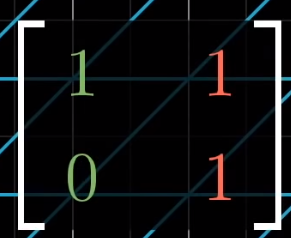
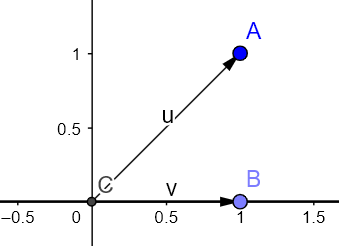
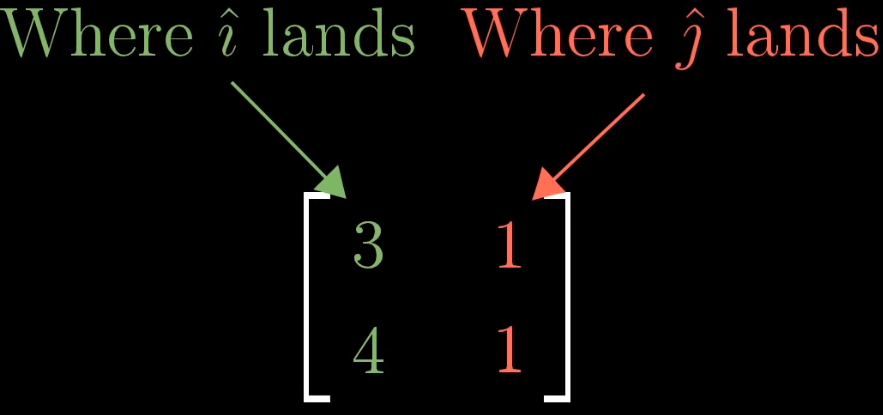
**Basics of Linear Transforms**

Linear transform: All grid lines are parallel and evenly spaced after a transform

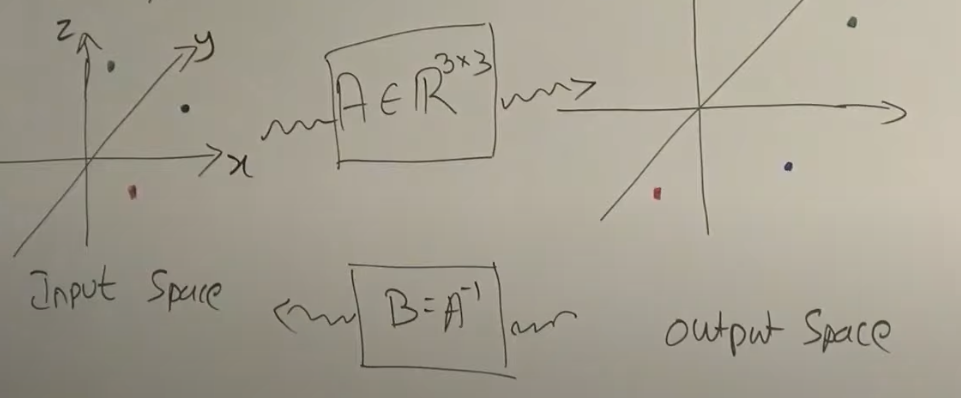
Column vectors: The new location of the basis vectors

Rank. Means the number of dimensions in the output. Also called the column space because it’s where the basis vectors land. The span of the column = the column space.

