MRC | Cognition and Brain Sciences Unit

Basics of Signal Analysis: Signals, Sampling, Noise

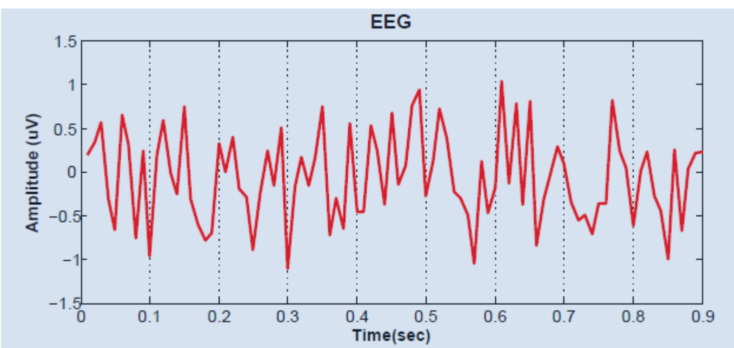
Alessandro Tomassini

MRC Cognition and Brain Sciences Unit
Alessandro.Tomassini@mrc-cbu.cam.ac.uk

***SIGNAL**: […]"**is a function that conveys information about the behaviour or attributes of some phenomenon**"[…]*

*In the physical world, **any quantity exhibiting variation in time or variation in space (such as an image) is potentially a signal** that might provide information on the status of a physical system, or convey a message between observers, among other possibilities. (Wikipedia)*

Time

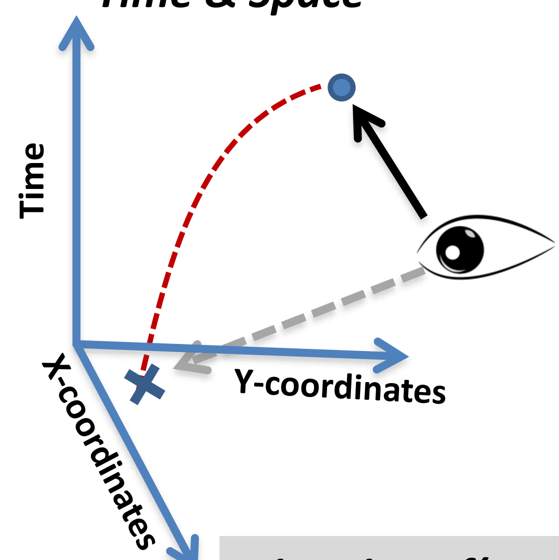


Amplitude = f(t)

Space

Luminance = f(x,y)

Time & Space



Direction= f(x,y,t)

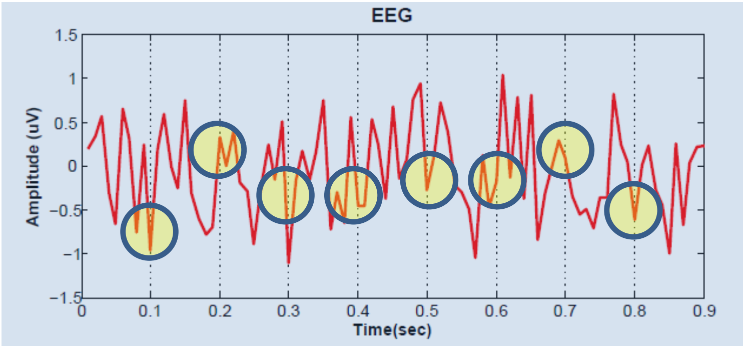
***Sampling**: is the conversion of a continuous signal (brain activation in time & space, 2D images etc) to a sequence of discrete sample (discretisation)*

Why does it matter? :

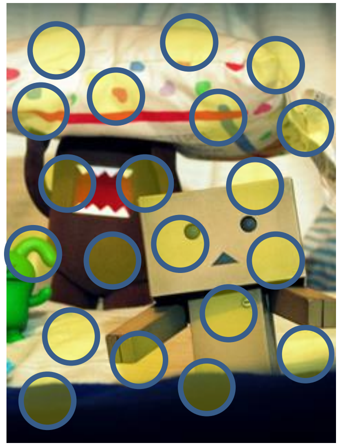
- Digital signal processing can only handle discrete numbers (finite precision)

