TIMESERIES 3

11.5.2018

RECAP

* oscillations/periodic signals
  * often sinusoidal!
  * even when not sinusoidal, can be decomposed into a sum of sinusoids
    * this is the **fourier transform**

RECAP

* to see which frequencies are present in a timeseries, simple fourier transforms are not the best tool

* instead, we use **power spectral density (psd)** estimators

  * this is like a *regularized* fourier transform

RECAP

* if we compute psd for small snippets of time and then stack them together into an array

  * this is the **spectrogram**

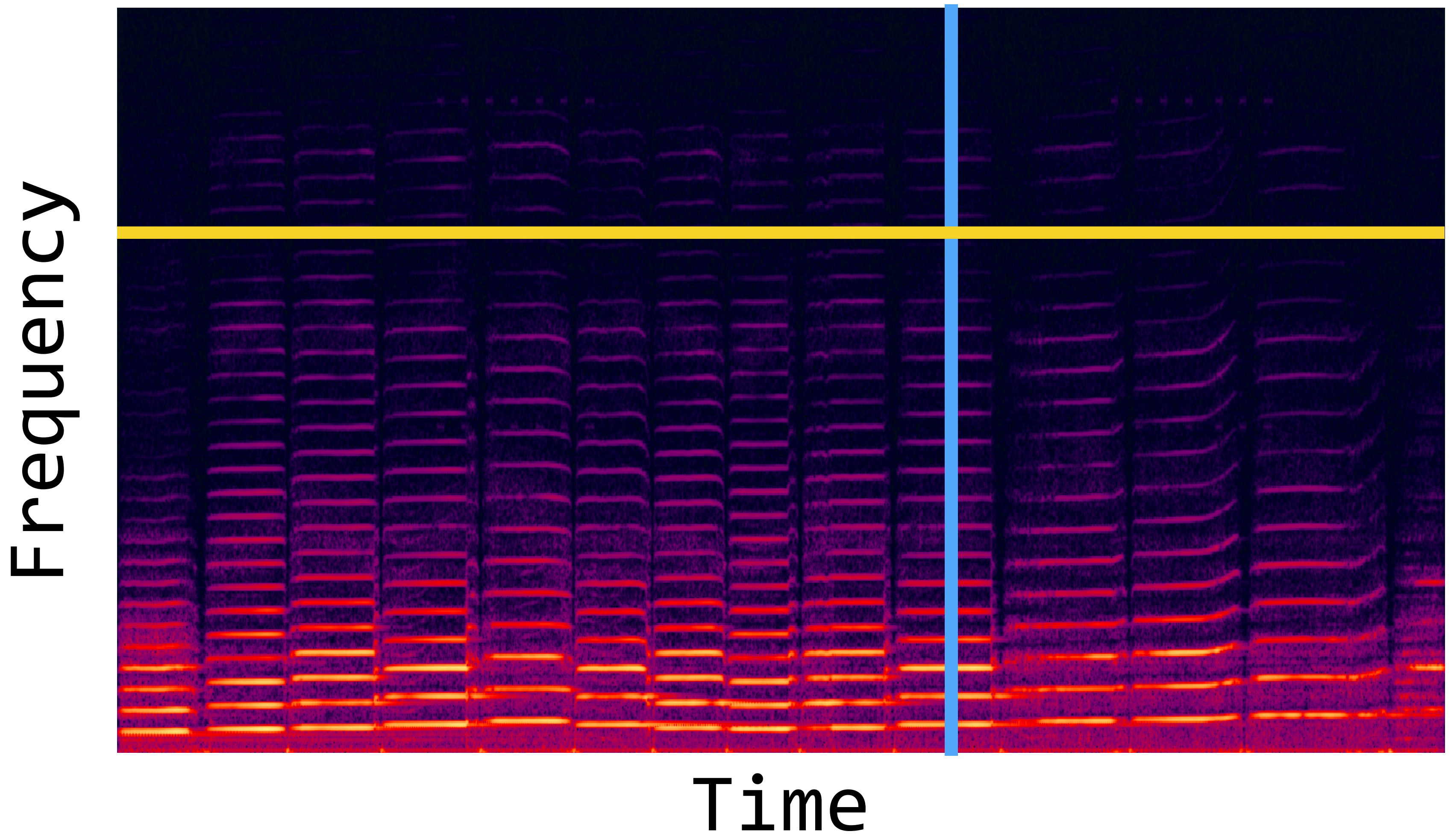
  * it shows which frequencies are present in a timeseries at each point in time

  * *you should know how to read a spectrogram*

THE SPECTROGRAM

* each column is the fourier transform of a short snippet

* what about each row? what does one row mean?



