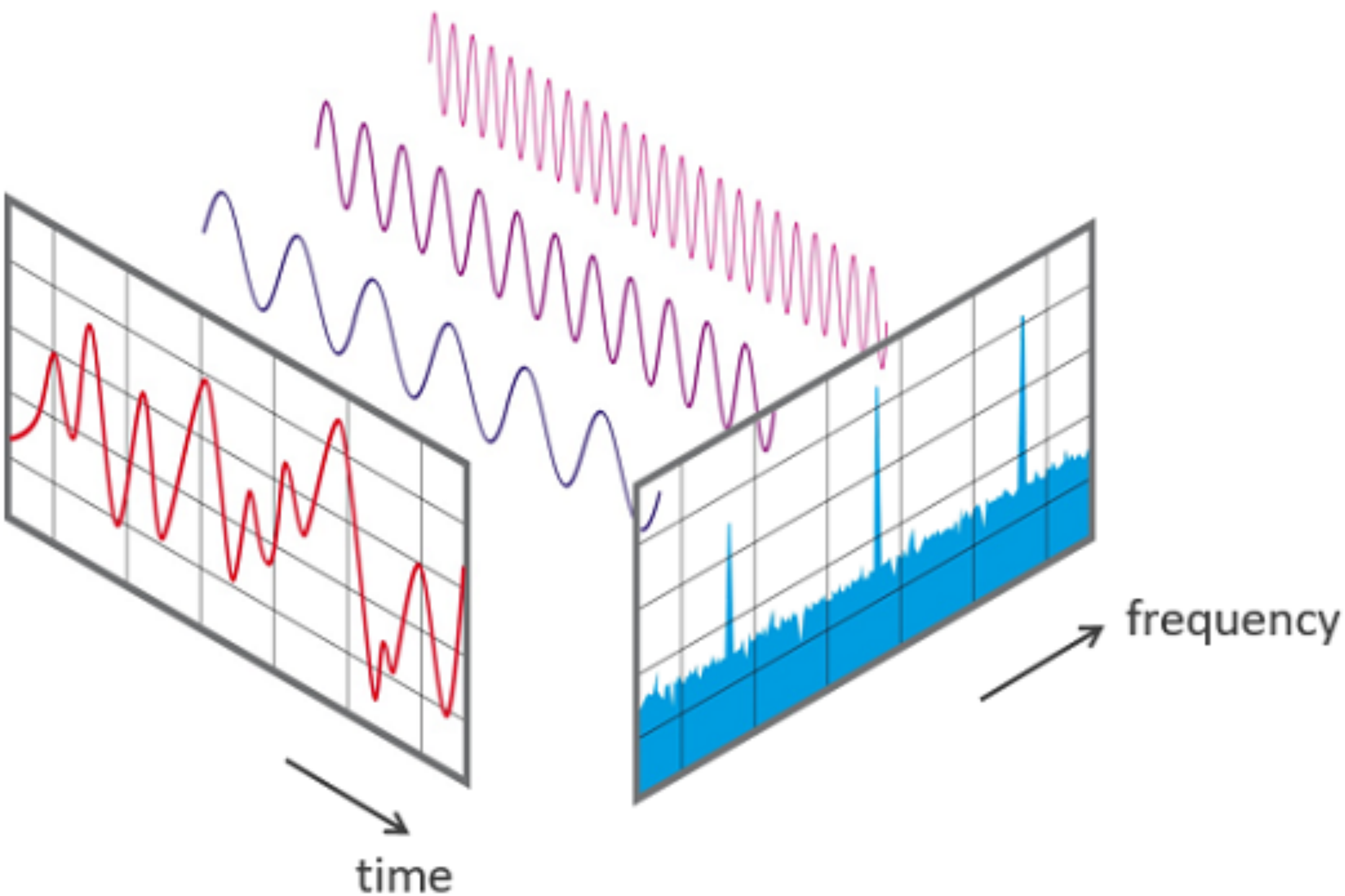
Math for Time Frequency Methods

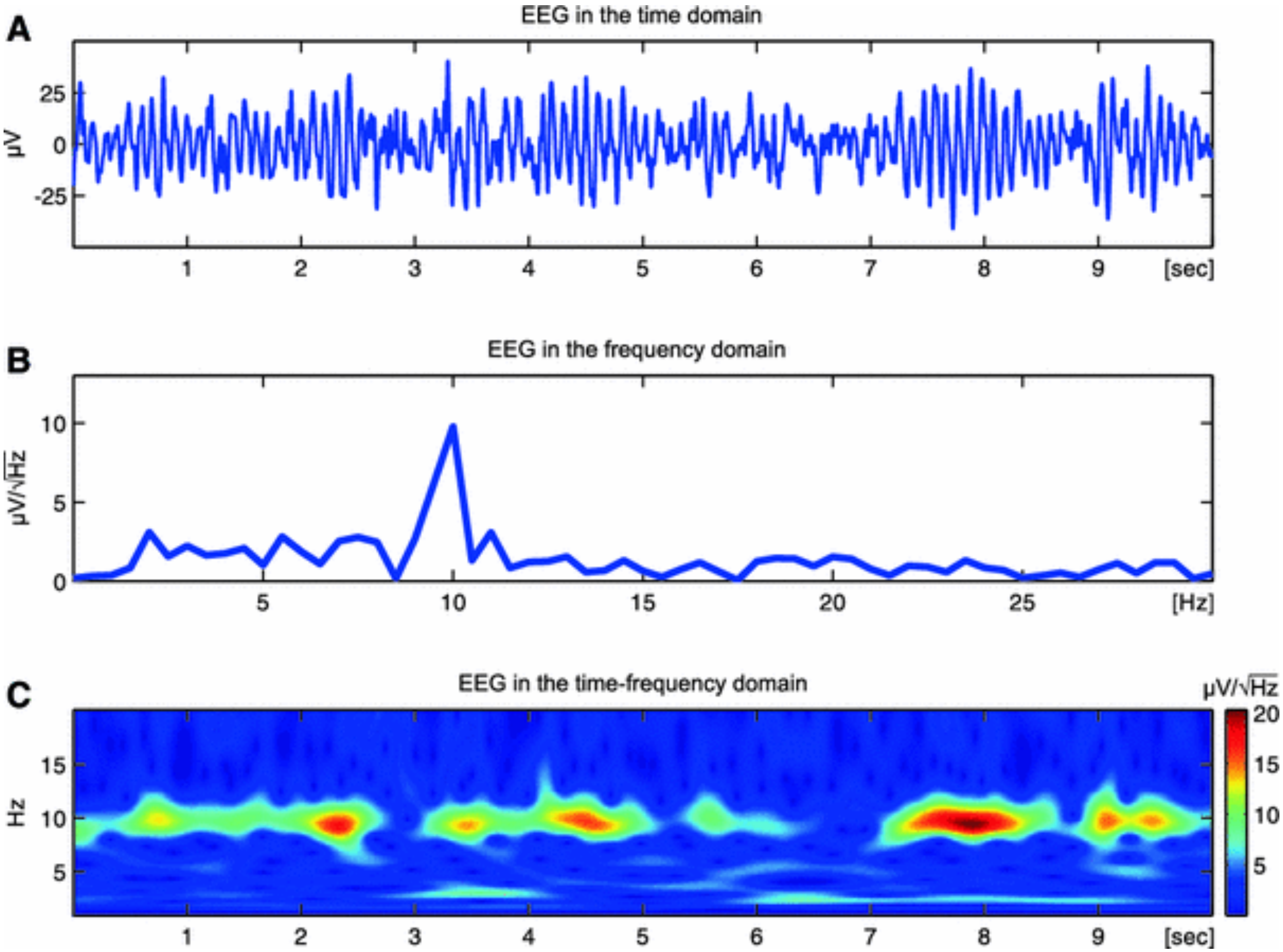
Jon Sprouse
University of Connecticut

Remember the two representations?

Important idea: EEG signals can be represented in two ways: the **time domain representation** and the **frequency domain representation**.



Time-frequency analysis of EEG



Vectors

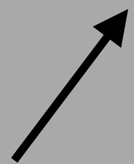
Watch this video (~9 minutes): https://www.youtube.com/watch?v=fNk_zzaMoSs&t=0s&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab&index=2

Watch this video up to 3:00: https://www.youtube.com/watch?v=k7RM-ot2NWY&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab&index=2

Watch this video up to 4:00: https://www.youtube.com/watch?v=LyGKycYT2v0&index=9&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab

Vectors and EEG

Arrow in space



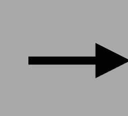
=

Ordered set

[2 3]

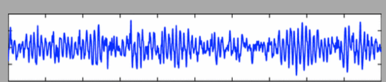
+

Bases



x y

EEG signal



=

Ordered set of amplitudes

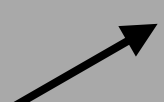
[2 3 1 4 3 2 3 ...]

+

Time

1 2 3 4 5 6 7 ...
(milliseconds)

EEG signal Arrow in space



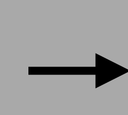
=

Ordered set of amplitudes

[2 3 1 4 3 2 3 ...]

+

Bases



x y

Does vector math (linear algebra) help us?

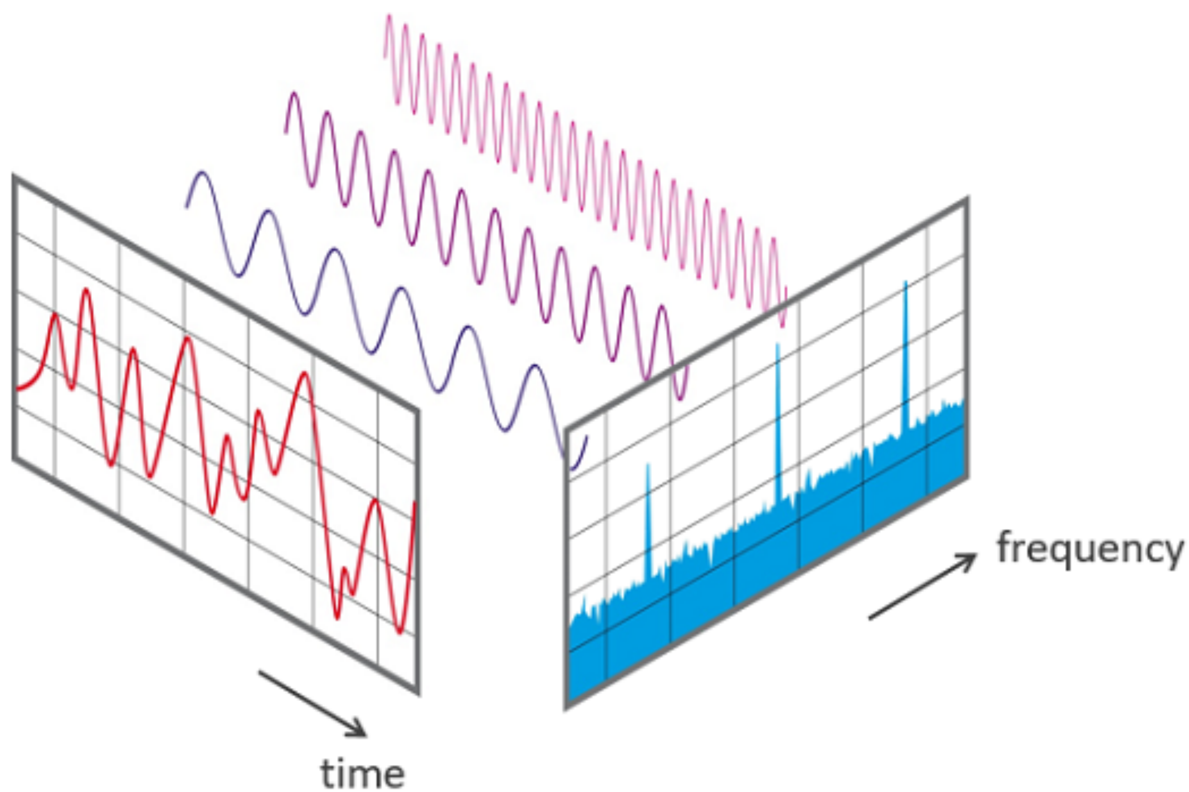
Important idea: Math exists to help us. First we need to figure out what we want to achieve, and then we can look for math that achieves it for us.

