***Writing a transform for the Epilepsy Data.***

Writing a “transform”, that is a function applied to the data such that it creates a feature vector, is pretty simple within the Splinter Group Framework. The general philosophy is that any transform method (or function) accepts an epilepsy package and returns a feature vector in a list. In this way, a transform can be built by concatenating the results of multiple other transforms and we can build up a feature vector from multiple transform methods. This document spells out the process of generating a transform method in general.

Diagrammatically, this is how it works:

Returns:

Passed into:

**Transform Method**

Feature Vector

[as List]

**Data Package**

The transform method should accept one and only one argument, the package. Therefore, if there are some parameters in the transform (such as would be the case with a filter, where a frequency range would be defined), the transform function should be a function that calls another function, passing the package and those parameters.

The general form of the transform method is:

**def** prototypeTransform(EEGpackage):

#Do your maths.

return features[]

For example, if I am to define a function that returns the fft:

**def** fft(package):

packet = package.packet

axis = packet.ndim – 1

fftArray = np.fft.rfft(packet, axis=axis).

fftArrayRealPartOnly = fftArray.real

fftFeatureVector = fftArrayRealPartOnly.tolist()

return fftFeatureVector