FILTERING

* **filtering** is a process that removes some frequencies from a timeseries and lets others remain (or even amplifies them)

* this is accomplished by convolving your timeseries with a **filter**, a small array that is designed to have a specific effect

FILTERING

* **low-pass filter:** removes high frequencies, allows low frequencies through

* **high-pass filter:** removes low frequencies, allows high frequencies through

* **band-pass filter:** removes all frequencies except for a specific band (the “pass band”)

FILTERING

* **low-pass filter:** removes high frequencies, allows low frequencies through

* **high-pass filter:** removes low frequencies, allows high frequencies through

* **band-pass filter:** removes all frequencies except for a specific band (the “pass band”)

FILTERING

* back to the spectrogram:

  * one row of a spectrogram is a lot like a **band-pass filtered** version of a timeseries

FILTERING

* suppose we have some EEG data from a human subject and we want to filter it so that only alpha-band oscillations remain

  * (this is a band-pass filter)

* how do you make a filter that has the properties you want?

FILTERING

* **scipy.signal** is a module in scipy that contains lots of useful functions for filter design

* **scipy.signal.firwin** creates “finite impulse response” filters with desired properties

ANALYZING A FILTER

* **scipy.signal.freqz** is a great function that tells you what the *frequency response* of your filter looks like

  * i.e. it tells you what the filter is going to do to your signal

FOURIER ANALYSIS

* fourier transforms have an interesting property related to convolution:

* given two timeseries, f and g, their convolution is equal to the element-wise product of their fourier transforms

$$f*g = F \cdot G$$

* the reverse is also true:

$$F*G = f \cdot g$$

FOURIER ANALYSIS

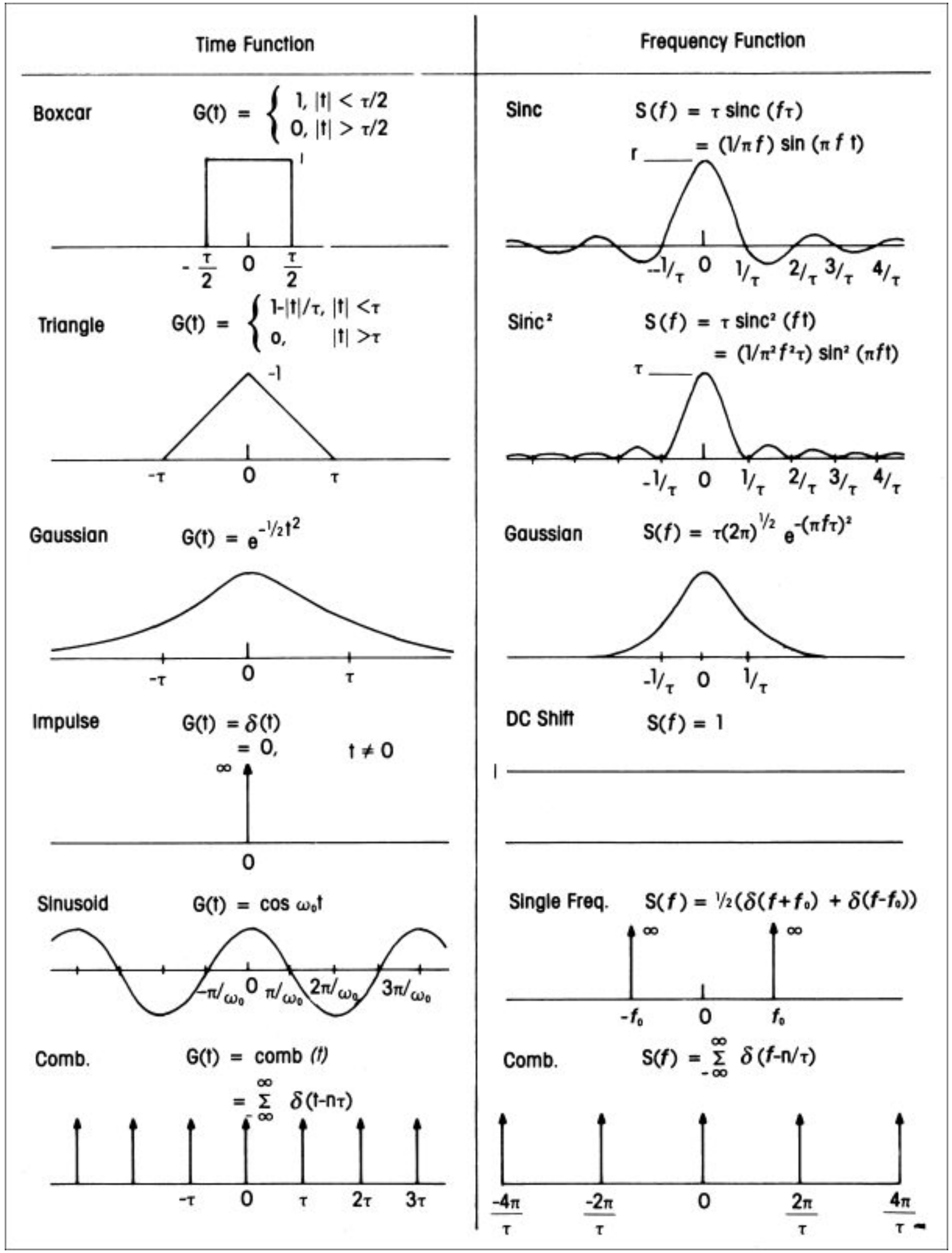
* this property is important, because convolution is expensive

* oftentimes it’s (much!) faster to
  (1) take the fourier transform of both,
  (2) take their element-wise product, and
  (3) take the inverse fourier transform

FOURIER ANALYSIS

* it also makes the effect of filtering much more intuitive

* filtering your timeseries X with a filter f is equivalent to taking the fourier transform of each and then (element-wise) multiplying them!