So we have two questions: (i) what do we want to achieve?, and (ii) will vector math help us achieve it? If so, then we can treat the EEG signal like a vector to take advantage of vector math!

So let's answer question 1.

What we want to achieve is to calculate how similar our EEG signal is to a perfect sine wave of a specific frequency.

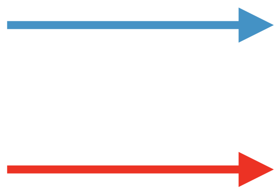
(And then we want to check all of the other frequencies too! And then do it over time!)



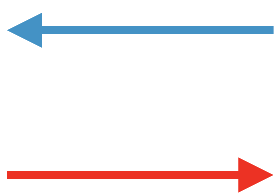
The dot-product

We want a measure of vector similarity

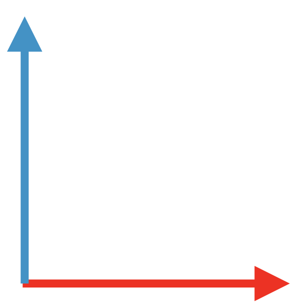
What would it mean for two vectors to be similar? Let's look at some examples, and decide what we would say about their similarity

1. 

These two are perfectly similar. Maybe we make them the maximum of our similarity score. Maybe 100% or 1.

2. 

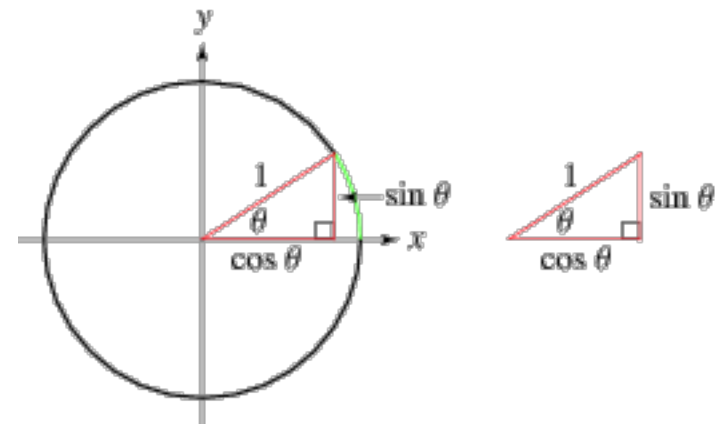
These two are perfectly similar, but opposites. Maybe we make them the maximum of our similarity score, but negative. Maybe -100% or -1.

3. 

What about these? They are orthogonal to each other. They aren't similar at all. Maybe give these a 0?

Cosine is similarity based on angle.

Cosine is a function that takes the angle between a vector and the x-axis, and returns the x-coordinate of the vector.



If we assume blue is 1 unit long, and treat the red vector as the x-axis, the cosine of the angle of the blue will be the similarity values that we want.

1. 

cosine = 1

2. 

cosine = -1

3. 

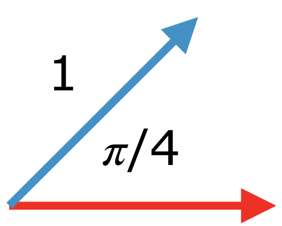
cosine = 0

This is why you will often see different kinds of “cosine similarity” tests in stats, or read about how covariance is basically a scaled cosine.

Adding magnitude: projection

Vectors have both direction and magnitude. Cosine is a measure of similarity that only takes direction into account. We want to add magnitude.

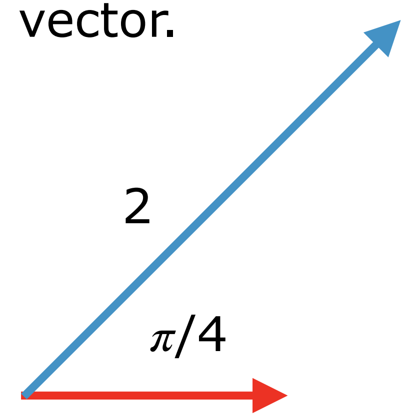
One thing we can do is scale the cosine by the magnitude of the blue vector, i.e., multiply cosine by the length of the blue vector.



cos = .71

|blue| * cos θ

1 * .71 = .71



cos = .71

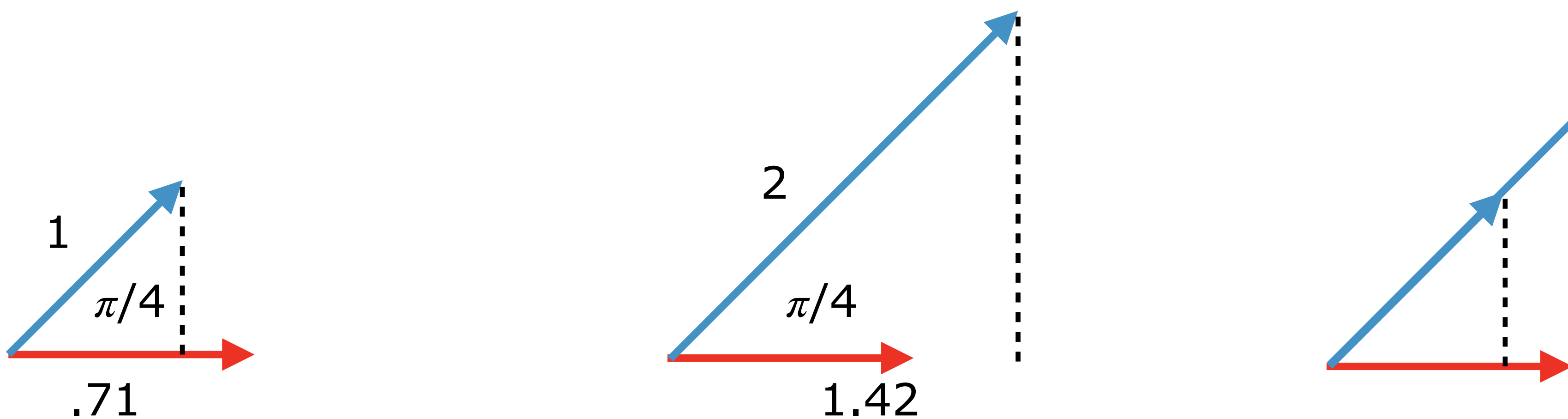
|blue| * cos θ

2 * .71 = 1.42

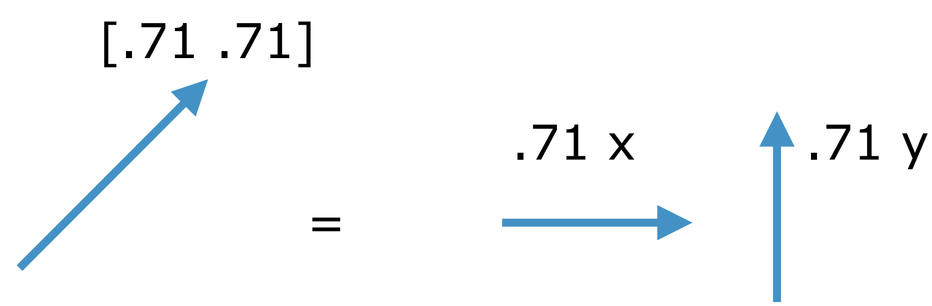
Adding magnitude: projection

This is the mathematical equation for **projection**. It is the length of the blue vector projected onto red vector (treating the red vector as an axis or basis).

projection = |blue| * cos θ



Remember that we can decompose any vector into scalars and bases.

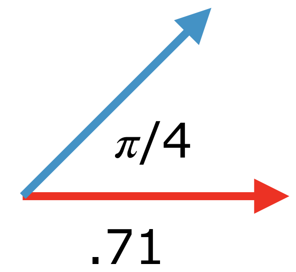


Projection says "treat the red vector as a basis, and decompose the blue one into its scalars… what is blue's scalar in the red basis?" It tells us **how much of the blue magnitude goes in the red direction.**

Adding the other magnitude: the dot product

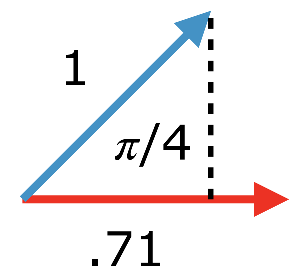
Cosine tells us how similar two vectors are in direction. It ranges from -1 to 1.

cos θ



Projection tells us how much of blue's magnitude goes in the red direction.

|blue| * cos θ



But we still haven't used the red magnitude. The **dot product** does this:

dot product = |red| * |blue| * cos θ

The dot product scales the projection by the length of the red vector. In other words, it treats the red vector like a unit vector, and scales the projection to the length it should be if the red were the unit vector.