-Sampling can provide the information necessary for the intended analysis while at the same time allow for efficient processing

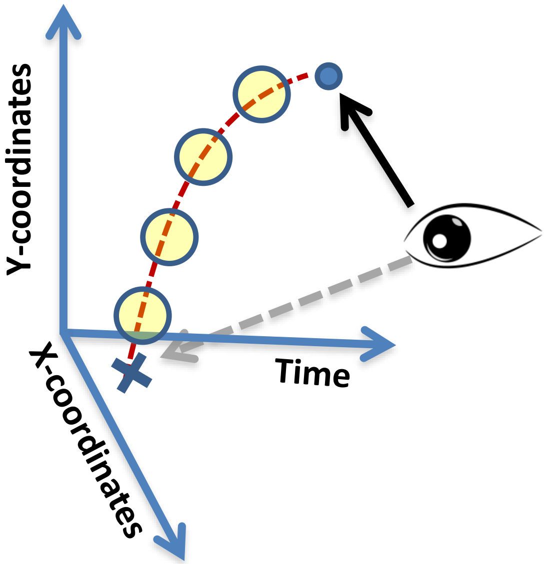
Time



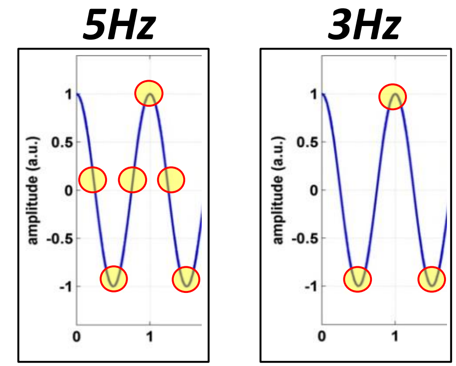
Space



Time & Space

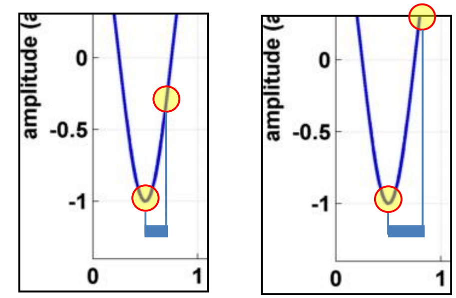


Basic Concepts

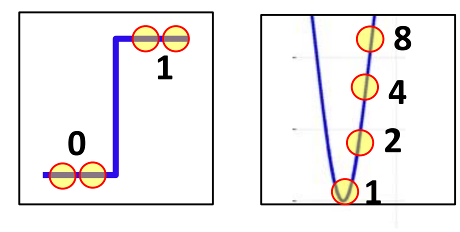


It is usually most convenient to sample **equidistantly**, i.e. neighbouring samples have the same distance to each no matter at what point of the sample they are

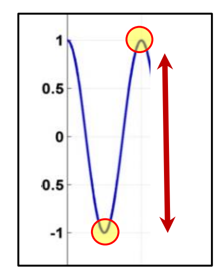
Sampling Rate/Frequency: How densely do we take samples? For example:
100 samples per second -> 100 samples/s -> 100 Hz
10 samples per centimetre -> 10 samples/cm
100 samples ("pixels") per square centimetre -> 100 samples/cm2



Sampling Interval/Distance: How far apart are the samples (in time, space etc.)?
*100 Hz -> (1/100)*1s = 0.01 s = 10 ms*
*10 samples/cm -> (1/10)*1 cm = 0.1 cm = 1mm*
*100 samples/cm2 = (1/100)*1 cm2 = 0.01 cm2 = (0.1*0.1) cm2 = 1 mm2*



Sampling depth (quantisation): For one particular sample, how many different values can we separate in digital representation?
"2 bit": Either 1 or 0, we can only separate 2 values (e.g. Black/White)
"8 bit": 1 2 4 8 16 32 64 128 => 256 different values [1/0 1/0 1/0 1/0 1/0 1/0 1/0 1/0]

