Introduction to linear algebra with R

August 2011
Version 4

Compiled Friday 5th August, 2011, 10:18
from: C:/Bendix/undervis/APC/00/LinearAlgebra/linalg-notes-BxC.tex

Søren Højsgaard — Department of Genetics and Biotechnology,
Aarhus University
DK–8830 Tjele, Denmark

a few additions by:
Bendix Carstensen — Steno Diabetes Center, Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
bxc@steno.dk
www.pubhealth.ku.dk/~bxc

Contents

Chapter 1

Matrix algebra

1.1 Introduction

These notes have two aims: 1) Introducing linear algebra (vectors and matrices) and 2) showing how to work with these concepts in R. They were written in an attempt to give a specific group of students a "feeling" for what matrices, vectors etc. are all about. Hence the notes/slides are are not suitable for a course in linear algebra.

1.2 Vectors

1.2.1 Vectors

A column vector is a list of numbers stacked on top of each other, e.g.

$$a = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$

A row vector is a list of numbers written one after the other, e.g.

$$b = (2, 1, 3)$$

In both cases, the list is ordered, i.e.

$$(2, 1, 3) \neq (1, 2, 3).$$

We make the following convention:

- In what follows all vectors are column vectors unless otherwise stated.
- However, writing column vectors takes up more space than row vectors. Therefore we shall frequently write vectors as row vectors, but with the understanding that it really is a column vector.

A general n–vector has the form

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

where the a_is are numbers, and this vector shall be written $a = (a_1, \ldots, a_n)$.

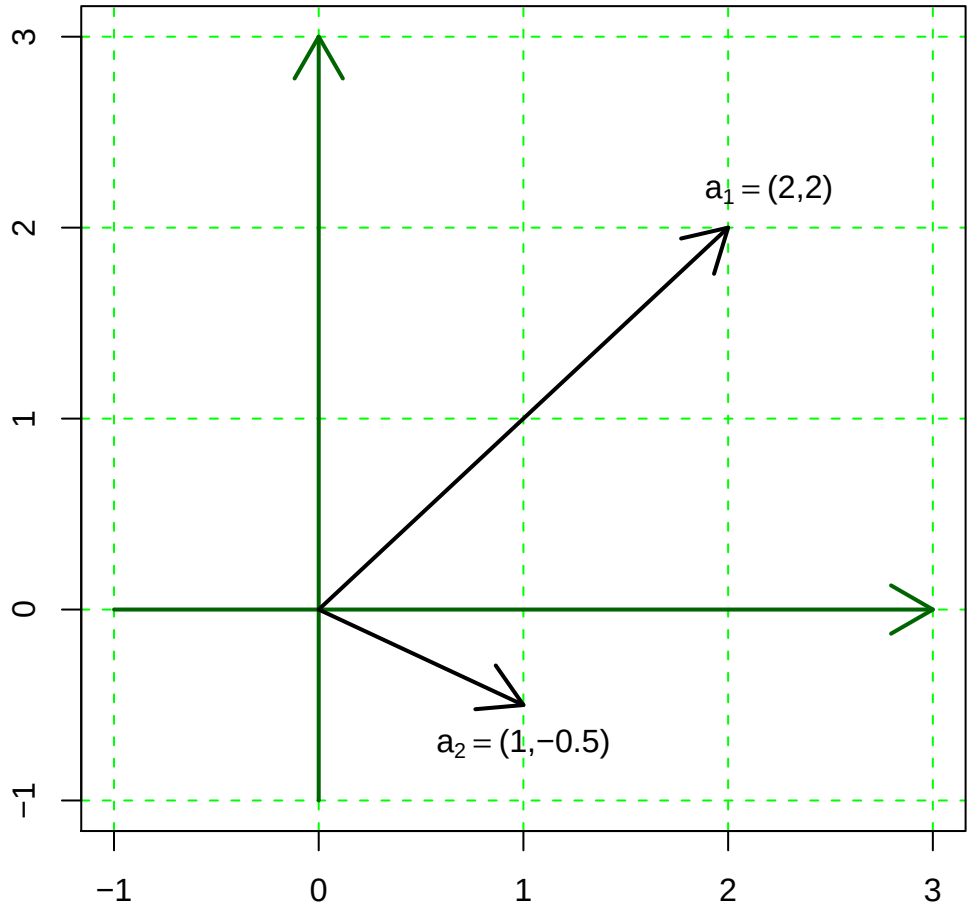

Figure 1.1: Two 2-vectors

A graphical representation of 2–vectors is shown Figure 1.1. Note that row and column vectors are drawn the same way.

```
> a <- c(1,3,2)
> a

[1] 1 3 2
```

The vector `a` is in R printed "in row format" but can really be regarded as a column vector, cfr. the convention above.

1.2.2 Transpose of vectors

Transposing a vector means turning a column (row) vector into a row (column) vector. The transpose is denoted by "$^\top$".

Example 1.2.1

$$\begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}^\top = [1,3,2] \quad \text{og} \quad [1,3,2]^\top = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

■

Hence transposing twice takes us back to where we started:

$$a = (a^\top)^\top$$

To illustrate this, we can transpose `a` to obtain a 1×3 matrix (which is the same as a 3–row vector):

```
> t(a)

     [,1] [,2] [,3]
[1,]    1    3    2
```

1.2.3 Multiplying a vector by a number

If a is a vector and α is a number then αa is the vector

$$\alpha a = \begin{bmatrix} \alpha a_1 \\ \alpha a_2 \\ \vdots \\ \alpha a_n \end{bmatrix}$$

See Figure 1.2.

Example 1.2.2

$$7 \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 7 \\ 21 \\ 14 \end{bmatrix}$$

■

Multiplication by a number:

```
> 7*a

[1]  7 21 14
```

1.2.4 Sum of vectors

Let a and b be n–vectors. The sum $a + b$ is the n–vector

$$a + b = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_n + b_n \end{bmatrix} = b + a$$

See Figure 1.3 and 1.4. Only vectors of the same dimension can be added.