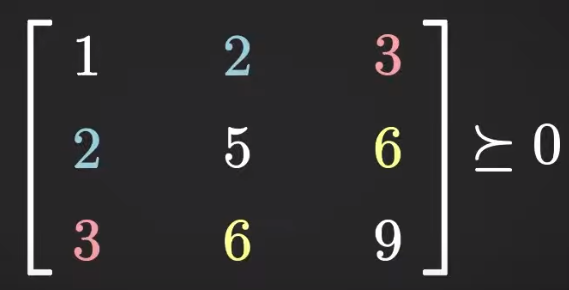


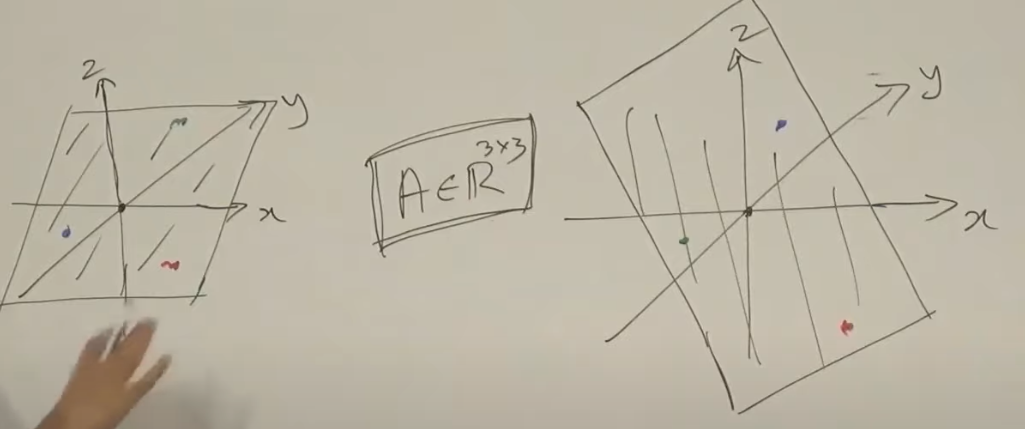
Full rank, unique 1:1 mapping between input and output space



Rank deficient there exists a lower dimension subspace, which must pass through the origin, and a corresponding matrix in the output matrix that also passes through the origin, with a 1:1 mapping.