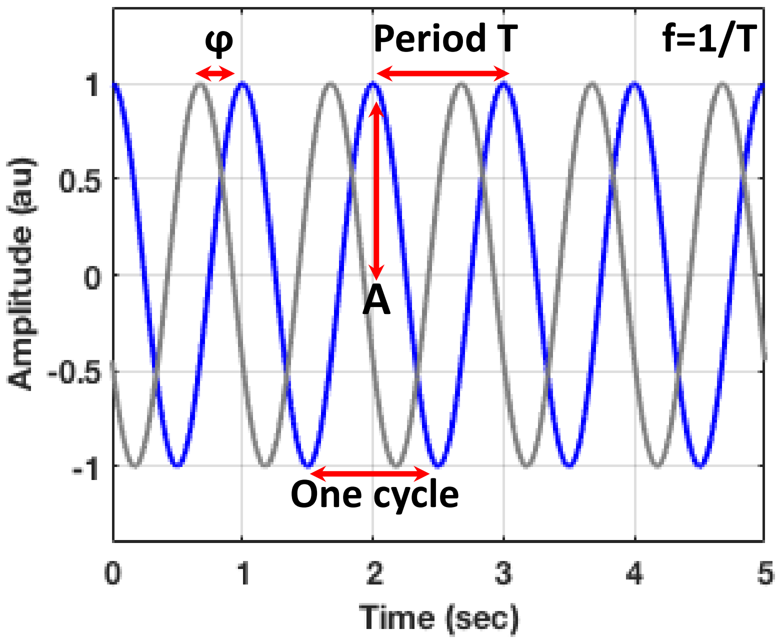


Sampling range: What are the maximum/minimum values we can sample?

Resolution/precision: Range divided by depth
For example: Range +/- 10 µV, 8 bit sampling depth => 20/256 ≈ 0.08 µV

Sampling Frequency is crucial

Let's define a sinusoidal signal with frequency 1Hz:



Y = A cos(ωt+φ) -> A cos(2πft+φ)

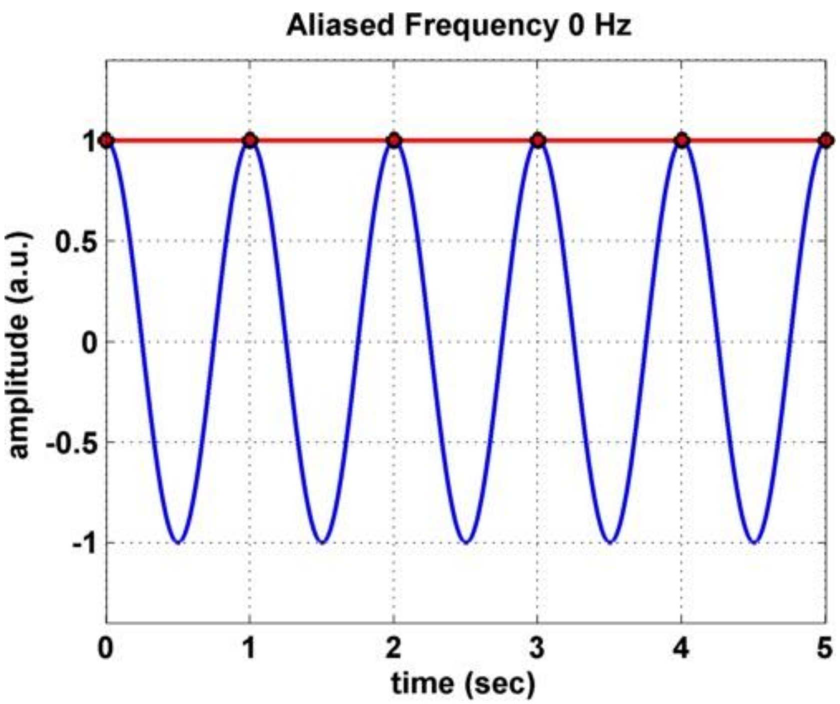
A: amplitude
ω = 2πf, # of cycles per unit time
φ: size of the phase shift

```
% I do not like floating figures…
set(0,'DefaultFigureWindowStyle','docked');

% let's generate a sinusoidal signal
tWin = 5;%temporal window
t = 0:0.0001:tWin;
Phi = 0; f = 1;%Hz
signal= cos(2*pi*f*t+Phi);
figure
hold on;grid on;
plot(t,signal,'b','linewidth',2);
```

Down-sampling can lead to Aliasing (i.e. distorted discrete signal)

Aliasing: artefact that results when the discrete signal (reconstructed from samples) differs from the original continuous signal.