Dot-product is (unscaled) similarity

The dot product combines both magnitude and direction into one equation for similarity:

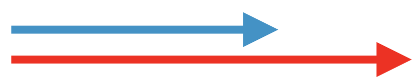
|red| * |blue| * cos θ

The trick to seeing it as a similarity measure is to realize that it is **unscaled** - it grows with the magnitudes of the vectors. But if you stick to a family of angles, you can see that their relative dot products encode similarity of both the directions and the magnitudes.

|red| = 3
|blue| = 3
cos θ = 1

3 * 3 * 1 = 9

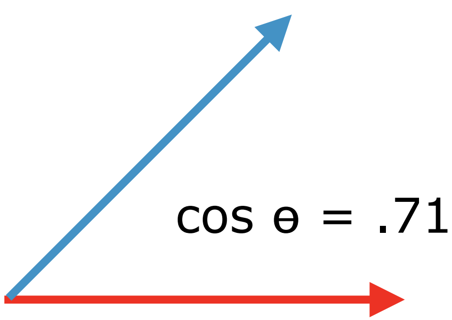
|red| = 3
|blue| = 2
cos θ = 1



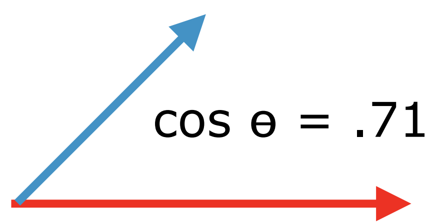
3 * 2 * 1 = 6

|red| = 3
|blue| = 1
cos θ = 1

3 * 1 * 1 = 3



3 * 3 * .71 = 6.4



3 * 2 * .71 = 4.3

3 * 1 * .71 = 2.1

Dot-product is (unscaled) similarity

To see this, we can scale the dot product by the **maximum product** if the two vectors were equal in length.

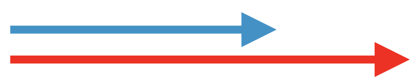
$$\frac{|red| * |blue| * \cos\theta}{\max(|red|,|blue|)^2}$$

The result is something that looks like a scaled similarity score (between -1 and 1). And, crucially, it takes both magnitude and direction into account.

|red| = 3
|blue| = 3
cos ɵ = 1

(3 * 3 * 1)/9 = **1**

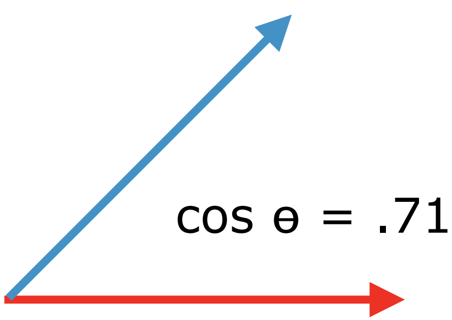
|red| = 3
|blue| = 2
cos ɵ = 1



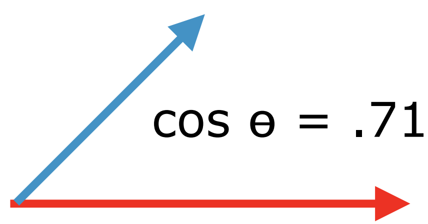
(3 * 2 * 1)/9= **.67**

|red| = 3
|blue| = 1
cos ɵ = 1

(3 * 1 * 1)/9 = **.33**



(3 * 3 * .71)/9 = **.71**



(3 * 2 * .71)/9 **.48**

(3 * 1 * .71)/9 = **.24**

The result doesn’t care which is longer

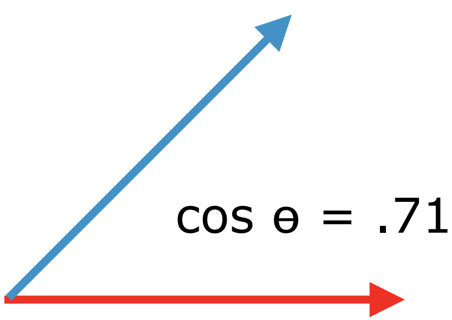
The dot product equation is symmetric. And the denominator for scaling chooses the largest value.

$$\frac{|red| * |blue| * \cos \theta}{\max(|red|,|blue|)^2}$$

This means that the result is the same, regardless of which vector is longer, both for the dot product itself (shown below), and for the scaled version (just divide by 9):

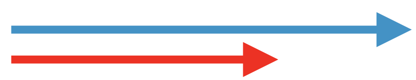
$|red| = 3$
$|blue| = 3$
$\cos \theta = 1$

$3 * 3 * 1 = 9$

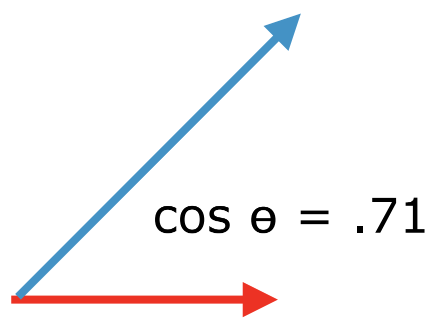


$3 * 3 * .71 = 6.4$

$|red| = 2$
$|blue| = 3$
$\cos \theta = 1$



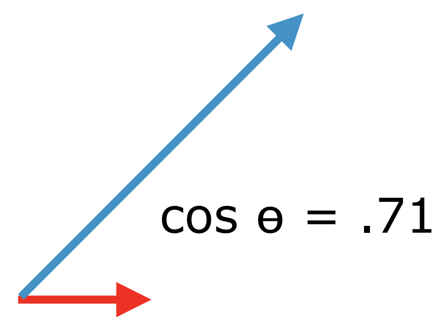
$2 * 3 * 1 = 6$



$2 * 3 * .71 = 4.3$

$|red| = 1$
$|blue| = 3$
$\cos \theta = 1$

$1 * 3 * 1 = 3$



$1 * 3 * .71 = 2.1$

Why not just use projection?

One could imagine taking the length of red into consideration differently… perhaps by calculating the projection, and dividing by it.

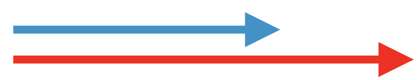
$$\frac{|\text{blue}| * \cos\theta}{|\text{red}|}$$

This looks just like the scaled dot product, so we might decide that this is an ok way to go…

|red| = 3
|blue| = 3
cos θ = 1

(3 * 1)/3 = **1**

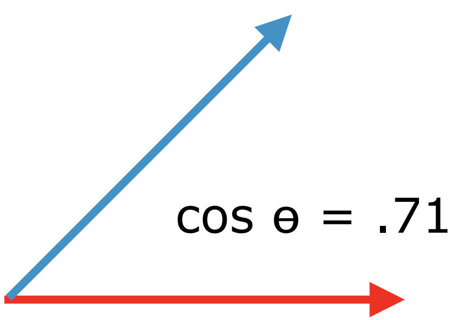
|red| = 3
|blue| = 2
cos θ = 1



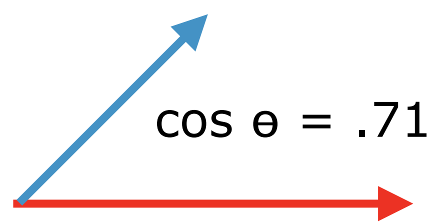
(2 * 1)/3= **.67**

|red| = 3
|blue| = 1
cos θ = 1

(1 * 1)/3 = **.33**



(3 * .71)/3 = **.71**



(2 * .71)/3 **.48**

(1 * .71)/3 = **.24**

But the result changes if we flip the lengths

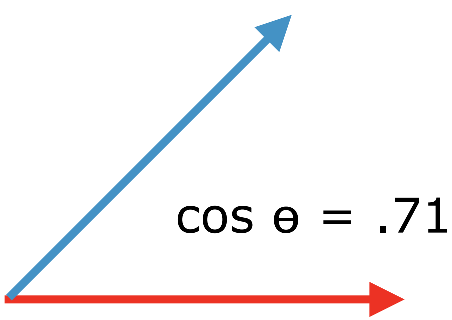
One could imagine taking the length of red into consideration differently… perhaps by calculating the projection, and dividing by it.

$$\frac{|\text{blue}| * \cos \theta}{|\text{red}|}$$

The problem is that this equation is inherently asymmetric — it is the projection of the blue, scaled by the length of the red. To use this, we would always need to choose the longer vector as the "red"…

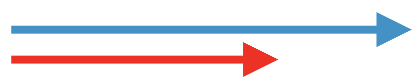
|red| = 3
|blue| = 3
cos θ = 1

(3 * 1)/3 = 1

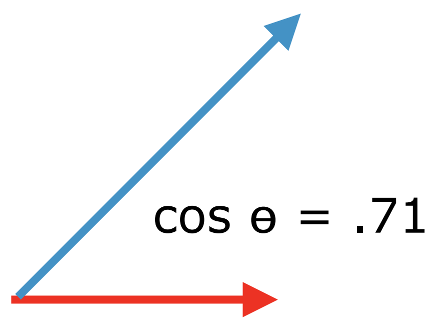


(3 * .71)/3 = .71

|red| = 2
|blue| = 3
cos θ = 1



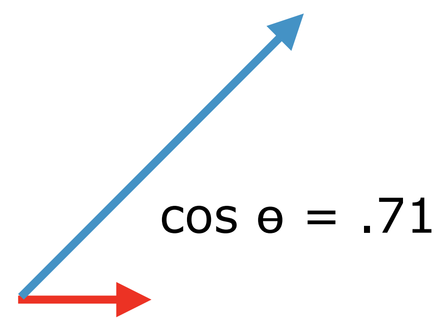
(3 * 1)/2 = .67



(3 * .71)/2 = 1.05

|red| = 1
|blue| = 3
cos θ = 1

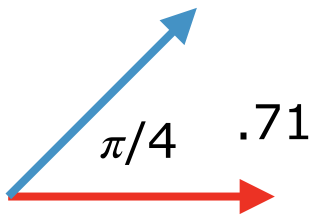
(3 * 1)/1 = 3



(3 * .71)/1 = 2.1

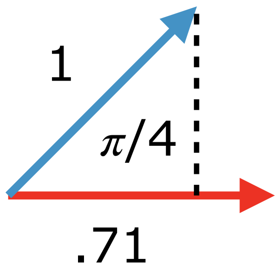
To recap

Cosine tells us how similar two vectors are in direction. It ranges from -1 to 1.



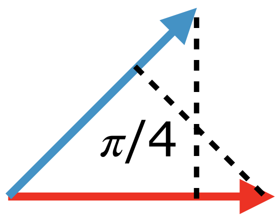
cos θ

Projection tells us how much of blue's magnitude goes in the red direction.



|blue| * cos θ

Dot product gives us an unscaled, symmetric measure of similarity that takes direction and both magnitudes into consideration.



|red| * |blue| * cos θ

Honesty — my trig is not good enough

I know that these two equations are identical (for 2D vectors):

dot product = |red| * |blue| * cos ө

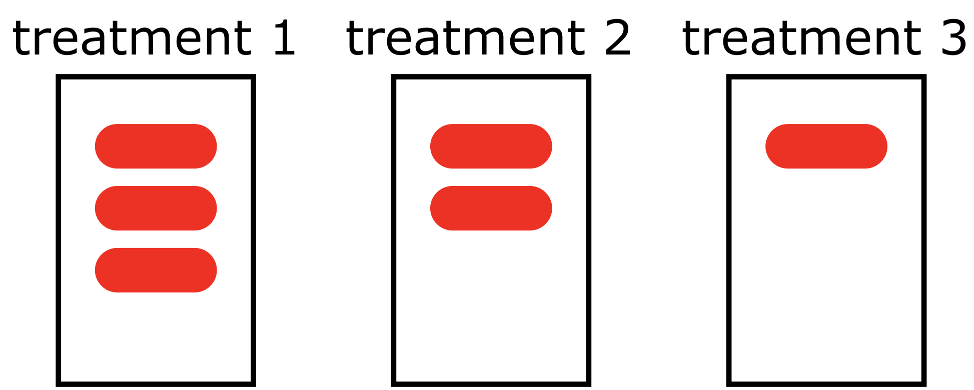
dot product = (x1*x2) + (y1*y2)

I can see that they will yield the same values by working through examples, but my trigonometry is not strong enough to derive one from the other.