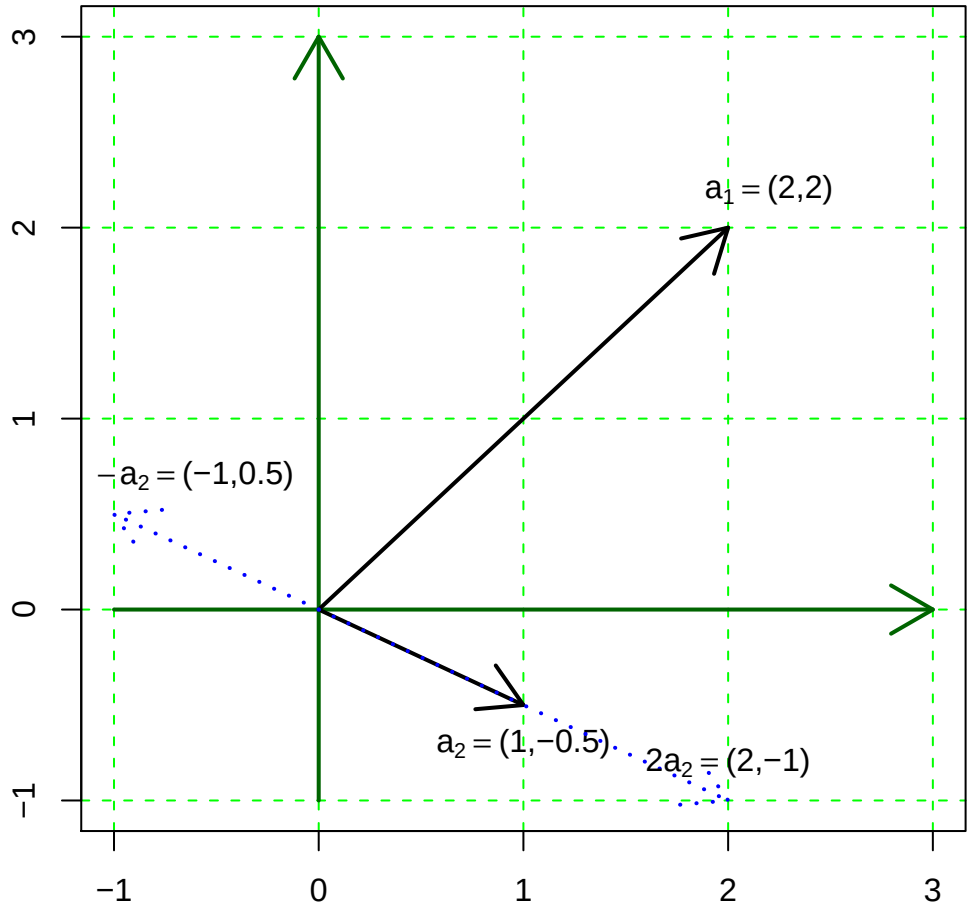

Figure 1.2: Multiplication of a vector by a number

Example 1.2.3

$$\begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} + \begin{bmatrix} 2 \\ 8 \\ 9 \end{bmatrix} = \begin{bmatrix} 1+2 \\ 3+8 \\ 2+9 \end{bmatrix} = \begin{bmatrix} 3 \\ 11 \\ 11 \end{bmatrix}$$

■

Addition of vectors:

```
> a <- c(1,3,2)
> b <- c(2,8,9)
> a+b

[1]  3 11 11
```

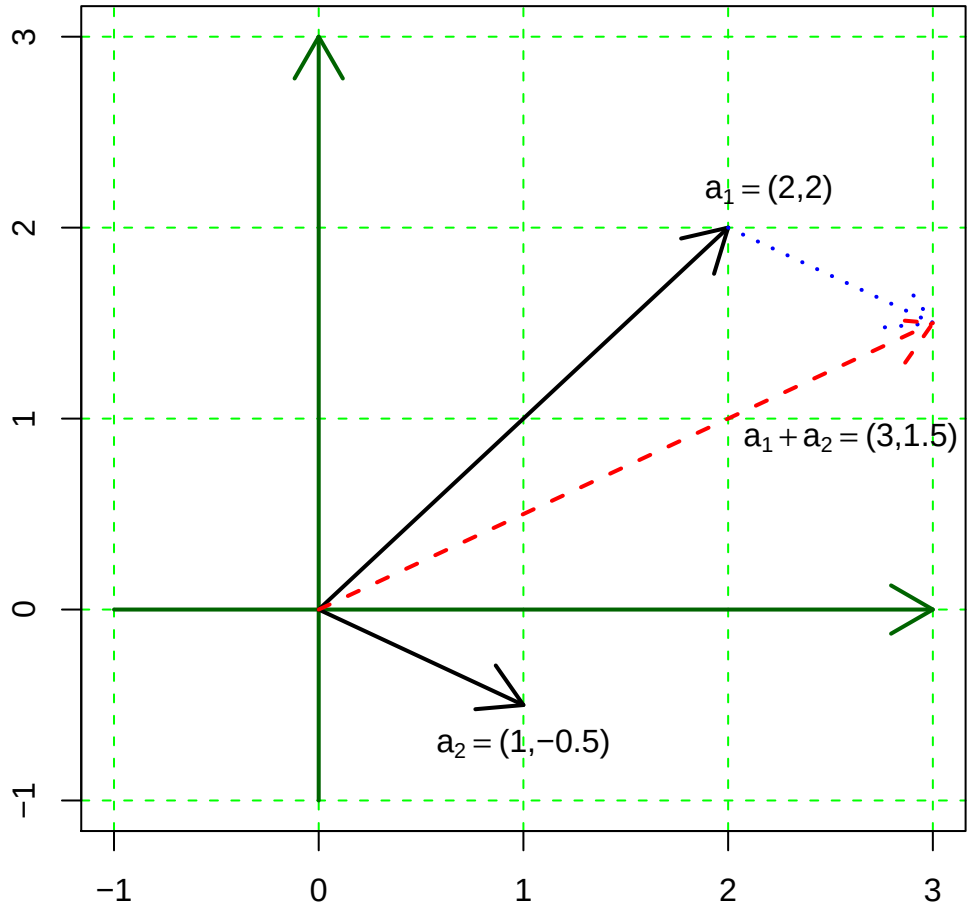

Figure 1.3: Addition of vectors

1.2.5 Inner product of vectors

Let $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$. The inner product of a and b is

$$a \cdot b = a_1 b_1 + \cdots + a_n b_n$$

Note, that the inner product is a number – not a vector:

```
> sum(a*b)
[1] 44
```

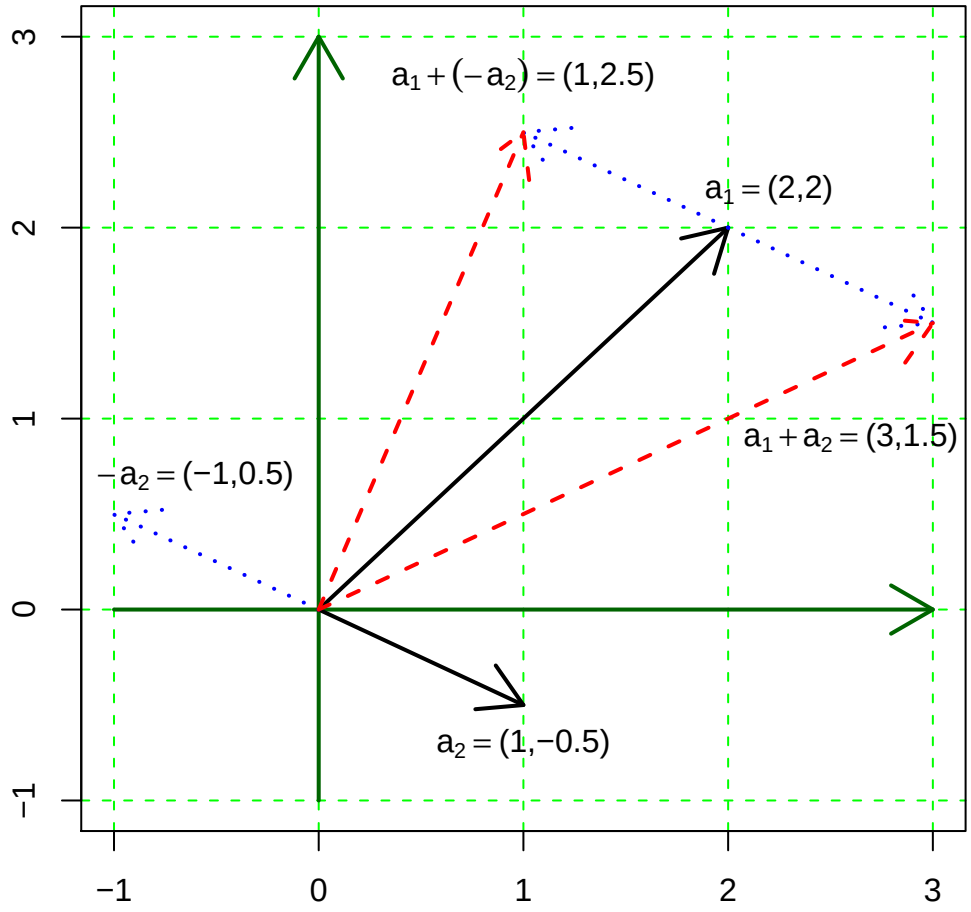

Figure 1.4: Addition of vectors and multiplication by a number

1.2.6 The length (norm) of a vector

The length (or norm) of a vector a is

$$||a|| = \sqrt{a \cdot a} = \sqrt{\sum_{i=1}^{n} a_i^2}$$

Norm (length) of vector:

```
> sqrt(sum(a*a))
[1] 3.741657
```

1.2.7 The 0–vector and 1–vector

The 0-vector (1–vector) is a vector with 0 (1) on all entries. The 0–vector (1–vector) is frequently written simply as 0 (1) or as 0_n (1_n) to emphasize that its length n.