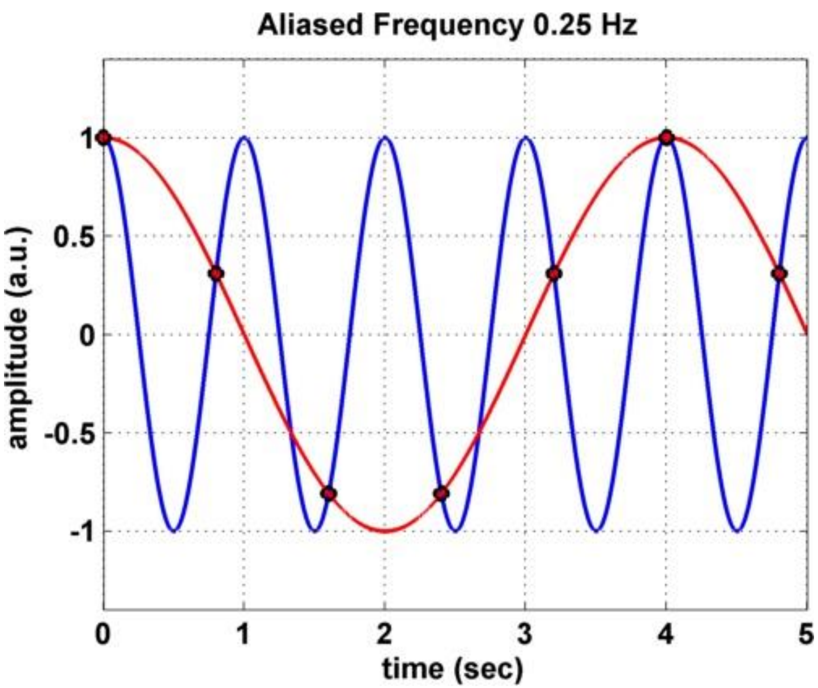
Signal frequency = 1Hz / Sampling frequency = 1Hz



Source: Imgur.com

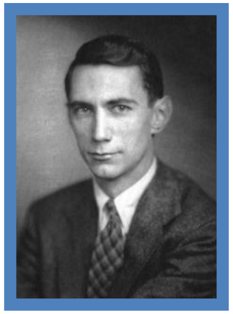
Down-sampling can lead to Aliasing (i.e. distorted discrete signal)

Signal frequency = 1Hz / Sampling frequency = 1.25Hz



Source: Imgur.com

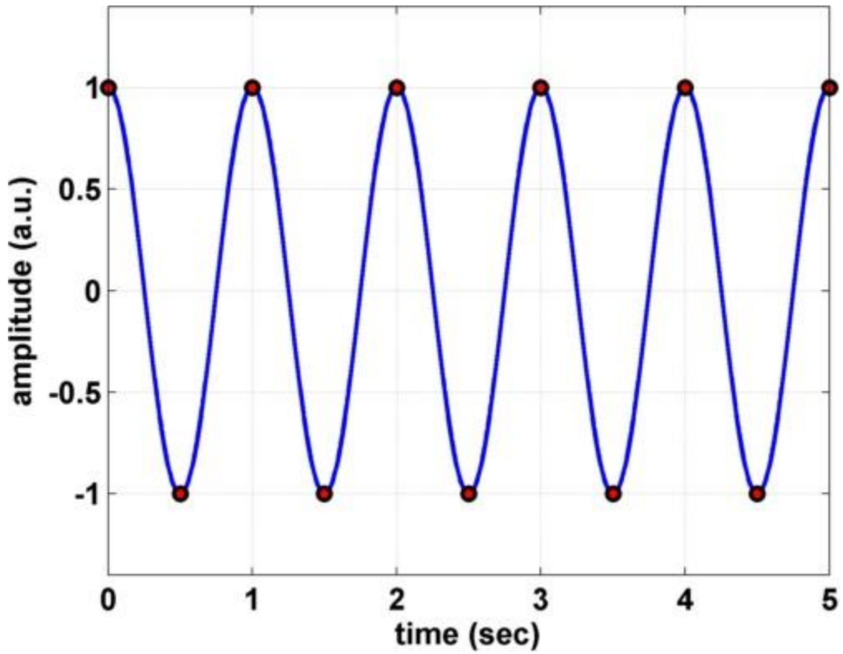
```
Fs = 1.25; %sampling frequency
ts = 0:1/Fs:tWin;

SampAlias= cos(2*pi*f*ts); %samples
plot(ts,SampAlias,'ko','linewidth',2,'MarkerFaceColor','r');

AliasF = abs(floor(Fs/f)*Fs-f); %calculate aliased frequency
ASignal=cos(2*pi*AliasF*t);% calculate aliased signal
plot(t,ASignal,'r','linewidth',2);
```