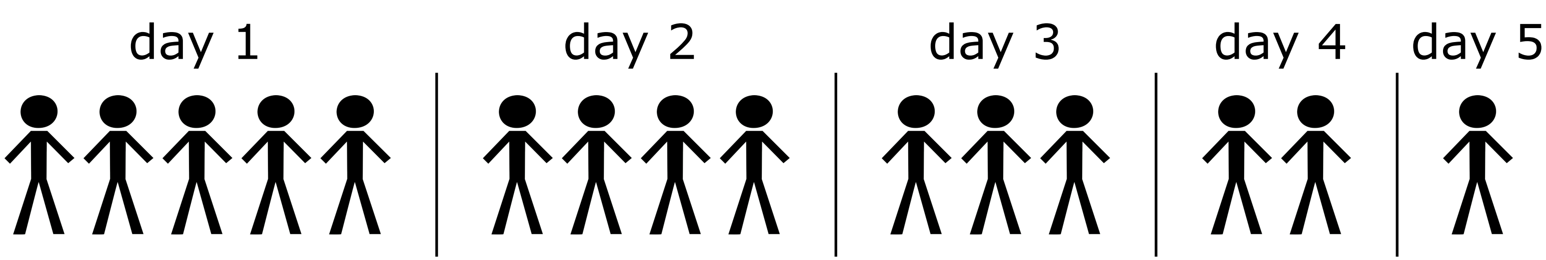
Convolution
(these slides only partially address the flipping

Convolution example from betterexplained.com

There is a disease that has a treatment plan that takes three treatments.
Treatment 1 is 3 pills, treatment 2 is 2 pills, and treatment 3 is 1 pill.



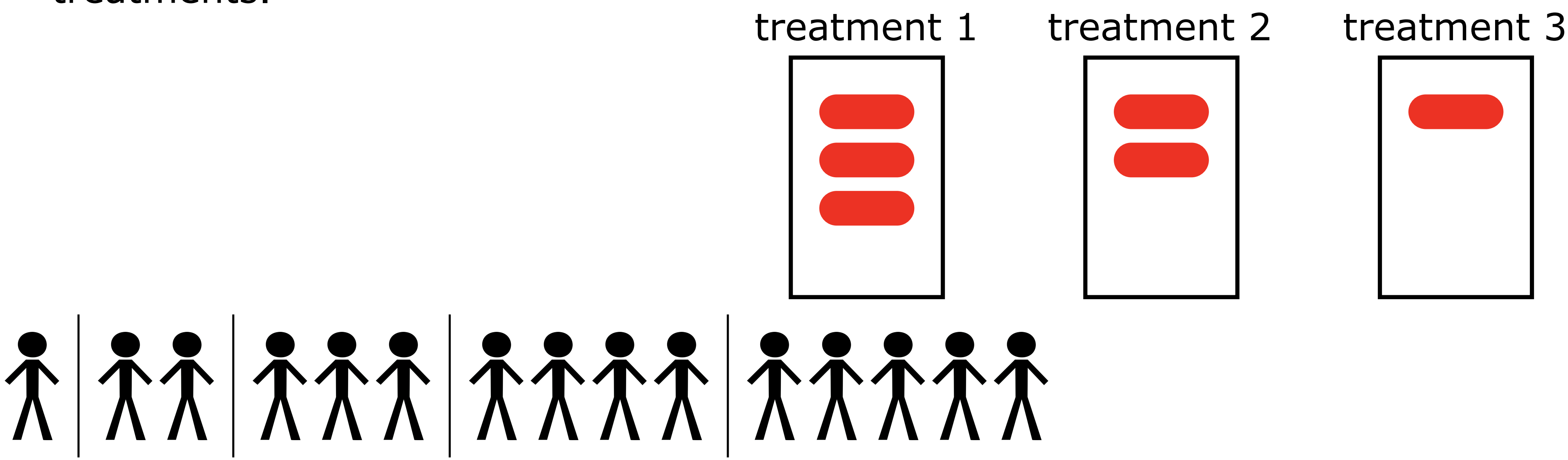
Now imagine you have a line of patients, organized by day.



How many pills do you need each day?

Convolution example from betterexplained.com

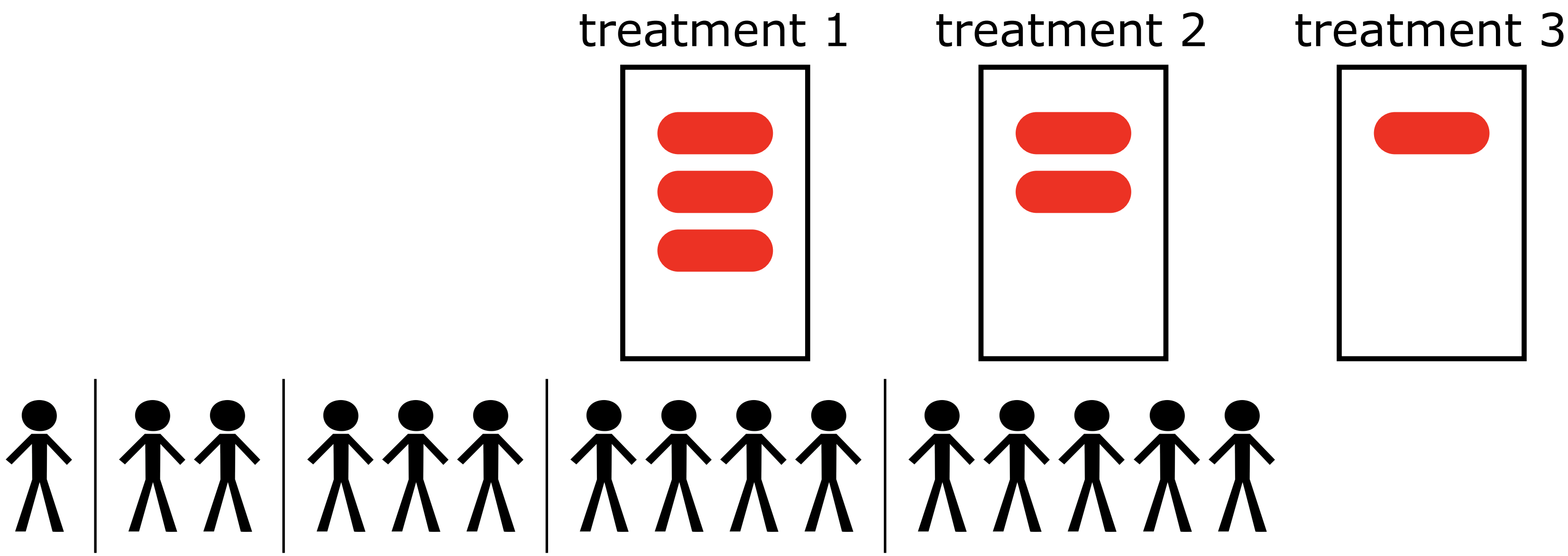
To figure this out, we can arrange the people and the treatments in the right order — by flinging the people around into a line that can file through the treatments.



day 1	15	**15**
day 2		
day 3		
day 4		
day 5		
day 6		
day 7		

Convolution example from betterexplained.com

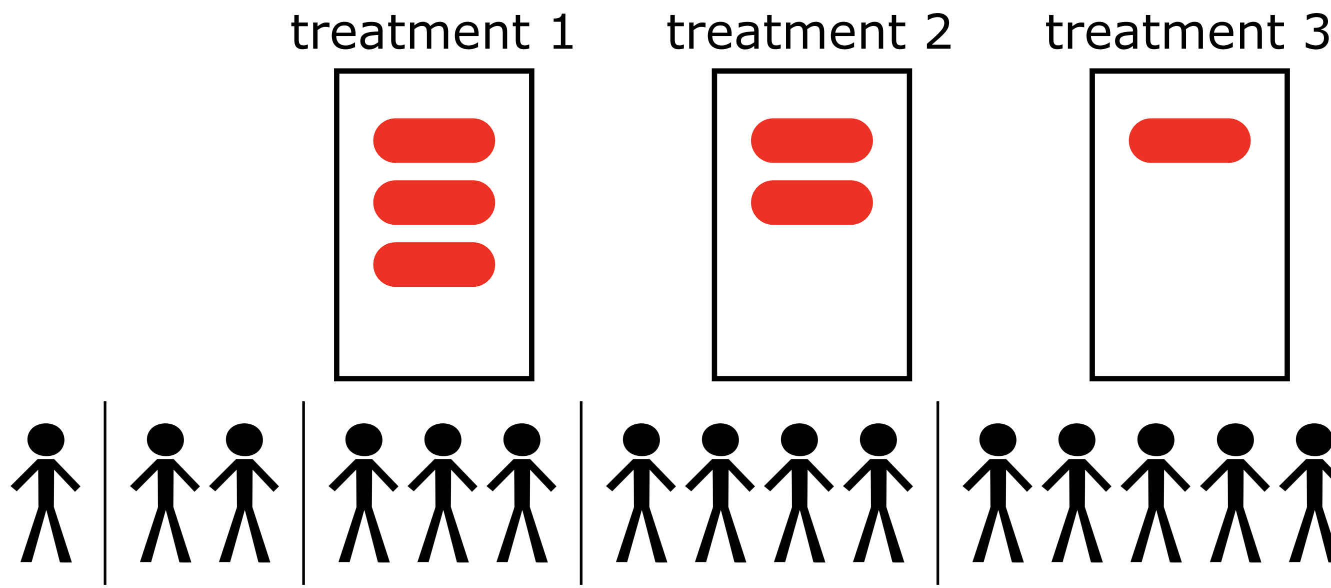
To figure this out, we can arrange the people and the treatments in the right order — by flinging the people around into a line that can file through the treatments.



day 1	15			**15**
day 2	12	10		**22**
day 3				
day 4				
day 5				
day 6				
day 7				