0–vector and 1–vector

```
> rep(0,5)

[1] 0 0 0 0 0

> rep(1,5)

[1] 1 1 1 1 1
```

1.2.8 Orthogonal (perpendicular) vectors

Two vectors v_1 and v_2 are orthogonal if their inner product is zero, written

$$v_1 \perp v_2 \Leftrightarrow v_1 \cdot v_2 = 0$$

Note that any vector is orthogonal to the 0–vector. Orthogonal vectors:

```
> v1 <- c(1,1)
> v2 <- c(-1,1)
> sum(v1*v2)

[1] 0
```

1.3 Matrices

1.3.1 Matrices

An $r \times c$ matrix A (reads "an r times c matrix") is a table with r *r*ows og c *c*olumns

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1c} \\ a_{21} & a_{22} & \dots & a_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1} & a_{r2} & \dots & a_{rc} \end{bmatrix}$$

Note that one can regard A as consisting of c columns vectors put after each other:

$$A = [a_1 : a_2 : \cdots : a_c]$$

Likewise one can regard A as consisting of r row vectors stacked on to of each other.

Create a matrix:

```
> A <- matrix(c(1,3,2,2,8,9),ncol=3)
> A

     [,1] [,2] [,3]
[1,]    1    2    8
[2,]    3    2    9
```

Note that the numbers $1, 3, 2, 2, 8, 9$ are read into the matrix column–by–column. To get the numbers read in row–by–row do>

```
> A2 <- matrix(c(1,3,2,2,8,9),ncol=3,byrow=T)
> A2

     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    8    9
```

1.3.2 Multiplying a matrix with a number

For a number α and a matrix A, the product αA is the matrix obtained by multiplying each element in A by α.

Example 1.3.1

$$7\begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 2 & 9 \end{bmatrix} = \begin{bmatrix} 7 & 14 \\ 21 & 56 \\ 14 & 63 \end{bmatrix}$$

■

Multiplication of matrix by a number:

```
> 7*A

     [,1] [,2] [,3]
[1,]    7   14   56
[2,]   21   14   63
```

1.3.3 Transpose of matrices

A matrix is transposed by interchanging rows and columns and is denoted by "$\top$".

Example 1.3.2

$$\begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 2 & 9 \end{bmatrix}^{\top} = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 8 & 9 \end{bmatrix}$$

■

Note that if A is an $r \times c$ matrix then $A^{\top}$ is a $c \times r$ matrix.

Transpose of matrix

```
> t(A)

     [,1] [,2]
[1,]    1    3
[2,]    2    2
[3,]    8    9
```

1.3.4 Sum of matrices

Let A and B be $r \times c$ matrices. The sum $A + B$ is the $r \times c$ matrix obtained by adding A and B element wise.

Only matrices with the same dimensions can be added.

Example 1.3.3

$$\begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 2 & 9 \end{bmatrix} + \begin{bmatrix} 5 & 4 \\ 8 & 2 \\ 3 & 7 \end{bmatrix} = \begin{bmatrix} 6 & 6 \\ 11 & 10 \\ 5 & 16 \end{bmatrix}$$

■

Addition of matrices

```
> B <- matrix(c(5,8,3,4,2,7),ncol=3,byrow=T)
> A+B

     [,1] [,2] [,3]
[1,]    6   10   11
[2,]    7    4   16
```

1.3.5 Multiplication of a matrix and a vector

Let A be an $r \times c$ matrix and let b be a c-dimensional column vector. The product Ab is the $r \times 1$ matrix

$$Ab = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1c} \\ a_{21} & a_{22} & \dots & a_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1} & a_{r2} & \dots & a_{rc} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{bmatrix} = \begin{bmatrix} a_{11}b_1 + a_{12}b_2 + \cdots + a_{1c}b_c \\ a_{21}b_1 + a_{22}b_2 + \cdots + a_{2c}b_c \\ \vdots \\ a_{r1}b_1 + a_{r2}b_2 + \cdots + a_{rc}b_c \end{bmatrix}$$

Example 1.3.4

$$\begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \end{bmatrix} = \begin{bmatrix} 1 \cdot 5 + 2 \cdot 8 \\ 3 \cdot 5 + 8 \cdot 8 \\ 2 \cdot 5 + 9 \cdot 8 \end{bmatrix} = \begin{bmatrix} 21 \\ 79 \\ 82 \end{bmatrix}$$

■

Multiplication of a matrix and a vector

```
> A%*%a

     [,1]
[1,]   23
[2,]   27
```

Note the difference to:

```
> A*a

     [,1] [,2] [,3]
[1,]    1    4   24
[2,]    9    2   18
```

Figure out yourself what goes on!

1.3.6 Multiplication of matrices

Let A be an $r \times c$ matrix and B a $c \times t$ matrix, i.e. $B = [b_1 : b_2 : \cdots : b_t]$. The product AB is the $r \times t$ matrix given by:

$$AB = A[b_1 : b_2 : \cdots : b_t] = [Ab_1 : Ab_2 : \cdots : Ab_t]$$

Example 1.3.5

$$\begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} 5 & 4 \\ 8 & 2 \end{bmatrix} = \left[\begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \end{bmatrix} \; : \; \begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix} \right]$$

$$= \begin{bmatrix} 1 \cdot 5 + 2 \cdot 8 & 1 \cdot 4 + 2 \cdot 2 \\ 3 \cdot 5 + 8 \cdot 8 & 3 \cdot 4 + 8 \cdot 2 \\ 2 \cdot 5 + 9 \cdot 8 & 2 \cdot 4 + 9 \cdot 2 \end{bmatrix} = \begin{bmatrix} 21 & 8 \\ 79 & 28 \\ 82 & 26 \end{bmatrix}$$

■

Note that the product AB can only be formed if the number of rows in B and the number of columns in A are the same. In that case, A and B are said to be conform.

In general AB and BA are not identical.

A mnemonic for matrix multiplication is :

$$\begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 2 & 9 \end{bmatrix}\begin{bmatrix} 5 & 4 \\ 8 & 2 \end{bmatrix} = \begin{array}{cc|cc} & & 5 & 4 \\ & & 8 & 2 \\ \hline 1 & 2 & 1\cdot 5+2\cdot 8 & 1\cdot 4+2\cdot 2 \\ 3 & 8 & 3\cdot 5+8\cdot 8 & 3\cdot 4+8\cdot 2 \\ 2 & 9 & 2\cdot 5+9\cdot 8 & 2\cdot 4+9\cdot 2 \end{array} = \begin{bmatrix} 21 & 8 \\ 79 & 28 \\ 82 & 26 \end{bmatrix}$$

Matrix multiplication:

```
> A <- matrix(c(1,3,2,2,8,9),ncol=2)
> B <- matrix(c(5,8,4,2), ncol=2)
> A%*%B

     [,1] [,2]
[1,]   21    8
[2,]   79   28
[3,]   82   26
```

1.3.7 Vectors as matrices

One can regard a column vector of length r as an $r \times 1$ matrix and a row vector of length c as a $1 \times c$ matrix.

