Primer on vector:
Basic stuffs for neuroscientists

Cyril Pernet, PhD.

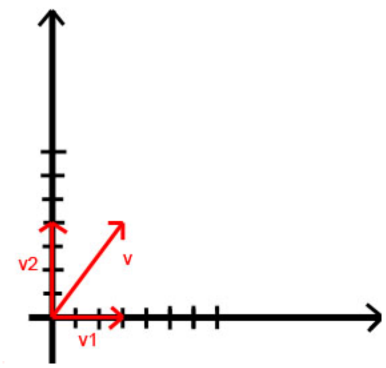
Notation

Normal text is in Time News Roman 12 with vector represented by letters in **bold** and the components of a vector (see bellow) in *italic*. The text comes with some matlab code in `Courier New 10`.

Vectors in 2D space

A vector can be represented geometrically as an arrow with an initial and a terminal point. Using a rectangular coordinate system and (0,0) as the initial point and (*v1*,*v2*) as terminal point one says that *v1* and *v2* are the components of **v**. This notation is very useful to describe the length of an object or the distance between points.

```
v1 = 3; v2 = 4; v = [v1,v2];
```

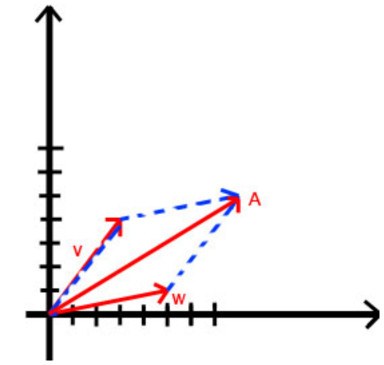


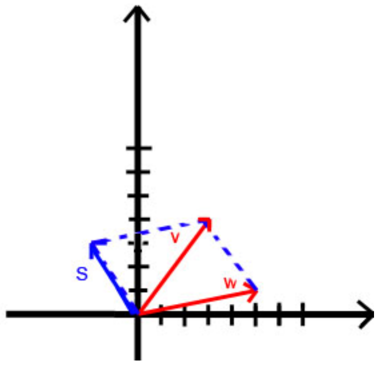
Arithmetic operations

Addition: (**v** + **w**) = (**w** + **v**)
(**v** + **w**) = (*v1* + *w1*, *v2* + *w2*)

Subtraction: (**v** - **w**) = (**v** + (-**w**))
(**v** - **w**) = (*v1*-*w1*, *v2*-*w2*)

```
v1 = 3; v2 = 4; v = [v1,v2];
w1 = 5; w2 = 1; w = [w1, w2];
A = [3+5, 4+1];
S = [3-5, 4-1];
```





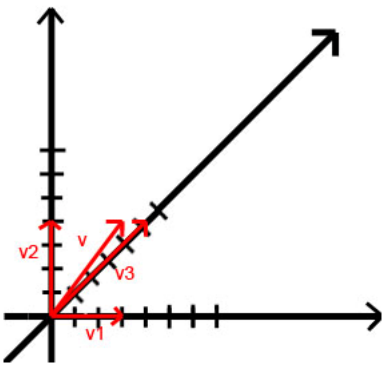
Multiplication by a scalar k: k**v** = (k*v1*, k*v2*)

```
v1 = 3; v2 = 4; v = [v1,v2];
k = 3; M1 = [3*3, 3*4];
```

Vectors in 3D space

Vectors are now described in a 3D space using triples of real numbers. Using a rectangular coordinate system, each number can be interpreted as a length along the different axes.

```
v1 = 3; v2 = 4; v3 = 6; v = [v1,v2, v3];
```



The length of a vector v is called the norm written ||**v**||. In 2D the norm of v is simply the square root of the sum of x and y (||**v**|| = sqrt(x^2+y^2)); this is the Pythagoras theorem - the same can be applied in 3D.

```
> sqrt(v1^2+v2^2+v3^2)
> sqrt(sum(v.^2))
> norm(v)
```

Dot product (also called Euclidian inner product)

For two vector **u** and **v** with a common origin, it exists an angle theta between 0 and pi between those vectors. The dot product is the product of the lengths by the cosinus of the angle: **u.v** = ||**u**||*||**v**||*cos(theta)

```
v = [0 2 2]; u = [0 0 1];
d = dot(u,v);
theta = acosd(d / (norm(u)*norm(v)));
% acosd = inverse of cosinus in degrees
```

The dot product can also be computed as the sum of the product of each component (hence the inner product name): **u.v** = (*u1*v1*) + (*u2*v2*) + (*u3*v3*)

Orthogonal vectors and projection

Two vectors are orthogonal (theta = 90) if **u.v** = 0.