Convolution example from betterexplained.com

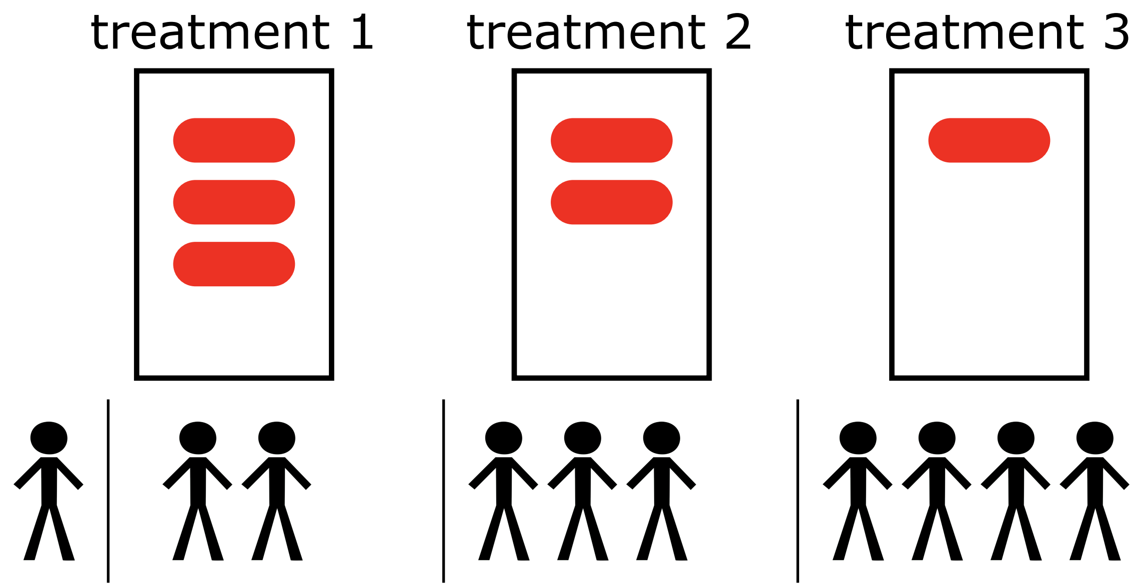
To figure this out, we can arrange the people and the treatments in the right order — by flinging the people around into a line that can file through the treatments.



day 1	15			**15**
day 2	12	10		**22**
day 3	9	8	5	**22**
day 4				
day 5				
day 6				
day 7				

Convolution example from betterexplained.com

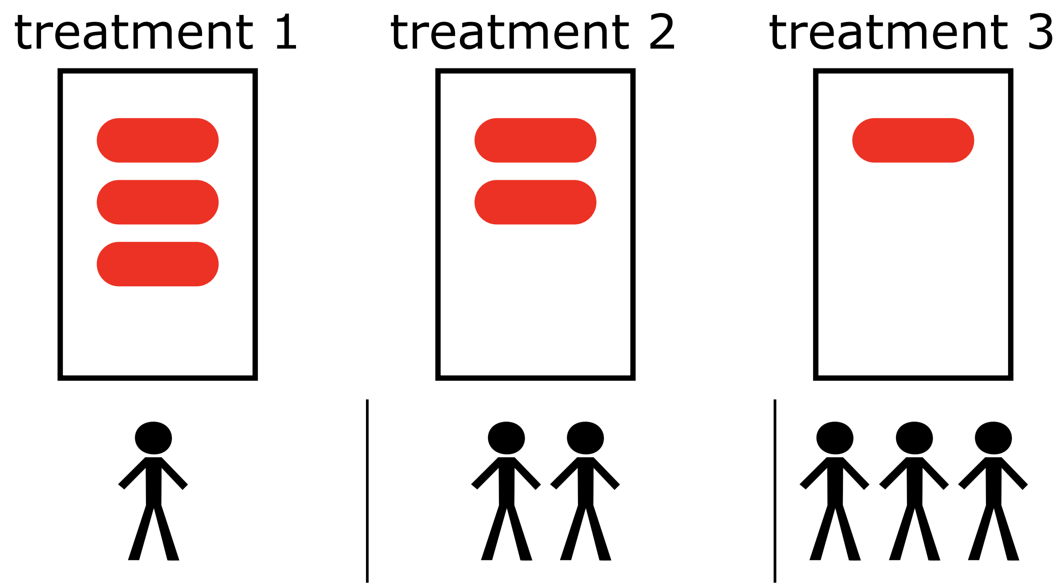
To figure this out, we can arrange the people and the treatments in the right order — by flinging the people around into a line that can file through the treatments.



day 1	15			**15**
day 2	12	10		**22**
day 3	9	8	5	**22**
day 4	6	6	4	**16**
day 5				
day 6				
day 7				

Convolution example from betterexplained.com

To figure this out, we can arrange the people and the treatments in the right order — by flinging the people around into a line that can file through the treatments.

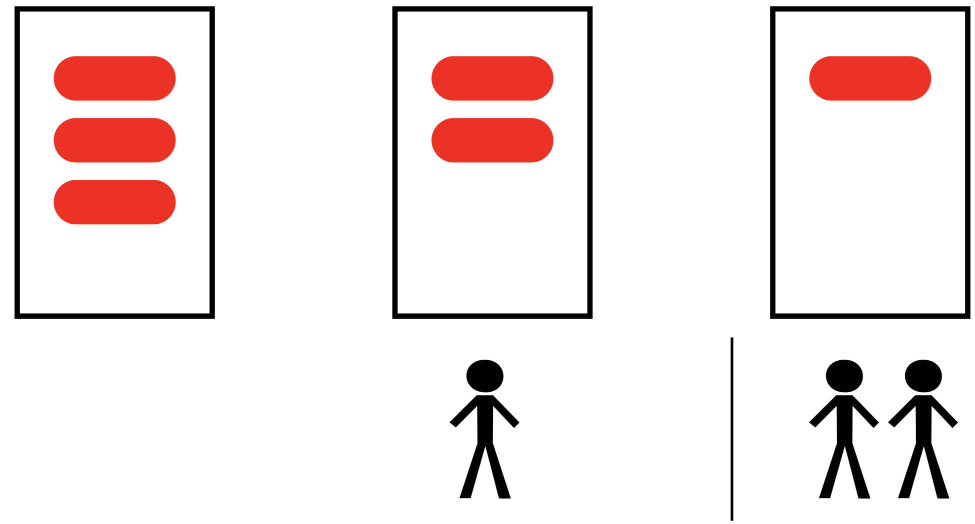


day 1	15			**15**
day 2	12	10		**22**
day 3	9	8	5	**22**
day 4	6	6	4	**16**
day 5	3	4	3	**10**
day 6				
day 7				

Convolution example from betterexplained.com

To figure this out, we can arrange the people and the treatments in the right order — by flinging the people around into a line that can file through the treatments.

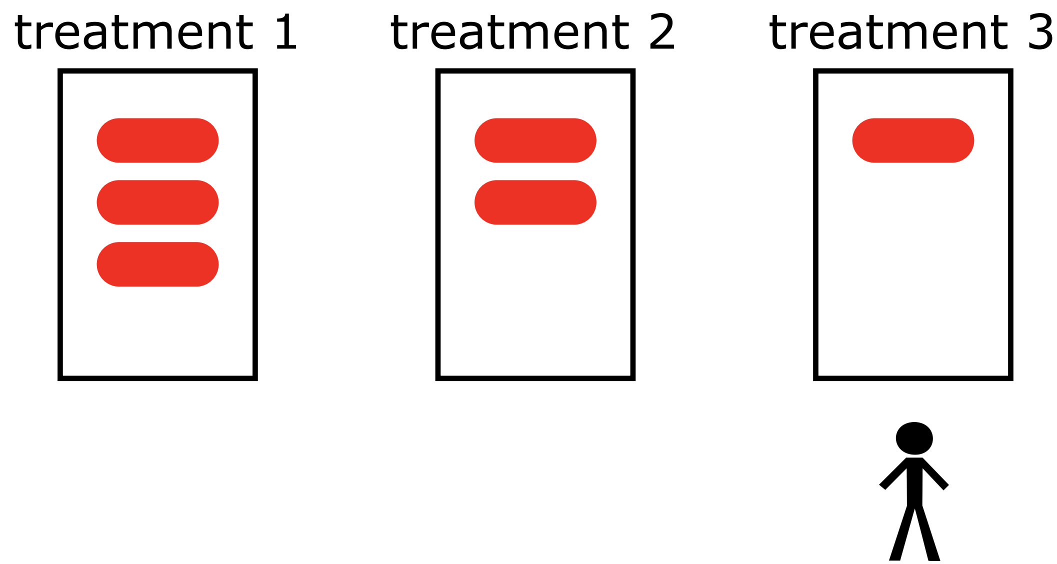
treatment 1 treatment 2 treatment 3



day 1	15			**15**
day 2	12	10		**22**
day 3	9	8	5	**22**
day 4	6	6	4	**16**
day 5	3	4	3	**10**
day 6		2	2	**4**
day 7				

Convolution example from betterexplained.com

To figure this out, we can arrange the people and the treatments in the right order — by flinging the people around into a line that can file through the treatments.



total = 90

day 1	15			**15**
day 2	12	10		**22**
day 3	9	8	5	**22**
day 4	6	6	4	**16**
day 5	3	4	3	**10**
day 6		2	2	**4**
day 7			1	**1**

Euler’s formula for complex numbers

Imaginary and complex numbers

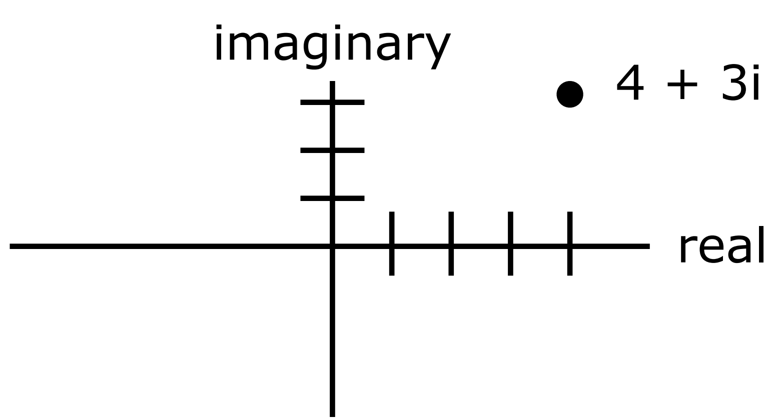
An **imaginary number** is the square root of a negative number. We define the unit **i** as the square root of -1, which allows us to calculate others:

$\sqrt{-1} = i$ $\sqrt{-9} = 3i$ $\sqrt{-25} = 5i$

A **complex number** is a combination of a real number and an imaginary number, like this:

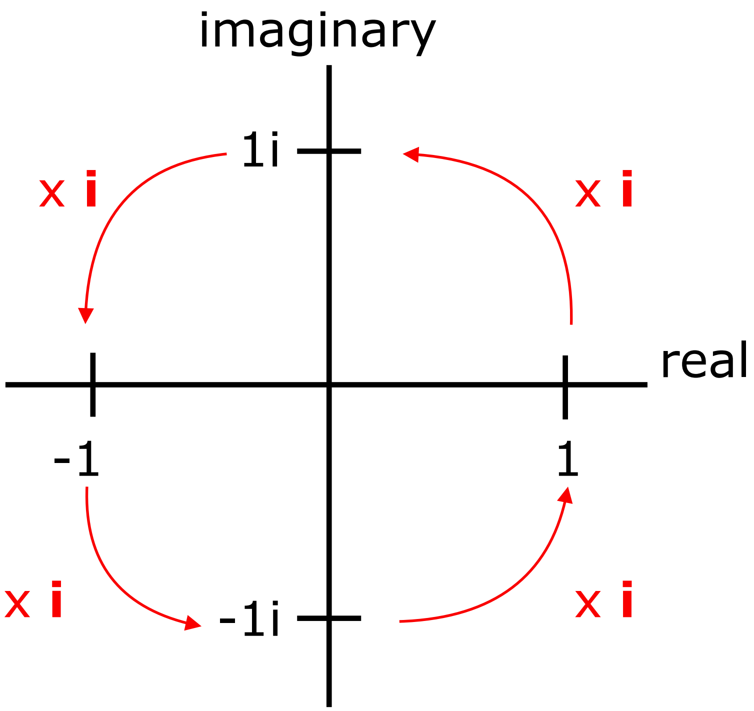
$3 + 4i$ $2 - 5i$ $4 + 2i$

Complex numbers can be used to define a coordinate system called the **complex plane**: the x-axis is the real component and the y-axis is the imaginary component:



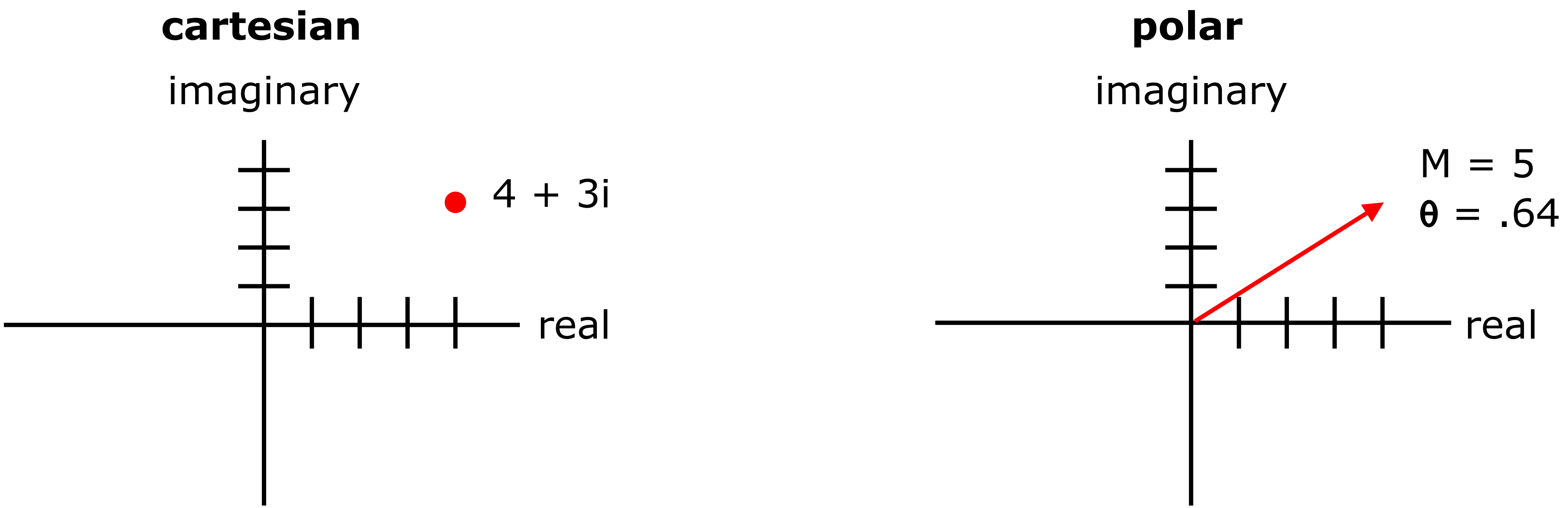
The big insight of the complex plane

The big insight of the complex plane is that one step **i** moves us off of the real axis onto the imaginary axis, and then another step **i** moves us back onto the real axis (-1).



Cartesian and Polar coordinates

Complex numbers of the form **a + bi** create a **cartesian coordinate system**. But there is another system called the **polar coordinate system** that is more useful for us.

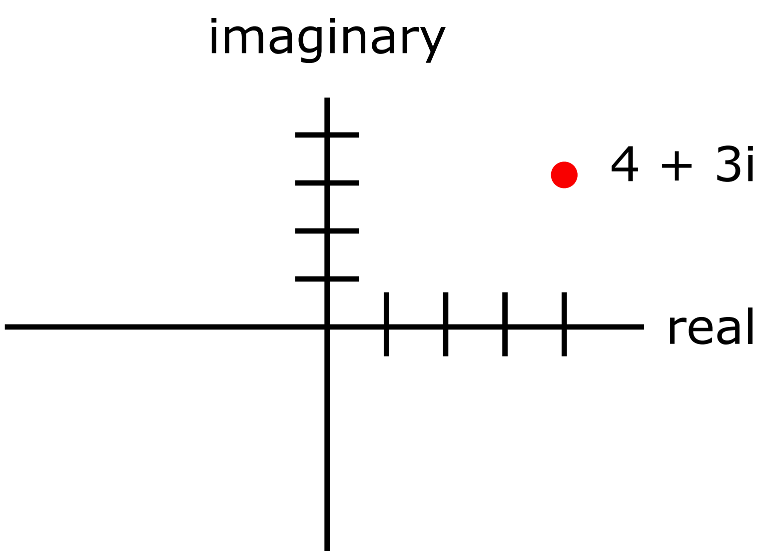


The polar coordinate system defines a point based on the magnitude of the vector that is defined by that point (**M**), and the angle of that vector relative to the real axis (**θ**).

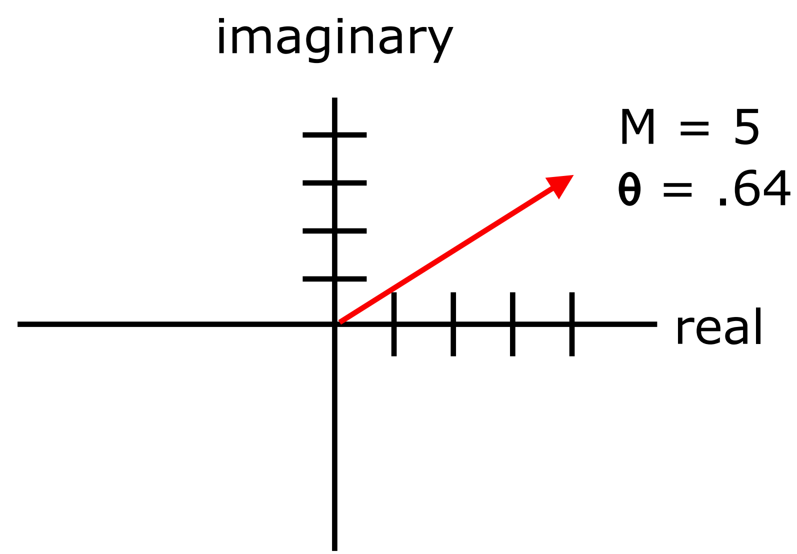
Cartesian and Polar coordinates

We can convert between the two systems with some concepts from trigonometry. It is worth taking a moment to see the equivalence.

cartesian



polar



real $= M\cos(\theta)$

imaginary $= M\sin(\theta)$

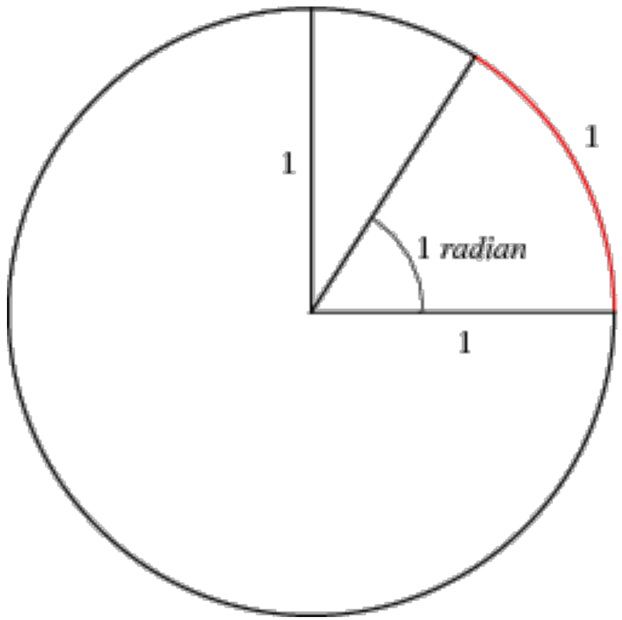
$$M = \sqrt{(\text{real}^2 + \text{imaginary}^2)}$$

$$\theta = \text{arctangent}(\text{imaginary}/\text{real})$$

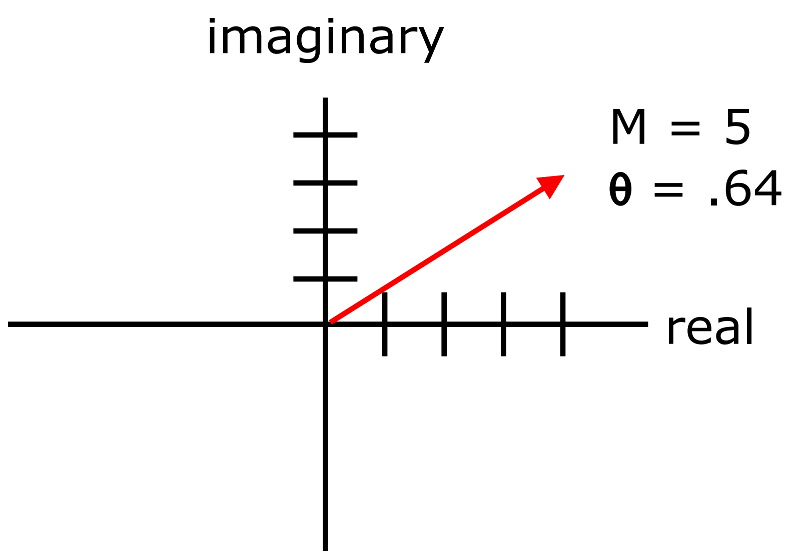
$$a + ib = M\cos(\theta) + i\,M\sin(\theta) = M[\cos(\theta) + i\sin(\theta)]$$

Angles are in radians

Remember that the angles used in polar coordinates are **radians.** Radians are about the length of an arc along the circumference swept by the angle — a 1 radian angle sweeps an arc equal to the radius of the circle. The circumference of a circle is **2πr**, so there are **2π** radians in a circle.



polar



Radians are technically pure numbers, with no unit after them. This is because they are arc length / radius length, so length cancels out.