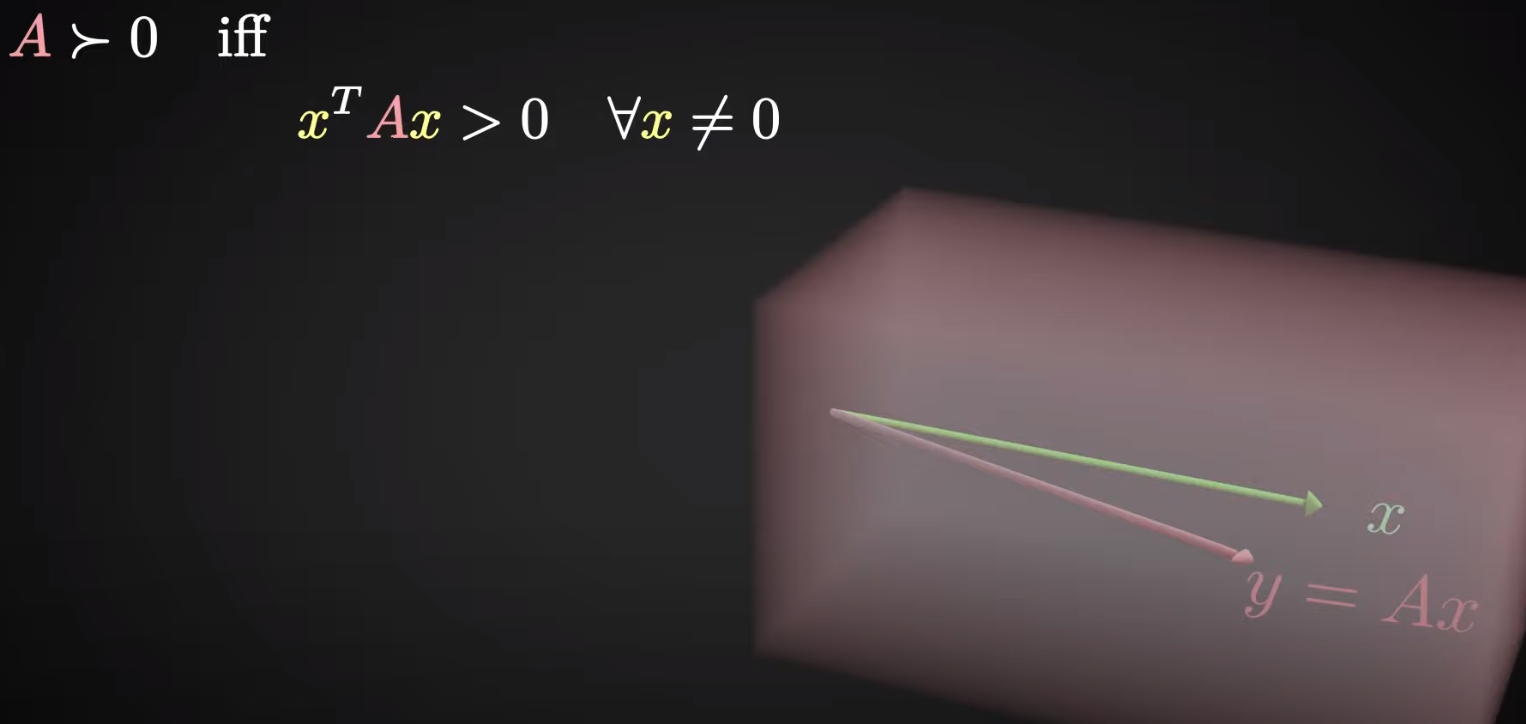
**Properties of Linear Transforms**

Symmetric. A = AT. Has important properties: their Eigen values are **real** and their eigenvectors are **orthogonal** 



Positive Definite. Analogous to finding if a number is greater > zero. But how can we declare if a matrix is ‘positive’?

Given y=Ax, a matrix A is PD if the dot product of y and x is > 0 for all x != 0. In other words, the transformed vector y is always on the same side as x. To check if PD, need to check the eiganvalues of a matrix.

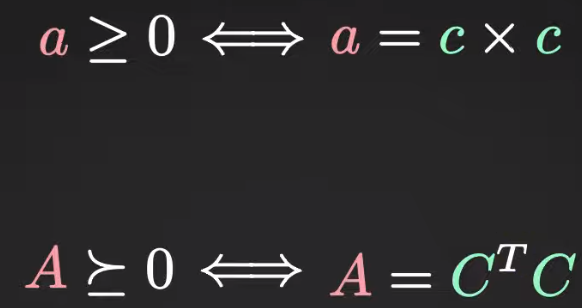


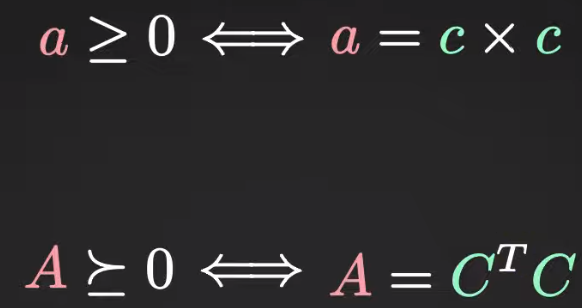
Positive Semi-Definite if the dot product can equal zero, then at least one transformed vector y exists that is orthogonal to the input vector x. Given y=Ax, a matrix A is PD if the dot product of y and x is >= 0 for all x != 0

PSD if eigenvalues >= 0. If eigen values are negative then the direction is reversed

PSD if xTAx >= 0

Funky properties of PSDs, they have a ‘square root’ C. Analogous to positive scalar numbers.

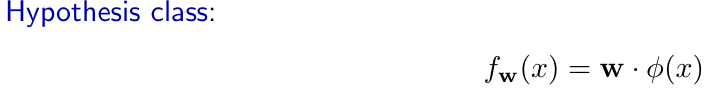




PSD programming was the hottest thing in mathematical programming in the 90s. It’s a special case of convex programming.

Lyapunov function. A linear dynamical system is **asymptotically stable** iff there exists a lyapunov function!

NP-Hard. No algo can solve efficiently.

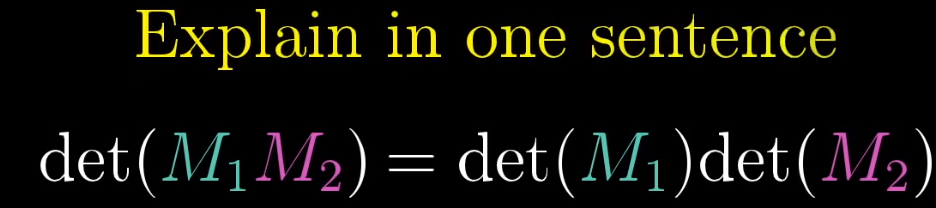
**Matrix trace** Sum of all elements in the diagonal of matrix. Only defined for square nxn matrix. It is used to [measure complexity](https://www.quora.com/What-is-a-trace-as-in-trace-of-a-matrix-and-why-is-it-used) of any linear machine learning model 

Determinant: measures how much a linear transform squishes or stretches a space. This tells us how *any* region in space will scale after the transform because all the grid space transformations are linear Determinants can be negative, which means the ‘orientation is flipped’.