1.3.8 Some special matrices

- An $n \times n$ matrix is a *square matrix*
- A matrix A is *symmetric* if $A = A^\top$.
- A matrix with 0 on all entries is the 0*–matrix* and is often written simply as 0.
- A matrix consisting of 1s in all entries is often written J.
- A square matrix with 0 on all off–diagonal entries and elements $d_1, d_2, \ldots, d_n$ on the diagonal a *diagonal matrix* and is often written $\text{diag}\{d_1, d_2, \ldots, d_n\}$
- A diagonal matrix with 1s on the diagonal is called the *identity matrix* and is denoted I. The identity matrix satisfies that $IA = AI = A$. Likewise, if x is a vector then $Ix = x$.

0-matrix and 1-matrix

```
> matrix(0,nrow=2,ncol=3)

     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0

> matrix(1,nrow=2,ncol=3)

     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
```

Diagonal matrix and identity matrix

```
> diag(c(1,2,3))

     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3

> diag(1,3)

     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

Note what happens when `diag` is applied to a matrix:

```
> diag(diag(c(1,2,3)))

[1] 1 2 3

> diag(A)

[1] 1 8
```

1.3.9 Inverse of matrices

In general, the inverse of an $n \times n$ matrix A is the matrix B (which is also $n \times n$) which when multiplied with A gives the identity matrix I. That is,

$$AB = BA = I.$$

One says that B is A's inverse and writes $B = A^{-1}$. Likewise, A is Bs inverse.

Example 1.3.6

Let

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \quad B = \begin{bmatrix} -2 & 1.5 \\ 1 & -0.5 \end{bmatrix}$$

Now $AB = BA = I$ so $B = A^{-1}$.

■

Example 1.3.7

If A is a 1×1 matrix, i.e. a number, for example $A = 4$, then $A^{-1} = 1/4$.

■

Some facts about inverse matrices are:

- Only square matrices can have an inverse, but not all square matrices have an inverse.
- When the inverse exists, it is unique.
- Finding the inverse of a large matrix A is numerically complicated (but computers do it for us).

Finding the inverse of a matrix in R is done using the `solve()` function:

```
> A <- matrix(c(1,3,2,4),ncol=2,byrow=T)
> A

     [,1] [,2]
[1,]    1    3
[2,]    2    4

> #M2 <- matrix(c(-2,1.5,1,-0.5),ncol=2,byrow=T)
> B <- solve(A)
> B

     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5

> A%*%B

     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

1.3.10 Solving systems of linear equations

Example 1.3.8

Matrices are closely related to systems of linear equations. Consider the two equations

$$\begin{aligned} x_1 + 3x_2 &= 7 \\ 2x_1 + 4x_2 &= 10 \end{aligned}$$

The system can be written in matrix form

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7 \\ 10 \end{bmatrix} \quad \text{i.e. } Ax = b$$

Since $A^{-1}A = I$ and since $Ix = x$ we have

$$x = A^{-1}b = \begin{bmatrix} -2 & 1.5 \\ 1 & -0.5 \end{bmatrix} \begin{bmatrix} 7 \\ 10 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

A geometrical approach to solving these equations is as follows: Isolate x_2 in the equations:

$$x_2 = \frac{7}{3} - \frac{1}{3}x_1 \quad x_2 = \frac{1}{0}4 - \frac{2}{4}x_1$$

These two lines are shown in Figure 1.5 from which it can be seen that the solution is $x_1 = 1, x_2 = 2$.

From the 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. No solutions – when the lines are parallel but not identical
3. Infinitely many solutions – when the lines coincide.

■

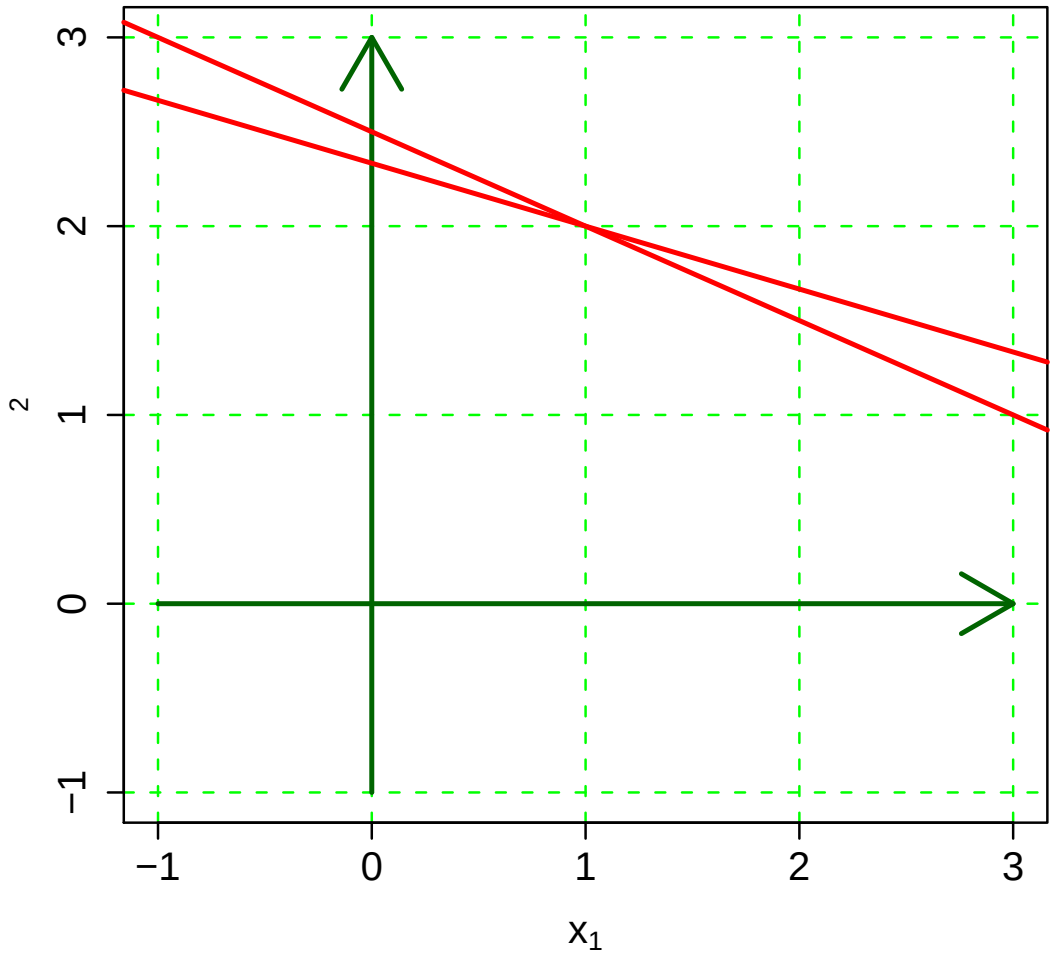

Figure 1.5: Solving two equations with two unknowns.