Time Function
Frequency Function
Boxcar
G(t) = 1, |t| < τ/2
0, |t| > τ/2
Sinc
S(f) = τ sinc (fτ)
= (1/πf) sin (π f t)
Triangle
G(t) = 1-|t|/τ, |t| <τ
0, |t| >τ
Sinc²
S(f) = τ sinc² (ft)
= (1/π²f²τ) sin² (πft)
Gaussian
G(t) = e^(-½t²)
Gaussian
S(f) = τ(2π)^½ e^(-(πfτ)²)
Impulse
G(t) = δ(t)
= 0, t ≠ 0
DC Shift
S(f) = 1
Sinusoid
G(t) = cos ω₀t
Single Freq.
S(f) = ½(δ(f+f₀) + δ(f-f₀))
Comb.
G(t) = comb (t)
= Σ δ(t-nτ)
Comb.
S(f) = Σ δ(f-n/τ)

NYQUIST FREQUENCY

- all of the timeseries we work with are *discrete* or *digital*, meaning that they are made up of **samples** separated by some even spacing in time
  - (note that **sample** is used in a different sense here than in statistics)

NYQUIST FREQUENCY

* the number of samples taken per unit time is called the **sampling rate**

    * e.g. in fMRI our sampling rate is typically 0.5 Hz (1 sample every 2 seconds)

    * in electrophysiology it could be as high as 25 kHz (25,000 samples per second)

NYQUIST FREQUENCY

* the sampling rate limits the frequencies that can be represented in a timeseries

* the highest frequency that a timeseries can represent is called the **Nyquist frequency**, and it is exactly half the sampling rate

Harry Nyquist

NYQUIST FREQUENCY

* for example if our fMRI data is sampled at 0.5 Hz, then the Nyquist frequency is 0.25 Hz

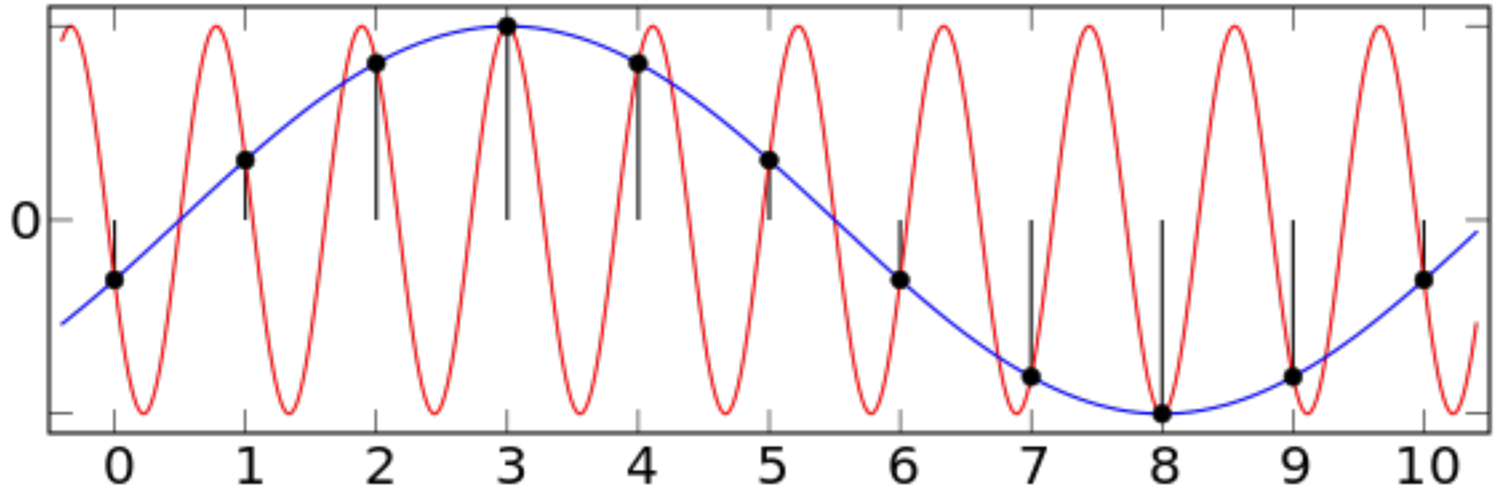
NYQUIST FREQUENCY

* why is this? why can’t higher frequency signals be represented?

NYQUIST FREQUENCY

* the problem is that any frequency above Nyquist would appear identical to some frequency below Nyquist

* this is called *aliasing*



END