For 2D transforms, the determinant tells us how a unit area scales.

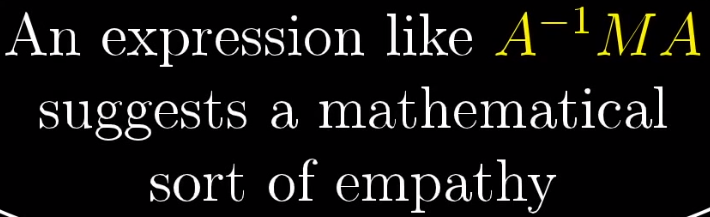
For 3D transforms: determinants tell us the scaling of the volume after a transformation. Negative determinants in 3D:

* determinant !=0 then there exists an inverse matrix.
* If determinant = 0 then no inverse matrix because the transform collapsed a dimension

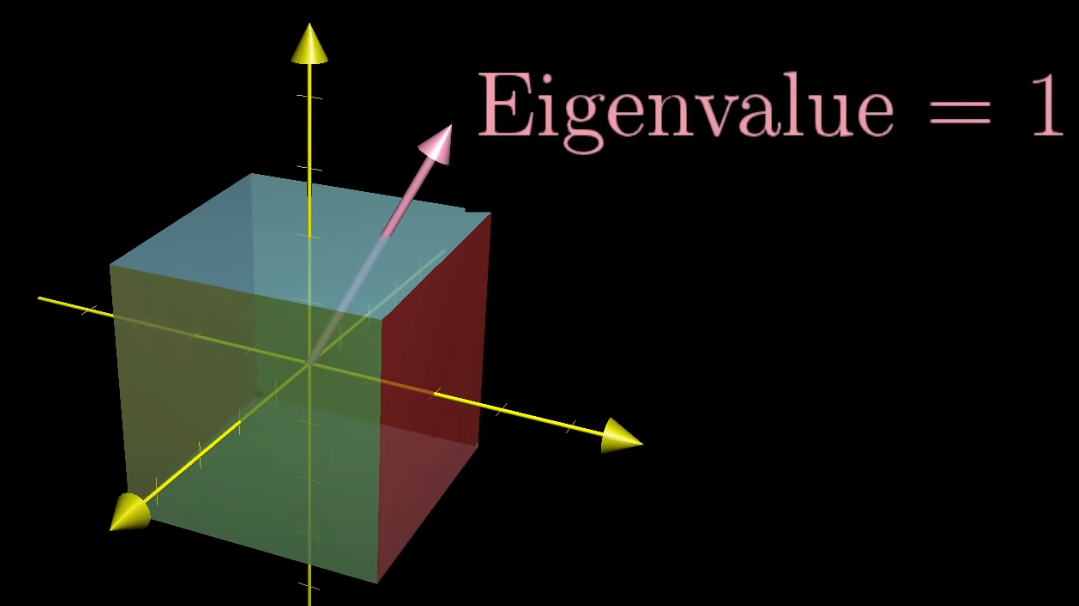


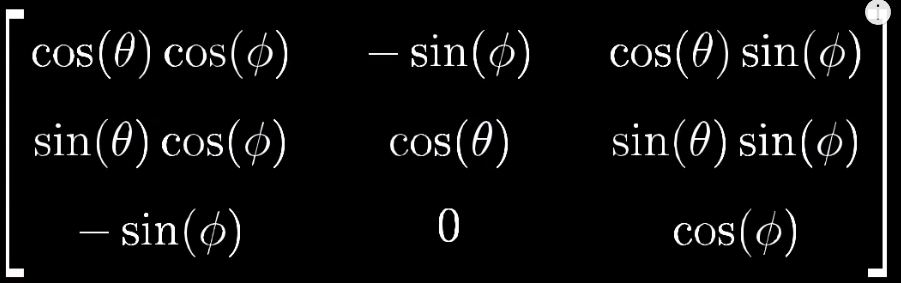
Null-space. The space of all vectors that land on zero-vector. Full rank matrix only has one solution: the origin, rank deficient matrix have lots of vectors that land on zero.

Inverse matrix: reverse a transformation to find the original vector.

Change of Basis vectors are the coordinates of the grid. M is a transform in our language, A is transform to the basis of Jennifer’s.