$a + ib = M\cos(\theta) + i\,M\sin(\theta) = M[\cos(\theta) + i\sin(\theta)]$

Euler's formula for complex numbers

Euler discovered a concise way to represent complex numbers: $\mathbf{Me^{i\theta}}$

To make life simpler, let's assume that M = 1, so that it is just $\mathbf{e^{i\theta}}$ and so that the cartesian form that we want to see is **cos(θ) + i sin(θ)**. Let's see if we can prove that these are equivalent.

Taylor series:

$$e^x = \frac{X^0}{0!} + \frac{X^1}{1!} + \frac{X^2}{2!} + \frac{X^3}{3!} + \frac{X^4}{4!} + \frac{X^5}{5!} + \frac{X^6}{6!} \cdots$$

Taylor series with i:

$$e^{ix} = \frac{(iX)^0}{0!} + \frac{(iX)^1}{1!} + \frac{(iX)^2}{2!} + \frac{(iX)^3}{3!} + \frac{(iX)^4}{4!} + \frac{(iX)^5}{5!} + \frac{(iX)^6}{6!} \cdots$$

$$e^{ix} = 1 + iX - \frac{X^2}{2!} - \frac{iX^3}{3!} + \frac{X^4}{4!} + \frac{iX^5}{5!} - \frac{X^6}{6!} \cdots$$

The **even exponents** reduce the **i** component to become -1.
The **odd exponents** reduce the **i** component to i.

Euler's formula for complex numbers

Taylor series with i:

$$e^{ix} = 1 + iX - \frac{X^2}{2!} - \frac{iX^3}{3!} + \frac{X^4}{4!} + \frac{iX^5}{5!} - \frac{X^6}{6!} \cdots$$

Next, we group the real numbers and the imaginary numbers separately:

$$e^{ix} = \left(1 - \frac{X^2}{2!} + \frac{X^4}{4!} - \frac{X^6}{6!} \cdots\right) + \left(iX - \frac{iX^3}{3!} + \frac{iX^5}{5!} - \frac{iX^7}{7!} \cdots\right)$$

$$\cos X = \left(1 - \frac{X^2}{2!} + \frac{X^4}{4!} - \frac{X^6}{6!} \cdots\right)$$

$$\sin X = \left(X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} \cdots\right)$$

These are just two known facts. They are called the Taylor series for cos and sin.

$$e^{ix} = \cos X + i \sin X$$

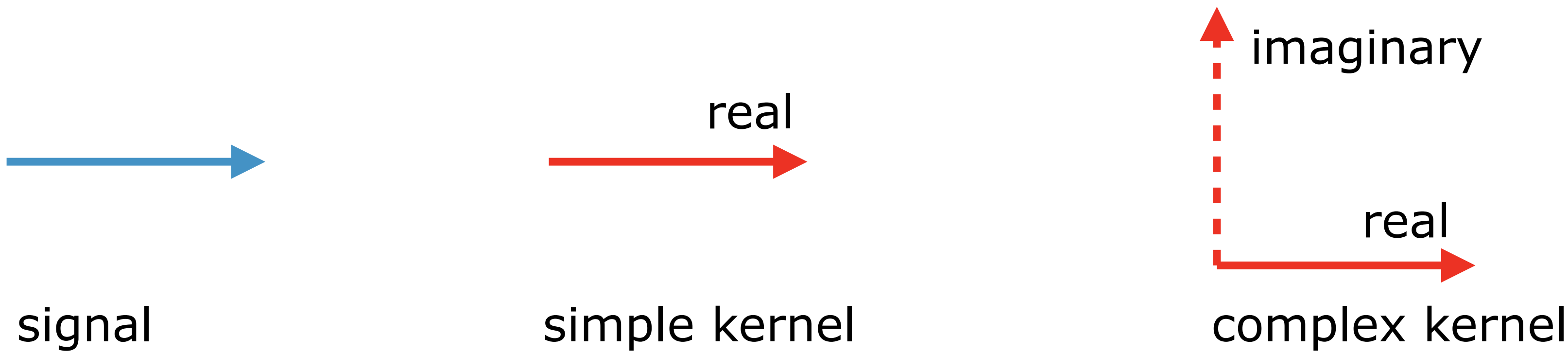
And this is what we wanted to prove. Euler's formula is equivalent to standard cartesian formula.

Why complex wavelets?

dot product with simple and complex numbers

Ultimately we want to look at the dot product between a sine wave (our EEG signal) and a complex wavelet (a wavelet with both a real and imaginary component).

But to simplify things, let's first look at simple two-dimensional vectors to see how the simple dot product and complex dot product work.



The crucial thing to remember here is that we will ultimately want to **keep the magnitude and the angle separate** in our results. This is because **magnitude represents EEG power**, and **angle represents phase**.

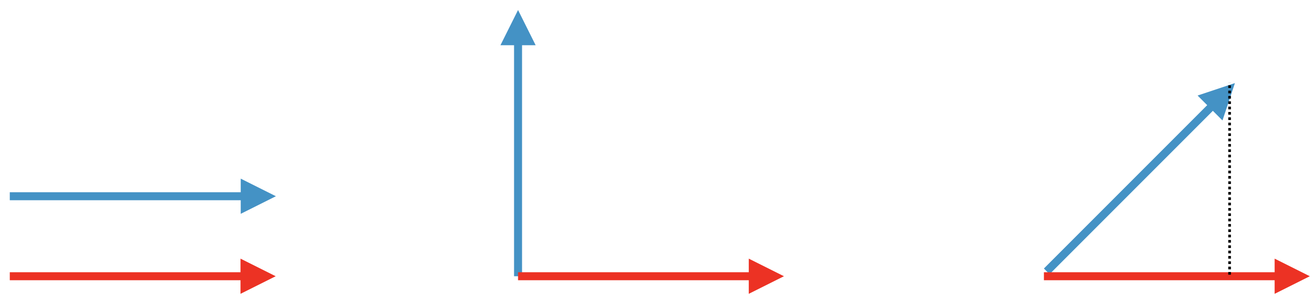
But for now I just want to show you that the simple kernel combines magnitude and angle into a single number, while the complex kernel keeps them separate… which is what we will ultimately want.

dot product with simple kernels

The dot product with the simple kernel, as the phase (angle) of the signal changes. All lengths are 1 to keep things simple.

signal

simple kernel



phase difference	0	$\pi/2$	$\pi/4$
dot product	1	0	.71

What we see is that the dot product combines angle similarity and magnitude similarity together into a single number. We already knew this from last time.

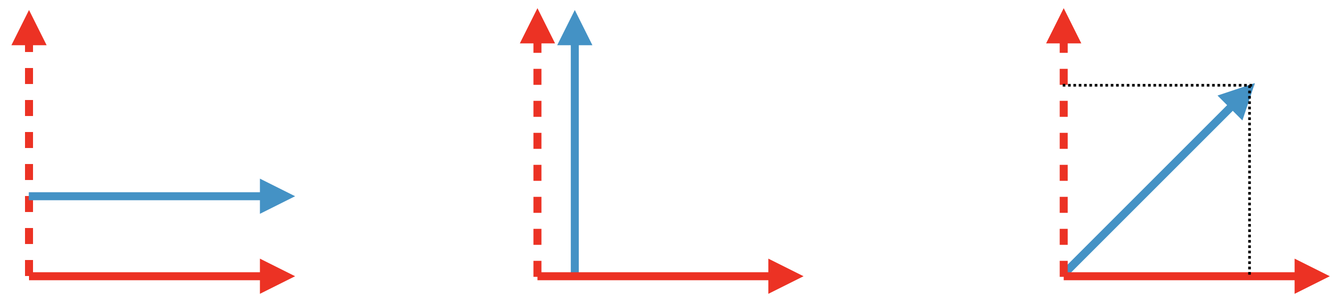
Now, let’s look at complex kernels to see what happens there.

dot product with complex kernels

The dot product with the simple kernel, as the phase (angle) of the signal changes. All lengths are 1 to keep things simple.

signal

complex kernel



phase (real)	0	$\pi/2$	$\pi/4$
dot (real)	1	0	.71
phase (imag)	$\pi/2$	0	$\pi/4$
dot (imag)	0	1	.71
complex dot	1 + 0i	0 + 1i	.71 + .71i

$$M = \sqrt{(\text{real}^2 + \text{imaginary}^2)}$$

$$\theta = \text{arctangent}(\text{imaginary}/\text{real})$$

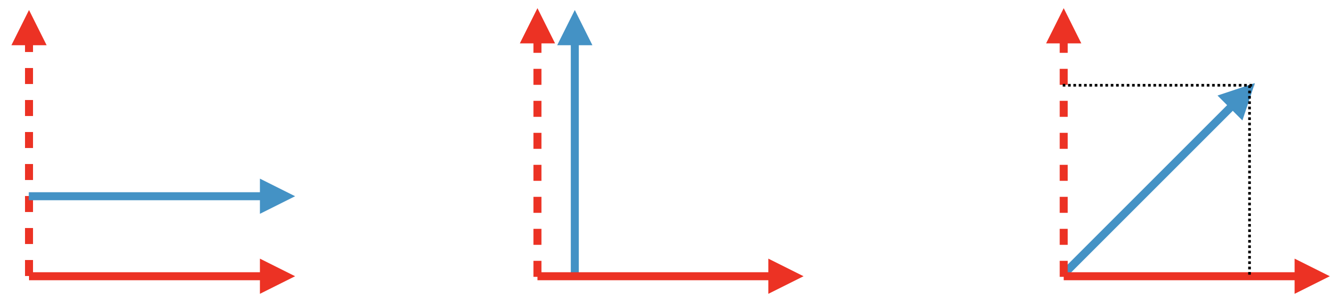
polar dot	$M=1\ \theta=0$	$M=1\ \theta=\pi/2$	$M=1\ \theta=\pi/4$

dot product with complex kernels

The dot product with the simple kernel, as the phase (angle) of the signal changes. All lengths are 1 to keep things simple.

signal

complex kernel



phase (real)	0	$\pi/2$	$\pi/4$
dot (real)	1	0	.71
phase (imag)	$\pi/2$	0	$\pi/4$
dot (imag)	0	1	.71
complex dot	1 + 0i	0 + 1i	.71 + .71i

The polar notation captures the magnitude similarity in the M component, and the phase offset of the signal and the real component in the theta component.

