linear algebra is in solving systems of linear equations. For example, we have the following equations:

$$(1) \quad l_1 : 3x_1 - 4x_2 = 6$$

$$(2) \quad l_2 : x_1 + 2x_2 = -3$$

Plot with R the two lines l_1 and l_2 corresponding to the two equations ((1) and (2)) and solve them graphically.

Solving :

- $(1) \quad 3x_1 - 4x_2 = 6$
- and $(2) \quad x_1 + 2x_2 = -3$
- $(1) \Leftrightarrow 4x_2 = 3x_1 - 6 \Leftrightarrow x_2 = (3x_1 - 6)/4$
- and $(2) \Leftrightarrow 2x_2 = -x_1 - 3 \Leftrightarrow x2 = (-x_1 - 3)/2$

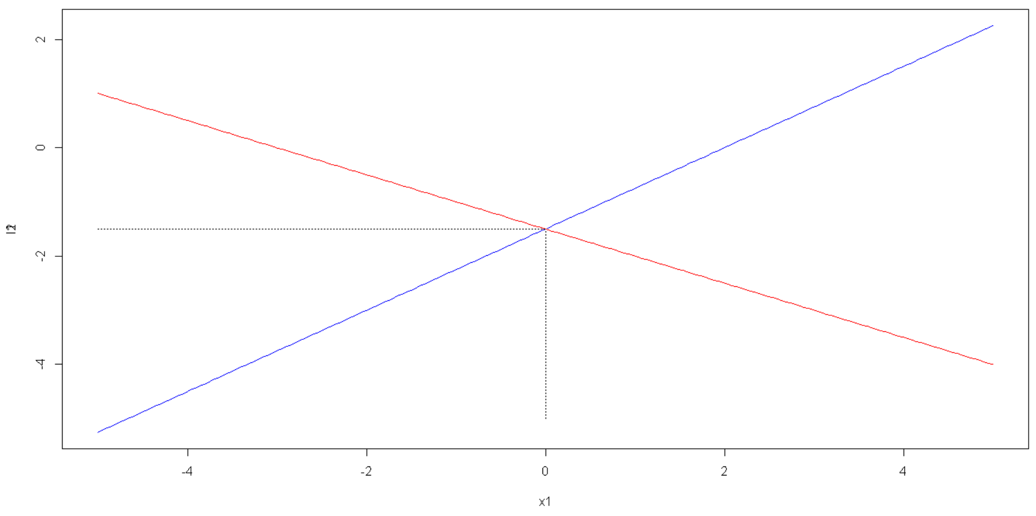
```
x1 <- seq(-5, 5, length=200)
l1 <- (3*x1 - 6) / 4
l2 <- (-x1 - 3) / 2

plot(x1, l1, col="blue", type="l",lwd=1, ylim=range(c(l1,l2)))
par(new=TRUE)
plot(x1, l2, col="red", type="l",lwd=1, ylim=range(c(l1,l2)))
```

These two lines l_1 (in blue) and l_2 (in red) are shown in following figure from which it can be seen that the solution is $x_1 = 0$ and $x_2 = -\frac{3}{2}$. From this figure it follows that there are 3 possible cases of solutions to the system:

1. Exactly one solution (when the lines intersect in one point)
2. Infinitely many solutions (when the lines coincide).
3. No solutions (when the lines are parallel but not identical); in this case, the system of linear equations is said to be inconsistent.

Plot the solution (intersection of the two lines) with two dotted segments.

```
segments(-5,-1.5,0,-1.5, col="black",lty=3)
segments(0,-5,0,-1.5, col="black", lty=3)
```



One of the most important applications of linear algebra is in solving systems of linear equations. For example, we represent the system

$$\left\{ \begin{array}{rcr} 3\,x_1 - 4\,x_2 & = & 6 \\ x_1 + 2\,x_2 & = & -3 \end{array} \right\} \text{ as } Ax = b \text{, where } A = \begin{bmatrix} 3 & -4 \\ 1 & 2 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad b = \begin{bmatrix} 6 \\ -3 \end{bmatrix},$$

and solve it as

$$x = A^{-1}b = \begin{bmatrix} 0.2 & 0.4 \\ -0.1 & 0.3 \end{bmatrix} \begin{bmatrix} 6 \\ -3 \end{bmatrix} = \begin{bmatrix} 0 \\ -1.5 \end{bmatrix}$$

In R, matrices are inverted and linear systems of equations are solved using the solve() (by using a method based on the LU decomposition) or qr.solve() (by using a method based on the QR decomposition) functions. The matrix multiplication can be performed using the operator %*%.