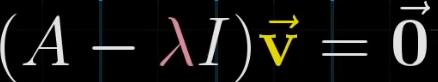
Eigen vectors. Most vectors get knocked off their span after a transformation. Those that stay on the same original span don’t and are called Eigen vectors. They are scaled by a scalar factor – the Eigen value.

Why it’s useful. Consider 3D rotation. If you find Eigen vector then you found the axis of its rotation. The corresponding eigenvalue is 1. Much easier to think about rotating about axis of rotation vs. its rotation matrix. 



The better way to get at the heart of a linear transform rather than thinking about its basis vectors is to think about its Eigen vectors.

Find Eigen vector by knowing after the linear transform, which squishes it into a lower dimension such that determinant = 0

 => 

Diagonal matrix all basis vectors are eigen vectors and all diagonals are the eigen values

**Operations**

Three types of multiplications

Vector-vector multiplication

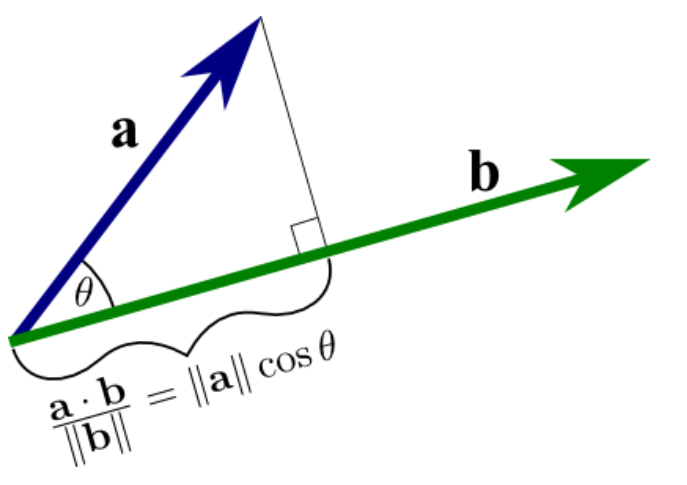
* Inner product = dot product. Result is scalar
* Outer product results in matrix. Why is it useful?

Matrix-vector multiplication

A (m x n) x(n x 1) = vector (m x1)

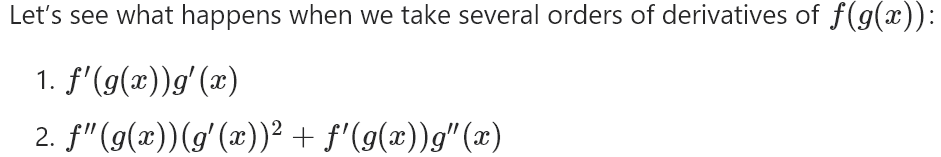
Matrix-matrix multiplication

A (m x n) x B (n x p) = C (m x p)

Projection is not the same as dot product. It’s the component of a vector.

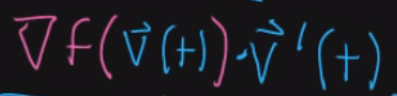
**Calculus**

Chain rule



**Chain rule in vector form[[1]](#footnote-1)**

Dot product between gradient of v and vector derivative v. Interpret as the directional derivative

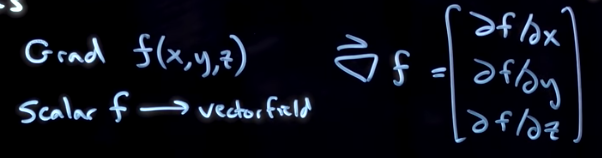
****

**Vector field** is the solution to a partial differential equation. The vector field also induces a new dynamical system, whereby if you dropped a particle into a vector field, it ‘sees’ the field dynamics, which you can use to predict its behaviour.

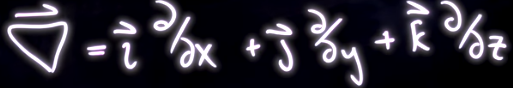
Div how much +div flowing away, -div is sucking in.

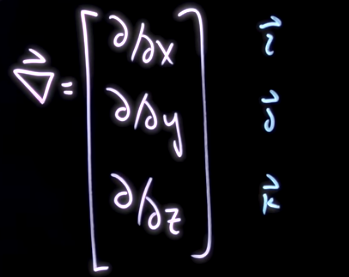
Curl +curl is spin.