Solving systems of linear equations: If $Mx = z$ where M is a matrix and x and z are vectors the solution is $x = M^{-1}z$:

```
> A <- matrix(c(1,2,3,4),ncol=2)
> b <- c(7,10)
> x <- solve(A)%*%b
> x
     [,1]
[1,]    1
[2,]    2
```

Actually, the reason for the name "`solve`" for the matrix inverter is that it solves (several) systems of linear equations; the second argument of `solve` just defaults to the identity matrix. Hence the above example can be fixed in one go by:

```
> solve(A,b)
[1] 1 2
```

1.3.11 Some additional rules for matrix operations

For matrices A, B and C whose dimension match appropriately: the following rules apply

$$(A+B)^\top = A^\top + B^\top$$

$$(AB)^\top = B^\top A^\top$$

$$A(B+C) = AB + AC$$

$$AB = AC \not\Rightarrow B = C$$

In general $AB \neq BA$

$$AI = IA = A$$

If α is a number then $\alpha AB = A(\alpha B)$

1.3.12 Details on inverse matrices

1.3.12.1 Inverse of a 2×2 matrix

It is easy find the inverse for a 2×2 matrix. When

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

then the inverse is

$$A^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

under the assumption that $ab - bc \neq 0$. The number $ab - bc$ is called the determinant of A, sometimes written $|A|$ or $\det(A)$. A matrix A has an inverse if and only if $|A| \neq 0$.

If $|A| = 0$, then A has no inverse, which happens if and only if the columns of A are linearly dependent.

1.3.12.2 Inverse of diagonal matrices

Finding the inverse of a diagonal matrix is easy: Let

$$A = \mathrm{diag}(a_1, a_2, \ldots, a_n)$$

where all $a_i \neq 0$. Then the inverse is

$$A^{-1} = \mathrm{diag}(\frac{1}{a_1}, \frac{1}{a_2}, \ldots, \frac{1}{a_n})$$

If one $a_i = 0$ then A^{-1} does not exist.

1.3.12.3 Generalized inverse

Not all square matrices have an inverse — only those of full rank. However all matrices (not only square ones) have an infinite number of generalized inverses. A generalized inverse (G-inverse) of a matrix A is a matrix A^- satisfying that

$$AA^-A = A.$$

Note that if A is $r \times c$ then necessarily A^- must be $c \times r$.

The generalized inverse can be found by the function `ginv` from the `MASS` package:

```
> library( MASS )
> ( A <- rbind(c(1,3,2),c(2,8,9)) )

     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    8    9

> ginv(A)

           [,1]       [,2]
[1,]  0.4066667 -0.1066667
[2,]  0.6333333 -0.1333333
[3,] -0.6533333  0.2533333

> A %*% ginv(A)

              [,1]          [,2]
[1,]  1.000000e+00 2.220446e-16
[2,] -8.881784e-16 1.000000e+00

> ginv(A) %*% A

           [,1]       [,2]        [,3]
[1,]  0.1933333 0.36666667 -0.14666667
[2,]  0.3666667 0.83333333  0.06666667
[3,] -0.1466667 0.06666667  0.97333333

> A %*% ginv(A) %*% A

     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    8    9
```

Note that since A is 2×3, A^- is 3×2, so the matrix AA^- is the smaller of the two square matrices AA^- and A^-A. Because A is of full rank (and only then) $AA^- = I$. This is the case for any G-inverse of a full rank matrix.

For many practical problems it suffices to find a generalized inverse. We shall return to this in the discussion of reparametrization of models.

1.3.12.4 Inverting an $n \times n$ matrix

In the following we will illustrate one frequently applied method for matrix inversion. The method is called Gauss–Seidels method and many computer programs, including `solve()` use variants of the method for finding the inverse of an $n \times n$ matrix.

Consider the matrix A:

```
> A <- matrix(c(2,2,3,3,5,9,5,6,7),ncol=3)
> A

     [,1] [,2] [,3]
[1,]    2    3    5
[2,]    2    5    6
[3,]    3    9    7
```

We want to find the matrix $B = A^{-1}$. To start, we append to A the identity matrix and call the result AB:

```
> AB <- cbind(A,diag(c(1,1,1)))
> AB

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2    3    5    1    0    0
[2,]    2    5    6    0    1    0
[3,]    3    9    7    0    0    1
```

On a matrix we allow ourselves to do the following three operations (sometimes called

elementary operations) as often as we want:

1. Multiply a row by a (non–zero) constant.
2. Multiply a row by a (non–zero) constant and add the result to another row.
3. Interchange two rows.

The aim is to perform such operations on `AB` in a way such that one ends up with a 3×6 matrix which has the identity matrix in the three leftmost columns. The three rightmost columns will then contain $B = A^{-1}$.

Recall that writing e.g. `AB[1,]` extracts the entire first row of AB.

- First, we make sure that `AB[1,1]=1`. Then we subtract a constant times the first row from the second to obtain that `AB[2,1]=0`, and similarly for the third row:

```
> AB[1,] <- AB[1,]/AB[1,1]
> AB[2,] <- AB[2,]-2*AB[1,]
> AB[3,] <- AB[3,]-3*AB[1,]
> AB
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1  1.5  2.5  0.5    0    0
[2,]    0  2.0  1.0 -1.0    1    0
[3,]    0  4.5 -0.5 -1.5    0    1
```

- Next we ensure that `AB[2,2]=1`. Afterwards we subtract a constant times the second row from the third to obtain that `AB[3,2]=0`:

```
> AB[2,] <- AB[2,]/AB[2,2]
> AB[3,] <- AB[3,]-4.5*AB[2,]
```

- Now we rescale the third row such that `AB[3,3]=1`:

```
> AB[3,] <- AB[3,]/AB[3,3]
> AB
     [,1] [,2] [,3]       [,4]      [,5]       [,6]
[1,]    1  1.5  2.5  0.5000000 0.0000000  0.0000000
[2,]    0  1.0  0.5 -0.5000000 0.5000000  0.0000000
[3,]    0  0.0  1.0 -0.2727273 0.8181818 -0.3636364
```

Then `AB` has zeros below the main diagonal.

- We then work our way up to obtain that `AB` has zeros above the main diagonal:

```
> AB[2,] <- AB[2,] - 0.5*AB[3,]
> AB[1,] <- AB[1,] - 2.5*AB[3,]
> AB
     [,1] [,2] [,3]       [,4]        [,5]       [,6]
[1,]    1  1.5    0  1.1818182 -2.04545455  0.9090909
[2,]    0  1.0    0 -0.3636364  0.09090909  0.1818182
[3,]    0  0.0    1 -0.2727273  0.81818182 -0.3636364

> AB[1,] <- AB[1,] - 1.5*AB[2,]
> AB
     [,1] [,2] [,3]       [,4]        [,5]       [,6]
[1,]    1    0    0  1.7272727 -2.18181818  0.6363636
[2,]    0    1    0 -0.3636364  0.09090909  0.1818182
[3,]    0    0    1 -0.2727273  0.81818182 -0.3636364
```

Now we extract the three rightmost columns of `AB` into the matrix `B`. We claim that `B` is the inverse of `A`, and this can be verified by a simple matrix multiplication

```
> B <- AB[,4:6]
> A%*% B

              [,1]          [,2]          [,3]
[1,]  1.000000e+00 3.330669e-16 1.110223e-16
[2,] -4.440892e-16 1.000000e+00 2.220446e-16
[3,] -2.220446e-16 9.992007e-16 1.000000e+00
```

So, apart from rounding errors, the product is the identity matrix, and hence $B = A^{-1}$. This example illustrates that numerical precision and rounding errors is an important issue when making computer programs.

Chapter 2

Linear models

2.1 Least squares

Consider the table of pairs (x_i, y_i) below.

x	1.00	2.00	3.00	4.00	5.00
y	3.70	4.20	4.90	5.70	6.00

A plot of y_i against x_i is shown in Figure 2.1.

The plot in Figure 2.1 suggests an approximately linear relationship between y and x, i.e.

$$y_i = \beta_0 + \beta_1 x_i \text{ for } i = 1, \ldots, 5$$

Writing this in matrix form gives

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_5 \end{bmatrix} \approx \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_5 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \mathbf{X}\beta$$

The first question is: Can we find a vector β such that $y = X\beta$? The answer is clearly no, because that would require the points to lie exactly on a straight line.