polar dot	M=1 $\theta=0$	M=1 $\theta=\pi/2$	M=1 $\theta=\pi/4$

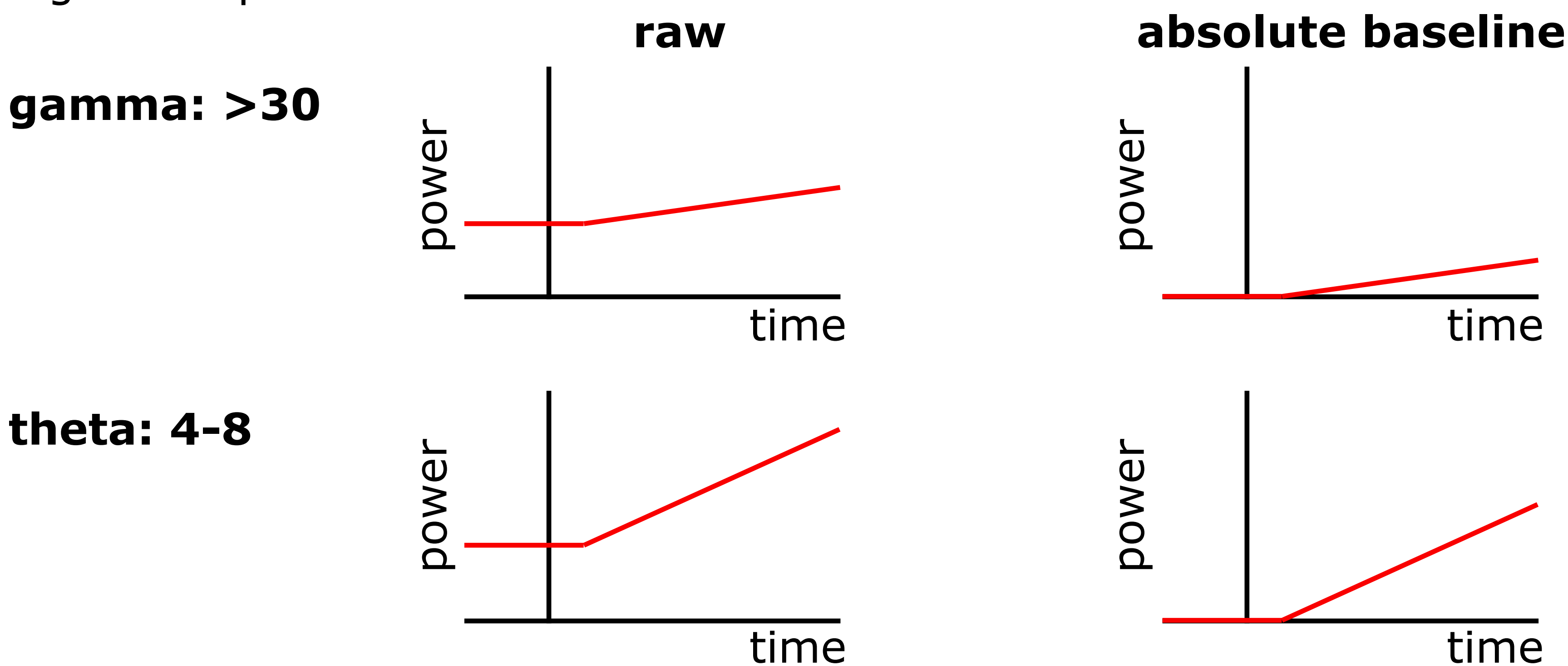
Baselines

Why not use the absolute baseline?

For ERPs, we used an **absolute baseline:** we calculated the mean activity in the baseline window, and subtracted this from each data point in the epoch. This is a type of mean centering.

The absolute baseline allows us to see changes from (approximately) 0.

The problem with the absolute baseline for time-frequency analysis is that the different frequency bands have different relative power. This is just a fact of EEG (and of frequencies in general): lower frequencies have more power than higher frequencies.



Logarithms

I guess logarithms answer the question "What exponent would we need to convert one number (the base) into another (the argument)?"

Some examples:

$\log_{10}(1000) = 3$

$\log_{10}(100) = 2$

$\log_{10}(10) = 1$

$\log_{10}(1) = 0$

$\log_{10}(0)$ = undefined

$\log_{10}(.1) = -1$

$\log_{10}(.01) = -2$

$\log_{10}(.001) = -3$

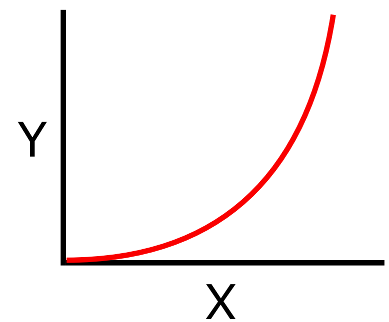
This is called a semi-log plot. The point is that logs make large numbers much smaller.

The general form is:

$\log_B(X) = Y$ $\quad\quad$ $B^Y = X$

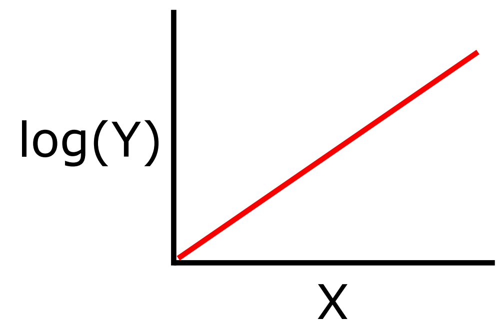
Visualizing the effect of logs:

$Y=10^X$



X	**Y**
1	10
2	100
3	1000

$\log(Y) = X \log(10)$



X	**log(Y)**
1	1
2	2
3	3

Small aside - logarithms and arithmetic

One useful function of logarithms is to do arithmetic in exponential (or power-law) space. We aren't doing that here, but I want you to see it.

Some examples:

$\log_{10}(1000) = 3$

$\log_{10}(100) = 2$

$\log_{10}(10) = 1$

$\log_{10}(1) = 0$

$\log_{10}(0) = \text{undefined}$

$\log_{10}(.1) = -1$

$\log_{10}(.01) = -2$

$\log_{10}(.001) = -3$

Addition in logs is multiplication in raw numbers:

$1000 \times 100 = 100{,}000$

$\log_{10}(1000) + \log_{10}(100) = \log_{10}(100{,}000)$

$3 + 2 = 5$

Subtraction in logs is division in raw numbers:

$1000 / 100 = 10$

$\log_{10}(1000) - \log_{10}(100) = \log_{10}(10)$

$3 - 2 = 1$

Bels and Decibels

The **bel** is named for Alexander Graham Bell. It is a measure of relative power between two signals. Crucially, it is the log of the ratio of the power of the two signals:

Bel: $\log_{10}(\text{power1}/\text{power2})$ — The common/decadic logarithm of the ratio of the power of two signals.

Bels are typically multiplied by 10 to form **decibels** in most human engineering systems, to make the numbers a bit larger (no decimals).

Decibel: $10 \times \log_{10}(\text{power1}/\text{power2})$

power1	power2	ratio	$\log_{10}$(power1/power2)	bel	decibel
10	10	1x	$\log_{10}(10/10)$	0	0
20	10	2x	$\log_{10}(20/10)$	.3	3
50	10	5x	$\log_{10}(50/10)$	.7	7
100	10	10x	$\log_{10}(100/10)$	1	10
1000	10	100x	$\log_{10}(1000/10)$	2	20

Small aside - sound intensity is dB

The most common use of the term **decibel** in daily life is for the measurement of sound intensity.

But we just saw that decibel is a measure of **relative power**.

So what is the second signal in the measure of sound intensity?

It is the reference intensity, postulated as a threshold for human hearing at a frequency of 1000Hz.

reference: 1×10^{-12} watts/m^2

decibel change from baseline

The decibel gives us our second baseline option (after the absolute baseline): the decibel change from the baseline period.

$$db_{tf} = 10 \times \log_{10}\left(\frac{activity_{tf}}{\overline{baseline_f}}\right)$$

Take the mean of the baseline period, calculate the ratio of the activity and this baseline mean, then take the logarithm, then multiply by 10.

ratio/division: The fact that it is a ratio (division) means that it is the activity relative to the baseline. If it were the pure ratio, a ratio of 1 would mean identity, 2 would mean 2x as big, etc.

logarithm: The fact that it is a logarithm means that growth is dampened — large numbers will have less of an impact. This helps correct for the inherent difference in power between frequency bands. It also means that a ratio of 1 is 0, so 0 becomes the identity value.

multiplication? We multiply by 10 to make the scale a little easier to work with. 0 is still the identify value.

percentage change from baseline

Our third baseline option is percentage change: how much does the activity differ from the baseline in terms of a percentage (or proportion) of the baseline:

$$\text{percentage}_{tf} = 100 \times \frac{\text{activity}_{tf} - \overline{\text{baseline}_f}}{\overline{\text{baseline}_f}}$$

subtraction: The subtraction is a type of mean centering. It transforms the activity into a change from the baseline

ratio/division: The division transforms the change from the baseline into a proportion based on the baseline activity.

percentage This just turns the proportion into larger numbers, potentially eliminating decimals.

no logarithm: There is no logarithm, so the scale is linear. This is a weaker correction for inherent power differences.

When to baseline

With ERPs, we use the absolute baseline (subtraction). This is a linear transformation, so we can apply it at any stage of the analysis.

trial	pre	post
1	5	7
2	4	6
3	3	5

baseline →

trial	pre	post
1	0	2
2	0	2
3	0	2

average →

pre	post
0	2

trial	pre	post
1	5	7
2	4	6
3	3	5

average →

pre	post
4	6

baseline →

pre	post
0	2

So, we have never had to think about when the right time to baseline is. **But, crucially, these are different numbers - the average of baselined trials, versus a baselined average.**

When to baseline

Let’s try it with decibel change. They are not perfectly identical. This is partly due to division, and partly due to the logarithm. But which is correct?

trial	pre	post
1	5	7
2	4	6
3	3	5

baseline →

trial	pre	post
1	0	1.46
2	0	1.76
3	0	2.22

average →

pre	post
0	1.81

trial	pre	post
1	5	7
2	4	6
3	3	5

average →

pre	post
4	6

baseline →

pre	post
0	1.76

In general, it is best to average first, then baseline. But this is not due to any deep principle. It is just because TF results vary more than ERPs across trials.

When to baseline

Let's try it with percent change. They are not perfectly identical. This is due to division. But which is correct?

trial	pre	post
1	5	7
2	4	6
3	3	5

baseline →

trial	pre	post
1	0	40
2	0	50
3	0	67

average →

pre	post
0	52.3

trial	pre	post
1	5	7
2	4	6
3	3	5

average →

pre	post
4	6

baseline →

pre	post
0	50

In general, it is best to average first, then baseline. But this is not due to any deep principle. It is just because TF results vary more than ERPs across trials.