**Gradient** (subset of a jacobian). Takes a scalar f and returns a vector field. There’s a gradient value for every point in space! Think of it as fluids! It tells you which direction the temperature is increasing locally the fastest. *“Follow the gradient direction to get to the place the fastest!”*

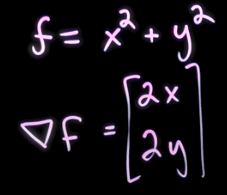
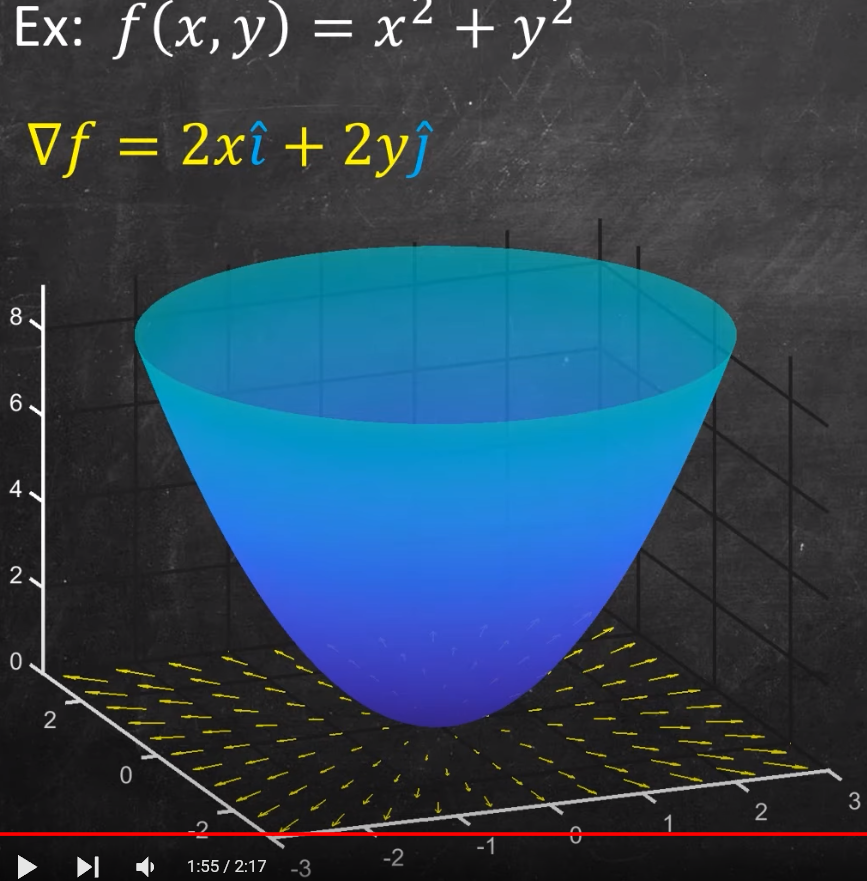


You can write grad in a few ways. Still the same thing

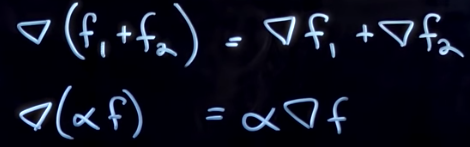




Example of gradient is the paraboloid. The gradient is a vector field. *f* in this case is the height of the paraboloid. Large gradient means its steep. Low gradient means its shallow. 0 gradient is no change in any direction.

 ****

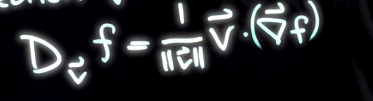
Grad is a linear operator. Steve Brunton says this property is very important because superposition holds.



We use the gradient to compute the directional derivative. The derivative of *f* in a direction *v*. We literally just take the dot product. If dot product is zero then the gradient isn’t changing.

****

**Typically should see it normalised because it doesn’t matter what the magnitude of v is. Just want to know the change ratio.**

****

When to use nabla? <https://www.youtube.com/watch?v=m2mW2FQJgEE>

Hessian

Positive Definite Hessian = Local Minimum

Negative Definite Hessian = Local Maximum

Indefinite Hessian = Saddle point

Other definiteness = Test is inconclusive

1. <https://www.youtube.com/watch?v=qZlBjnC3iro> [↑](#footnote-ref-1)