A more modest question is: Can we find a vector $\hat{\beta}$ such that $X\hat{\beta}$ is in a sense "as close to y as possible". The answer is yes. The task is to find $\hat{\beta}$ such that the length of the vector

$$e = y - X\beta$$

is as small as possible. The solution is

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

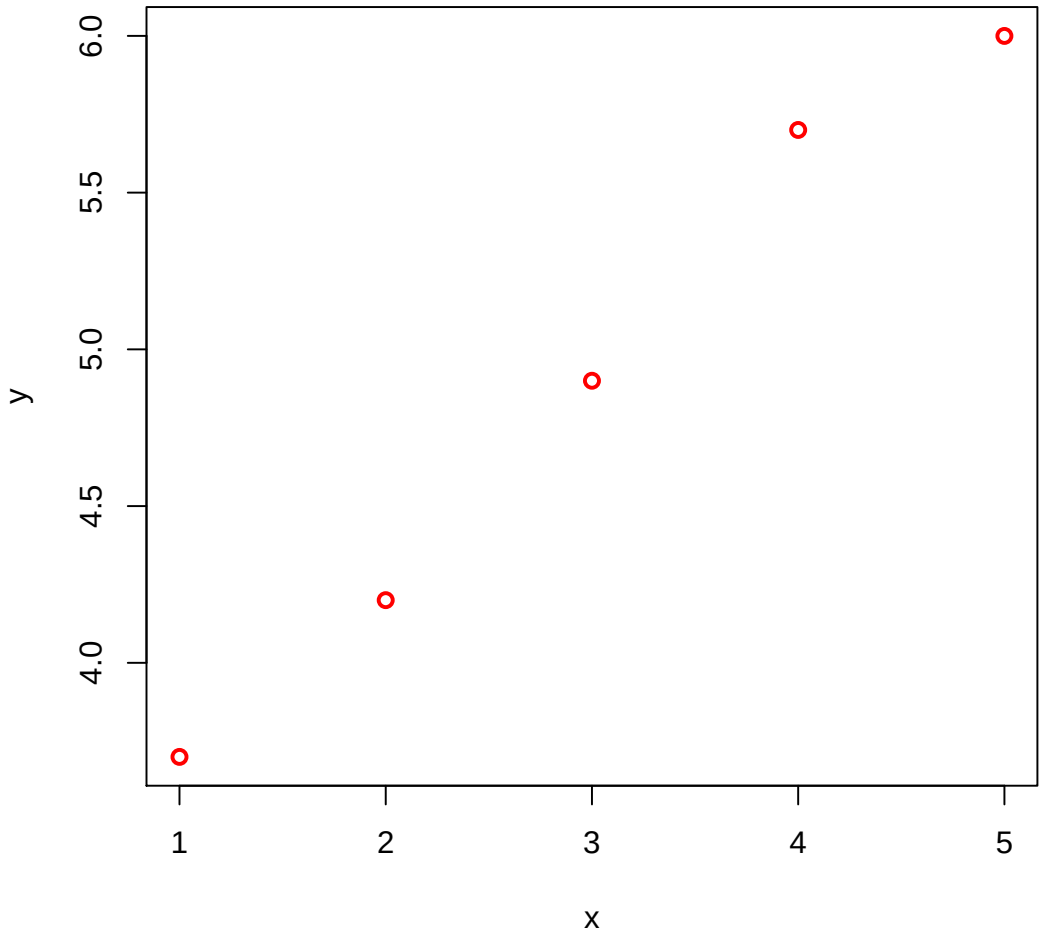

Figure 2.1: Regression

```
> y
[1] 3.7 4.2 4.9 5.7 6.0
> X
      x
[1,] 1 1
[2,] 1 2
[3,] 1 3
[4,] 1 4
[5,] 1 5
> beta.hat <- solve(t(X)%*% X)%*% t(X)%*%y
> beta.hat
  [,1]
  3.07
x 0.61
```

2.1.1 A neat little exercise — from a bird's perspective

Exercise 2.1.1

On a sunny day, two tables are standing in an English country garden. On each table birds of unknown species are sitting having the time of their lives.

A bird from the first table says to those on the second table: "Hi – if one of you come to our table then there will be the same number of us on each table". "Yeah, right", says a bird from the second table, "but if one of you comes to our table, then we will be twice as many on our table as on yours".

How many birds are on each table? More specifically,

- Write down two equations with two unknowns.
- Solve these equations using the methods you have learned from linear algebra.
- Simply finding the solution by trial-and-error is considered cheating.

■

2.2 Linear models

2.2.1 What goes on in least squares?

A linear multiple regression model is formulated as:

$$y = X\beta + e$$

where y is an n-vector, X is a $n \times p$ matrix of covariates, β is a p-vector of parameters and e is an n-vector of residuals (errors) usually taken as independent normal variates, with identical variance σ^2, say.

Hence:

$$y \sim \mathcal{N}(X\beta, \sigma^2 I_n)$$

where I_n is the $n \times n$ identity matrix.

The least squares fitted values are $X\hat{\beta} = X(X^\top X)^{-1}X^\top y$.

This is the projection of the y-vector on the column space of X; it represents the vector which has the smallest distance to y and which is a linear combination of the columns of X. The matrix applied to y is called the projection matrix

$$\mathcal{P}_X = X(X^\top X)^{-1}X^\top$$

In least squares regression the fitted value are $\mathcal{P}_X y$. The projection is the matrix that satisfies that the difference between y and its projection is orthogonal to the columns in X

$$(y - \mathcal{P}_X y) \perp X$$

The orthogonality means that all of the columns in X are orthogonal to the columns of residuals, $y - \mathcal{P}_X y = (I - \mathcal{P}_X)y$, for any y. Hence any column in X should be orthogonal to any column in $(I - \mathcal{P}_X)$, so if we take the transpose of X (which is $p \times n$ and multiply with the matrix $(I - \mathcal{P}_X)$ (which is $n \times n$) we should get 0:

$$X^\top(I - \mathcal{P}_X) = X^\top - X^\top X(X^\top X)^{-1}X^\top = X^\top - X^\top = 0$$

The orthogonality was formulated as:

$$(I - \mathcal{P}_X) \perp X \quad \Leftrightarrow \quad X^\top(I - \mathcal{P}_X) = 0$$

Orthogonality of two vectors means that the inner product of them is 0. But this requires an inner product, and we have implicitly assumed that the inner product between two n-vectors was defined as:

$$\langle a|b\rangle = \sum_i a_i b_i = a^\top b$$

But for any positive definite matrix M we can define an inner product as:

$$\langle a|b\rangle = a^\top M b$$

In particular we can use a diagonal matrix with positive values in the diagonal

$$\langle a|b\rangle = a^\top W b = \sum_i a_i w_i b_i$$

A projection with respect to this inner product on the column space of X is:

$$\mathcal{P}_W = X(X^\top W X)^{-1} X^\top W$$

Exercise 2.2.1

Show that

$$(y - \mathcal{P}_W y) \perp_W X$$

where $\perp_W$ is orthogonality with respect to the inner product induced by the diagonal matrix W.
■

2.2.2 Projections in `Epi`

As part of the machinery used in `apc.fit`, the `Epi` package has a function that perform this type of a projection, `projection.ip`, the "`ip`" part referring to the possibility of using any inner product defined by a diagonal matrix (*i.e.* weights, w_i as above).

The reason that a special function is needed for this, is that a blunt use of the matrix formula above will set up an $n \times n$ projection matrix, which will be prohibitive if $n = 10,000$, say.