We can thus create two vectors (*w1* and *w2*) such as **u** = *w1* + *w2* and *w1*.*w2* = 0. Say now we want to know the length of **u** along a vector **a** then we can decompose u such as *w1* is along **a** and *w2* is perpendicular to **a**. *w1* is called the orthogonal projection of **u** *on* **a.**

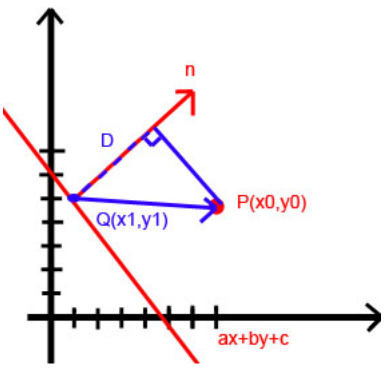
We start with
u = *w1* + *w2*
*w1***.***w2* = 0
w1 = k**a**

It follows
u = k**a** + **w2**
u.a = (k**a**+*w2*)**.a** = k||**a**||^2 + *w2***.a**
*w2***.a** = 0 ⇒ **u.a** = k||**a**||^2
k = **u.a** / ||**a**||^2

since *w1* = k**a,** w1 = (**u.a** / ||**a**||^2)**a**

The most common application of this kind of computation is for finding the shortest distance between a line and a point (understand shortest as perpendicular).

Say we have a line described such as ax+by+c = 0 and we want to know the distance of the point P(x0,y0). The distance can be computed simply as the orthogonal projection of the vector QP on n with Q(x1,y1) being a point on the line and n(a,b) a perpendicular vector to the line with origin Q.



D = |QP . n| / ||n||
QP = (x0-x1, y0-y1)
QP.n = a(x0-x1) + b(y0-y1)
||n|| = sqrt(a^2+b^2)

D = |ax0 + by0 + c| / ||n||

⇒ 3x+4y-6 = 0 and P(1,-2)
⇒ D = -11 / 5

<u>Cross product – 3D spaces</u>

The notion of cross product is useful in the context of orthogonal vectors in 3D spaces (or more); for two vectors **u**(*u1*, *u2*, *u3*) and **v**(*v1*, *v2*, *v3*) we have

u * **v** = (*u2v3 – u3v2*, *u3v1 – u1v3*, *u1v2 – u2v1*)

The relationships between dot product and cross product are

- **u** * (**v** * **w**) = (**u.w**)**v** - (**u.v**)**w**
- (**u** * **v**) * **w** = (**u.w**)**v** – (**v.w**)**u**
- || **u** * **v**||^^{2} = ||**u**||^^{2}||**v**||^^{2} – (**u.v**)^^{2} % Lagrange's identity
- **u.**(**u** * **v**) = 0 % **u** * **v** is orthogonal to **u**
- **v.**(**u** * **v**) = 0 % **u** * **v** is orthogonal to **v**

<u>Euclidian Vector Spaces</u>

All the operations describe above work in higher dimensional spaces. For instance if one measures a variable (e.g. RT) 100 times in a subject, then the data can be represented as a vector **y** with 100 values. This vector is also a point in R^{100} and usual statistics will look for a particular combination of vectors to be as close as possible to this point (e.g. RT=ax1+bx2+e).

Linear transformations

One very useful application of linear algebra and vector related arithmetic is linear transformation, i.e. how to move from one space to another. In medical imaging it is use to flip images, align, normalize, etc.

Linear transformation can be viewed as usual functions; for instance f(x)=x^2 is a real valued function which allows to move from R to R^2. Bellow is listed a series of operators for affine transformations

Application in R^2

Operation	*Equation*	*Matrix notation*
Reflection about the y-axis	w1 = -x w2 = y	$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$
Reflection about the x-axis	w1 = x w2 = -y	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Reflection about the line y=x	w1 = y w2 = x	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Orthogonal projection on the x-axis	w1 = x w2 = 0	$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$
Orthogonal projection on the x-axis	w1 = 0 w2 = y	$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$
Rotation through an angle θ	w1 = x cosθ - y sinθ w2 = x sinθ + y cosθ	$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
Contraction/Dilatation by a factor k	w1 = kx w2 = ky	$\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$

Application in R^3

Operation	*Equation*	*Matrix notation*
Reflection about the xy-plane	w1 = x w2 = y w3 = -z	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$
Reflection about the xz-plane	w1 = x w2 = -y w3 = z	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Reflection about the yz-plane	w1 = -x w2 = y w3 = z	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Orthogonal projection on the xy-plane	w1 = x w2 = y w3 = 0	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Orthogonal projection on the xz-plane	w1 = x w2 = 0 w3 = y	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Orthogonal projection on the yz-plane	w1 = 0 w2 = y w3 = z	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Counterclockwise rotation along x trough an angle θ (pitch)	w1 = x w2 = x cosθ - z sinθ w3 = y sinθ + z cosθ	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$
Counterclockwise rotation along y trough an angle θ (roll)	w1 = x cosθ + z sinθ w2 = y w3 = -x sinθ + z cosθ	$\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$
Counterclockwise rotation along z trough an angle θ (yaw)	w1 = x cosθ - y sinθ w2 = x sinθ + y cosθ w3 = z	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Contraction/Dilatation by a factor k	w1 = kx w2 = ky w3 = kz	$\begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & k \end{bmatrix}$

The matrix notation is often seen when it come to manipulate and display 3D images – the transformation matrix is stored in the header so that the image is correctly displayed; for instance in SPM (http://www.fil.ion.ucl.ac.uk/spm/) the matrix appears in full