Example:

```
A <- matrix(c(3, 1, -4, 2), ncol=2) ; print(A)
Ainv <- solve(A) ; print(Ainv)
b <- matrix(c(6, -3), ncol=1) ; print(b)
x <- Ainv %*% b ; print(x)
```

The matrix x is the solution:

$$x = \begin{bmatrix} 0 \\ -1.5 \end{bmatrix}$$

$\Leftrightarrow x_1 = 0$ and $x_2 = -3/2$

Problem

With R, find the solution of the following equations:

$$(1) \quad 3x_1 + 2x_2 = 7$$

$$(2) \quad x_1 - 3x_2 = -5$$

Solution

The numerical solution is presented in Marc Sebban's lesson.

```
> A <- matrix(c(3, 1, 2, -3), ncol=2) ; print(A)
     [,1] [,2]
[1,]    3    2
[2,]    1   -3
> Ainv <- solve(A) ; print(Ainv)
           [,1]       [,2]
[1,] 0.27272727  0.1818182
[2,] 0.09090909 -0.2727273
> b <- matrix (c(7, -5) , ncol=1) ; print(b)
     [,1]
[1,]    7
[2,]   -5
> x <- Ainv %*% b ; print (x)
     [,1]
[1,]    1
[2,]    2
```

The matrix x is the solution:

$$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$\Leftrightarrow x_1 = 1$ and $x_2 = 2$
We can check it:

```
> x1 <- 1
> x2 <- 2
> 3*x1 + 2*x2
[1] 7
> x1 - 3*x2
[1] -5
```

2 Vectors and Matrices in R

2.1 Constructing Matrix Objects

Matrices can be constructed using the functions matrix() (for creating a matrix from the given set of values), cbind() or rbind() (for constructing a matrix by combining R objects by columns or rows).

For example, Hilbert matrices are often studied in numerical linear algebra because they are easy to construct but have surprising properties.

```
H3 <- matrix(c(1, 1/2, 1/3, 1/2, 1/3, 1/4, 1/3, 1/4, 1/5), nrow=3)
H3
```

Here $H3$ is the 3×3 Hilbert matrix, where entry (i, j) is $1/(i + j - 1)$. Note that ncol is not required in the command that created it, since the data argument has been assigned a vector consisting of nine elements; it is clear that if there are three rows there must also be three columns. We could also construct this matrix by binding columns together as follows:

```
1/cbind(seq(1, 3), seq(2, 4), seq(3, 5))
```

The seq function will generate a sequence, for example seq(1,3) will produce the results 1 2 3 (similar as 1:3). In this example, rbind() would give the same result, because of symmetry.

Matrices are not necessarily square, for example:

```
> X <- matrix(seq(1, 12), nrow=3)
> X
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

will produce a matrix with 3 rows and $12/3 = 4$ columns.

Another useful function for creating matrices in R is diag(). It extracts or replaces the diagonal of a matrix, or constructs a diagonal matrix. This function is used for creating identity matrices (a square matrix of size $n \times n$ with ones on the main diagonal and zeros elsewhere), for example:

```
> diag(5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
```

Exercise

Use the matrix(), seq() and rep() functions to construct the following 5×5 Hankel matrix:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

2.2 Accessing Matrix elements

Indexing of matrix elements is the same as for data frames: the (i, j) element is located in the i^{th} row and j^{th} column. For example, the $(3, 2)$ element of X is 6. We can access this element using the indices in square brackets:

```
> X[3,2]
[1] 6
```

2.3 Row and Column Names

It is possible for the rows and/or columns of a matrix to have individual names with rownames() and colnames() functions. For example,

```
> colnames(X) <- c("X1", "X2", "Y", "Z")
> rownames(X) <- c("obs1", "obs2", "obs3")
> X
     X1 X2 Y  Z
obs1  1  4 7 10
obs2  2  5 8 11
obs3  3  6 9 12
```

With this name, we can access to a specific cell, a row or a column:

```
> X["obs1","Y"]
[1] 7
> X[,"Z"]
obs1 obs2 obs3
  10   11   12
> X["obs2",]
X1 X2  Y  Z
 2  5  8 11
```

Exercise

Construct the two vectors of heights (in cm) and weights (in kg) for 5 individuals:

```
height <- c(172, 168, 167, 175, 180)
weight <- c(62, 64, 51, 71, 69)
```

Bind these vectors into a matrix, and modify the result to obtain:

```
        height weight
Neil       172     62
Cindy      168     64
Pardeep    167     51
Deepak     175     71
Hao        180     69
```

Pardeep's height is really 162 cm, Hao's height is really 181 cm and his weight is really 68 kg. Correct the matrix accordingly.

2.4 Matrix Properties

The **dimension** of a matrix is its number of rows and its number of columns. For example,

```
> dim(X)
[1] 3 4
```

The **determinant** of a 2×2 matrix $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$ matrix can be calculated as $ad - bc$. For larger square matrices, the calculation becomes more complicated. It can be found in R using the det() function, as in

```
> det(H3)
[1] 0.000462963
```

The **diagonal** elements can be obtained using the diag() function (already seen earlier), as in

```
> diag(X)
[1] 1 5 9
> diag(H3)
[1] 1.0000000 0.3333333 0.2000000
```

We can then compute the **trace** (the sum of the diagonal entries) using a home-made function such as

```
trace <- function(data) sum(diag(data))
```

The trace function uses the sum function that returns the sum of all the values present in its arguments. It can be applied to the matrices X and $H3$:

```
> trace(X)
[1] 15
> trace(H3)
[1] 1.533333
```

The t() function is used to calculate the **matrix transpose** X^T:

```
> t(X)
   obs1 obs2 obs3
X1    1    2    3
X2    4    5    6
Y     7    8    9
Z    10   11   12
```

2.5 Triangular Matrices

The functions lower.tri() and upper.tri() can be used to obtain the lower and upper triangular parts of matrices. The output of the functions is a matrix of logical elements, with TRUE representing the relevant triangular elements. For example,

```
> lower.tri(H3)
      [,1]  [,2]  [,3]
[1,] FALSE FALSE FALSE
[2,]  TRUE FALSE FALSE
[3,]  TRUE  TRUE FALSE
```

2.6 Matrix arithmetic

Multiplication of a matrix by a scalar constant is the same as multiplication of a vector by a constant. For example, using the X matrix from the previous section, we can multiply each element by 2 as in

```
> Y <- 2 * X
> Y
      X1 X2  Y  Z
obs1   2  8 14 20
obs2   4 10 16 22
obs3   6 12 18 24
```

Elementwise addition of matrices also proceeds as for vectors. For example,

```
> Y + X
      X1 X2  Y  Z
obs1   3 12 21 30
obs2   6 15 24 33
obs3   9 18 27 36
```

When adding matrices, always ensure that the dimensions match properly. If they do not match correctly, an error message will appear.

The command `X * Y` performs elementwise multiplication. Note that this differs from the usual form of matrix multiplication that we will discuss below. For example,

```
> X * Y
      X1 X2   Y   Z
obs1   2 32  98 200
obs2   8 50 128 242
obs3  18 72 162 288
```

Again, in order for this kind of multiplication to work, the dimensions of the matrices must match.

3 Matrix Multiplication and Inversion

If A and B are matrices, then the matrix product AB is the matrix representing the composition of the two operations: first apply B, then apply A to the result. For matrix multiplication to be a properly defined operation, the matrices to be multiplied must *conform*. That is, the number of columns of the first matrix must match the number of rows of the second matrix. The resulting matrix AB will have its row dimension taken from A and its column dimension taken from B. In R, this form of matrix multiplication can be performed using the operator `%*%`, for example

```
> t(Y) %*% X
    X1  X2   Y   Z
X1  28  64 100 136
X2  64 154 244 334
Y  100 244 388 532
Z  136 334 532 730
```

From the previous section we saw that t(Y) has three columns and X has three rows, so we can perform the multiplication $Y^T X$. The result is a 2 × 2 matrix, since t(Y) has two rows and X has two columns. If we failed to transpose Y, we would obtain an error.

The crossprod() function is a somewhat more efficient way to calculate $Y^T X$:

```
> crossprod(Y,X)
     X1  X2   Y   Z
X1   28  64 100 136
X2   64 154 244 334
Y   100 244 388 532
Z   136 334 532 730
```

3.1 Matrix Inversion

The inverse of a square $n \times n$ matrix A, denoted by A^{-1}, is the solution to the matrix equation $AA^{-1} = I$, where I is the $n \times n$ identity matrix. We can view this as n separate systems of linear equations in n unknowns, whose solutions are the columns of A^{-1}.

For example, with

$$A = \begin{bmatrix} 3 & -4 \\ 1 & 2 \end{bmatrix},$$

the matrix equation

$$\begin{bmatrix} 3 & -4 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

is equivalent to the two equations:

$$\begin{bmatrix} 3 & -4 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} 3 & -4 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} b_{12} \\ b_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The usual computational approach to finding A^{-1} involves solving these two equations.

However, this is not always a good idea! Often the reason we are trying to find A^{-1} is so that we can solve a system $Ax = b$ with the solution $x = A^{-1}b$. It doesn't make sense from a computational point of view to solve n systems of linear equations in order to obtain a result which will be used as the solution to one system. If we know how to solve systems, we should use that knowledge to solve $Ax = b$ directly. Furthermore, using A^{-1} may give a worse approximation to the final result than the direct approach, because there are so many more operations involved, giving opportunities for much more rounding error to creep into our results.

In R, matrices are inverted and linear systems of equations are solved using the solve() or qr.solve() functions. solve() uses a method based on the *LU* decomposition; qr.solve() is based on the *QR* decomposition.

As an example, we compute the inverse of the 3 × 3 Hilbert matrix introduced previously:

```
> H3inv <- solve(H3)
> H3inv
      [,1] [,2] [,3]
[1,]     9  -36   30
[2,]   -36  192 -180
[3,]    30 -180  180
```

To verify that this is the inverse of $H3$, we can check that the product of $H3inv$ and $H3$ is the 3×3 identity:

```
> H3inv %*% H3
              [,1]          [,2]          [,3]
[1,]  1.000000e+00 8.881784e-16  6.882515e-16
[2,] -3.774758e-15 1.000000e+00 -3.420875e-15
[3,]  6.144391e-15 0.000000e+00  1.000000e+00
```

3.2 Solving linear systems

The function solve(A, b) gives the solution to systems of equations of the form $Ax = b$. For example, let us find x such that $H3x = b$ where $H3$ is the 3×3 Hilbert matrix and $b = [123]^T$.

```
> b <- c(1, 2, 3)
> x <- solve(H3, b)
> x
[1]    27 -192   210
```

In other words, the solution vector is $x = [27, -192, 210]^T$.

Exercise

Let $[x_1, x_2, x_3, x_4, x_5, x_6]^T = [10, 11, 12, 13, 14, 15]^T$.
Find the coefficients of the quintic polynomial $f(x)$ for which

$$[f(x_1), f(x_2), f(x_3), f(x_4), f(x_5), fx_6)]^T = [25, 16, 26, 19, 21, 20]^T.$$

The quintic polynomial $f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5$ can be viewed as the matrix product of the row vector $[1, x, x^2, x^3, x^4, x^5]$ with the column vector $[a_0, a_1, a_2, a_3, a_4, a_5]^T$.
Work out the matrix version of this to give $[f(x_1), f(x_2), f(x_3), f(x_4), f(x_5), f(x_6)]^T$.

4 Recommended Readings

- Braun and Murdoch (2007), "A First Course in Statistical Programming with R,"
  Chapter 6 "Computational linear algebra."
- Ciarlet et al. (1989), "Introduction to numerical linear algebra and optimisation"
- Lay (2012), "Linear Algebra and Its Applications,"
  Chapter 2 "Matrix Algebra,"
  Chapter 3 "Determinants,"
  Chapter 5 "Eigenvalues and Eigenvectors" and
  Chapter 7 "Symmetric Matrices and Quadratic Forms."
- Venables et al. (2013), "An Introduction to R,"
  Chapter 5 "Arrays and matrices."
- Teetor (2011), "R Cookbook",
  Chapter 13 "Beyond Basic Numerics and Statistics."

References

Braun, W. J. and D. J. Murdoch (2007). *A First Course in Statistical Programming with R*. Cambridge University Press.

Ciarlet, P. G., B. Miara, and J.-M. Thomas (1989). *Introduction to numerical linear algebra and optimisation*. Cambridge texts in applied mathematics. Cambridge University Press.

Lay, D. C. (2012). *Linear Algebra and Its Applications* (4^{th} ed.). Addison-Wesley.

Teetor, P. (2011). *R Cookbook*. O'Reilly.

Venables, W. N., D. M. Smith, and the R Core Team (2013). An introduction to R –notes on R: A programming environment for data analysis and graphics.
URL `http://cran.r-project.org/doc/manuals/R-intro.html`.