2.2.3 Constructing confidence intervals

2.2.3.1 Single parameters

One handy usage of matrix algebra is in construction of confidence intervals of parameter functions. Suppose we have a vector of parameter estimates $\hat{\beta}$ with corresponding estimated standard deviations $\hat{\sigma}$, both p-vectors, say.

The estimates with confidence intervals are:

$$\hat{\beta}, \quad \hat{\beta} - 1.96\hat{\sigma}, \quad \hat{\beta} + 1.96\hat{\sigma}$$

These three columns can be computed from the first columns by taking the $p \times 2$ matrix $(\hat{\beta}, \hat{\sigma})$ and post multiplying it with the 2×3 matrix:

$$\begin{matrix} 1 & 1 & 1 \\ 0 & -1.96 & 1.96 \end{matrix}$$

This is implemented in the function `ci.mat` in the `Epi` package:

```
> library(Epi)
> ci.mat()

     Estimate      2.5%    97.5%
[1,]        1  1.000000 1.000000
[2,]        0 -1.959964 1.959964

> ci.mat(alpha=0.1)

     Estimate      5.0%    95.0%
[1,]        1  1.000000 1.000000
[2,]        0 -1.644854 1.644854
```

So for a set of estimates and standard deviations we get:

```
> beta <- c(1.83,2.38)
> se <- c(0.32,1.57)
> cbind(beta,se) %*% ci.mat()

     Estimate       2.5%    97.5%
[1,]     1.83  1.2028115 2.457188
[2,]     2.38 -0.6971435 5.457143

> cbind(beta,se) %*% ci.mat(0.1)

     Estimate       5.0%    95.0%
[1,]     1.83  1.3036468 2.356353
[2,]     2.38 -0.2024202 4.962420
```

2.2.4 Showing an estimated curve

Suppose that we model an age-effect as a quadratic in age; that is we have a design matrix with the three columns $[1|a|a^2]$ where a is the ages. If the estimates for these three columns are $(\hat{\alpha}_0, \hat{\alpha}_1, \hat{\alpha}_2)$, then the estimated age effect is $\hat{\alpha}_0 + \hat{\alpha}_1 a + \hat{\alpha}_2 a^2$, or in matrix notation:

$$\begin{pmatrix} 1 & a & a^2 \end{pmatrix} \begin{pmatrix} \hat{\alpha}_0 \\ \hat{\alpha}_1 \\ \hat{\alpha}_2 \end{pmatrix}$$

If the estimated variance-covariance matrix of $(\alpha_0, \alpha_1, \alpha_2)$ is $\mathbf{\Sigma}$ (a 3×3 matrix), then the variance of the estimated effect at age a is:

$$\begin{pmatrix} 1 & a & a^2 \end{pmatrix} \mathbf{\Sigma} \begin{pmatrix} 1 \\ a \\ a^2 \end{pmatrix}$$

This rather tedious approach is an advantage if we simultaneously want to compute the estimated rates at several ages $a_1, a_2, \ldots, a_n$, say. The estimates and the variance covariance of these are then:

$$\begin{pmatrix} 1 & a_1 & a_1^2 \\ 1 & a_2 & a_2^2 \\ \vdots & \vdots & \vdots \\ 1 & a_n & a_n^2 \end{pmatrix} \begin{pmatrix} \hat{\alpha}_0 \\ \hat{\alpha}_1 \\ \hat{\alpha}_2 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & a_1 & a_1^2 \\ 1 & a_2 & a_2^2 \\ \vdots & \vdots & \vdots \\ 1 & \cdots & a_n^2 \end{pmatrix} \mathbf{\Sigma} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_3 \\ a_1^2 & a_2^2 & \cdots & a_3^2 \end{pmatrix}$$

The matrix we use to multiply with the parameter estimates is the age-part of the design matrix we would have if observations were for ages $a_1, a_2, \ldots, a_n$. The product of this piece of the design matrix and the parameter vector represents the function $f(a)$ evaluated in the ages $a_1, a_2, \ldots, a_n$.

We can illustrate this approach by an example from the `Epi` package where we model the birth weight as a function of the gestational age, using the `births` dataset:

```
> data(births)
> str(births)

'data.frame':        500 obs. of  8 variables:
 $ id     : num  1 2 3 4 5 6 7 8 9 10 ...
 $ bweight: num  2974 3270 2620 3751 3200 ...
 $ lowbw  : num  0 0 0 0 0 0 0 0 0 0 ...
 $ gestwks: num  38.5 NA 38.2 39.8 38.9 ...
 $ preterm: num  0 NA 0 0 0 0 0 0 0 0 ...
 $ matage : num  34 30 35 31 33 33 29 37 36 39 ...
 $ hyp    : num  0 0 0 0 1 0 0 0 0 0 ...
 $ sex    : num  2 1 2 1 1 2 2 1 2 1 ...

> ma <- lm( bweight ~ gestwks + I(gestwks^2), data=births )
> ( beta <- coef( ma ) )

 (Intercept)      gestwks I(gestwks^2)
-8247.693621   406.461711    -2.893052

> ( cova <- vcov( ma ) )

             (Intercept)      gestwks I(gestwks^2)
(Intercept)  5307195.933 -292330.6903  3995.943420
gestwks      -292330.690   16204.3875  -222.720377
I(gestwks^2)    3995.943    -222.7204     3.075777
```

If we want to predict the birth weight for gestational ages 32 to 42 weeks, say, we just set up the matrices needed and perform the computations:

```
> ga.pts <- 32:42
> G <- cbind( 1, ga.pts, ga.pts^2 )
> wt.eff <- G %*% beta
> sd.eff <- sqrt( diag(G %*% cova %*% t(G)) )
> wt.pred <- cbind( wt.eff, sd.eff ) %*% ci.mat()
> matplot( ga.pts, wt.pred, type="l", lwd=c(3,1,1), col="black", lty=1 )
```

This machinery has been implemented in the function `ci.lin` which takes a subset of the parameters from the model (selected via `grep`), and multiplies them (and the corresponding variance-covariance matrix) with a supplied matrix:

```
> wt.ci <- ci.lin( ma, ctr.mat=G )[,c(1,5,6)]
> matplot( ga.pts, wt.ci, type="l", lty=1, lwd=c(3,1,1), col="black" )
```

The example with the age-specific birth weights is a bit irrelevant because they could have been obtained by the `predict` function.

The interesting application is with more than one continuous variable, say gestational age and maternal age. Suppose we have a quadratic effect of both gestational age and maternal age:

```
> ma2 <- lm( bweight ~ gestwks + I(gestwks^2) +
+                      matage  + I(matage^2), data=births )
> ci.lin( ma2 )

                  Estimate      StdErr          z           P          2.5%
(Intercept)  -7404.0059740 2592.084329 -2.8563909 0.004284873 -12484.397903
gestwks        409.2978884  127.755598  3.2037570 0.001356469    158.901518
I(gestwks^2)    -2.9284416    1.759843 -1.6640354 0.096105363     -6.377671
matage         -53.8976266   75.249616 -0.7162512 0.473836261   -201.384163
I(matage^2)      0.7959908    1.118262  0.7118107 0.476582033     -1.395762
                    97.5%
(Intercept)  -2323.614045
gestwks        659.694259
I(gestwks^2)     0.520788
matage          93.588910
I(matage^2)      2.987744
```

and that we want to show the effect of gestational age for a maternal age of 35 (pretty close

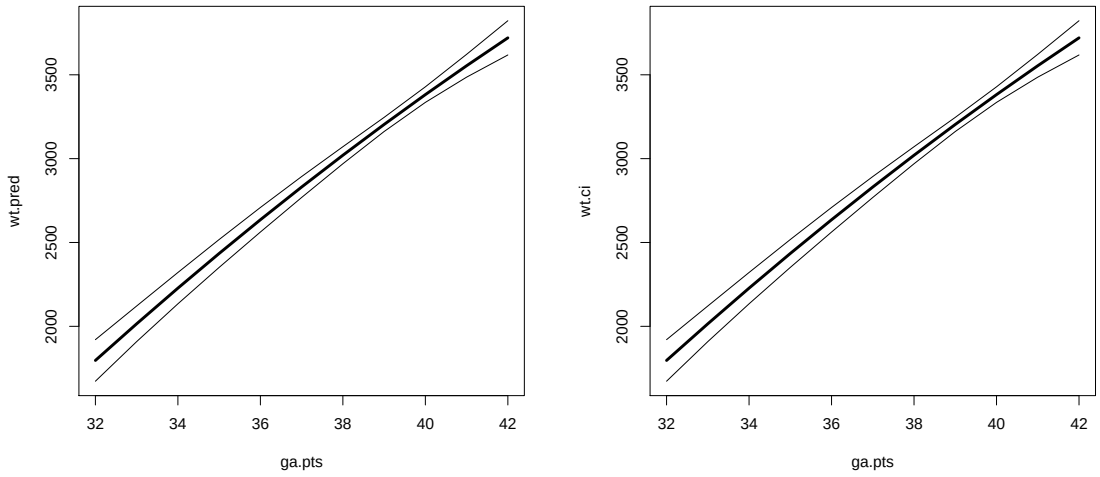

Figure 2.2: *Prediction of the gestational age-specific birth weight. Left panel computed "by hand", right panel using* `ci.lin`.

to the median) and the effect of maternal age relative to this reference point. (These two curves can be added to give predicted values of birth weight for any combination of gestational age and maternal age.)

First we must specify the reference point for the maternal age, and also the points for maternal age for which we want to make predictions

```
> ma.ref <- 35
> ma.pts <- 20:45
> M <- cbind(ma.pts,ma.pts^2)
> M.ref <- cbind(ma.ref,ma.ref^2)
```

Then we derive the estimated birth weight for a maternal age 35, as function of gestational age. To this end we need to have a contrast matrix which in each row has the 1, the gestational age, gestational age squared, the latter two for all the gestational ages of interest (that is the matrix `G` defined above) and the reference maternal age and the same squared (i.e. identical rows):

```
> bw.ga <- ci.lin( ma2, ctr.mat=cbind(G,M.ref[rep(1,nrow(G)),]) )[,c(1,5,6)]
> matplot( ga.pts, bw.ga, type="l", lty=1, lwd=c(2,1,1), col="black" )
```

Then to show the effect of maternal age on birth weight relative to the maternal age reference of 35, we sue only the maternal age estimates, but subtract rows corresponding to the reference point:

```
> bw.ma <- ci.lin( ma2, subset="matage", ctr.mat=M-M.ref[rep(1,nrow(M)),] )[,c(1,5,6)]
> matplot( ma.pts, bw.ma, type="l", lty=1, lwd=c(2,1,1), col="black" )
```

Exercise 2.2.2

Redo the two graphs in figure 2.3 with y-axes that have the same extent (in grams birth weight).

Why would you want to do that?

■

2.2.4.1 Splines

The real advantage is however when simple functions such as the quadratic is replaced by spline functions, where the model is a parametric function of the variables, and implemented in the

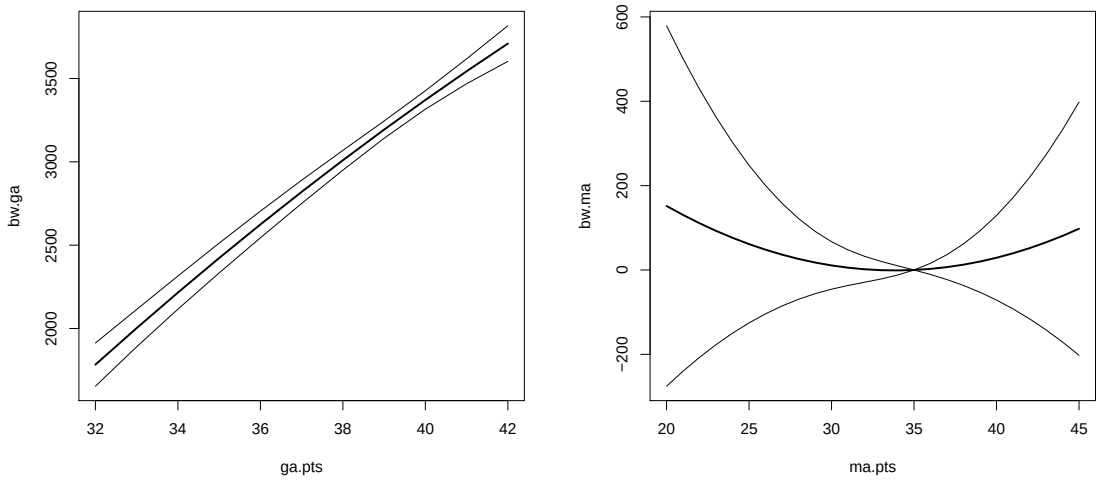

Figure 2.3: *Prediction of the gestational age-specific birth weight for maternal age 35 (left) and the effect of maternal age on birth weight, relative to age 35.*

functions `ns` and `bs` from the `splines` package.

Exercise 2.2.3

Repeat the exercise above, where you replace the quadratic terms with natural splines (using `ns()` with explicit knots), both in the model and in the prediction and construction of predictions and contrasts.

■

2.2.5 Reparametrizations

Above we saw how you can compute linear functions of estimated parameters using `ci.lin`. But suppose you have a model parametrized by the linear predictor $X\beta$, but that you really wanted the parametrization $A\gamma$, where the columns of X and A span the same linear space. So $X\beta = A\gamma$, and we assume that both X and A are of full rank, $\dim(X) = \dim(A) = n \times p$, say.

We want to find γ given that we know $X\beta$ and that $X\beta = A\gamma$. Since we have that $p < n$, we have that $A^- A = I$, by the properties of G-inverses, and hence:

$$\gamma = A^- A\gamma = A^- X\beta$$

The essences of this formula are:

1. Given that you have a set of fitted values in a model (*in casu* $\hat{y} = X\beta$) and you want the parameter estimates you would get if you had used the model matrix A. Then they are $\gamma = A^- \hat{y} = A^- X\beta$.

2. Given that you have a set of parameters β, from fitting a model with design matrix X, and you would like the parameters γ, you would have got had you used the model matrix A. Then they are $\gamma = A^- X\beta$.

The latter point can be illustrated by linking the two classical parametrizations of a model with a factor, either with or without intercept:

```
> FF <- factor( rep(1:3,each=2) )
> ( X <- model.matrix(~FF  ) )

  (Intercept) FF2 FF3
1           1   0   0
2           1   0   0
3           1   1   0
4           1   1   0
5           1   0   1
6           1   0   1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$FF
[1] "contr.treatment"

> ( A <- model.matrix(~FF-1) )

  FF1 FF2 FF3
1   1   0   0
2   1   0   0
3   0   1   0
4   0   1   0
5   0   0   1
6   0   0   1
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$FF
[1] "contr.treatment"

> library(MASS)
> ginv(A) %*% X

     (Intercept) FF2 FF3
[1,]           1   0   0
[2,]           1   1   0
[3,]           1   0   1
```

The last resulting matrix is clearly the one that translates from an intercept and two contrasts to three group means.

The practical use of this may be somewhat limited, because the requirement is to know the "new" model matrix A, and if that is available, you might just fit the model using this.