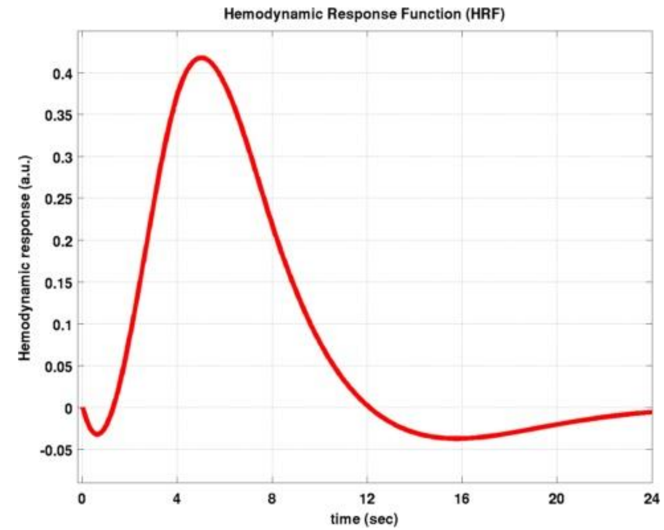
Nyquist - Shannon Sampling Theorem

- If you sample a signal with a sampling rate of **X** Hz, make sure the signal doesn't contain frequencies above **X/2** Hz
- ***Nyquist Frequency***: half of the sampling rate of a discrete signal
- **The largest frequency in the signal should be smaller than the Nyquist Frequency**

Sampling frequency = 2Hz; Nyquist Frequency = 1Hz



Examples



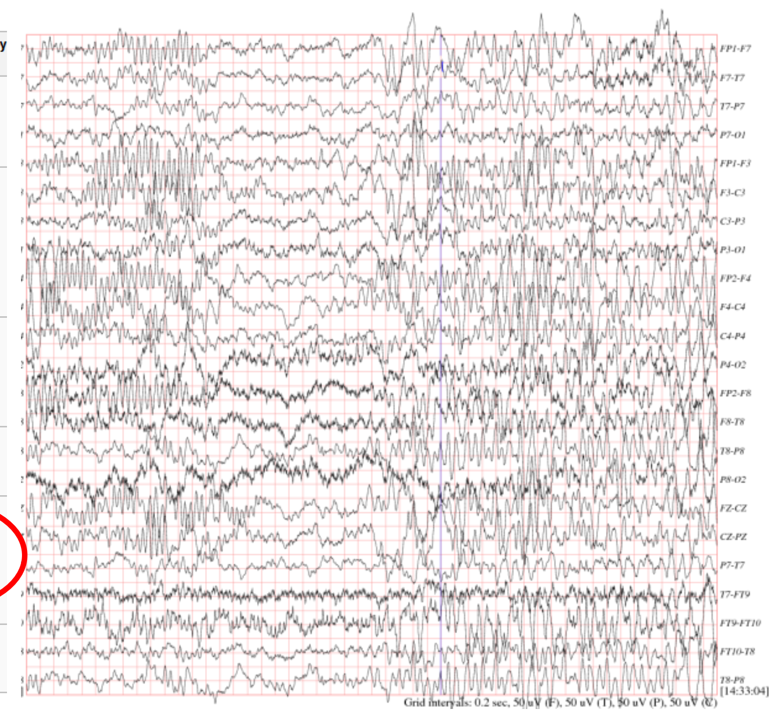
fMRI

Typically sampled every 2 seconds

(0.5 Hz with TR = 2s)

***Nyquist Frequency** = Fsamp/2 = 0.25 Hz*

Type	Frequency (Hz)
Delta	up to 4
Theta	4 – 8
Alpha	8 – 13
Beta	>13 – 30
Gamma	30 – 100+
Mu	8 – 13



EEG/MEG

Typically sampled every 2 msec

(500Hz)

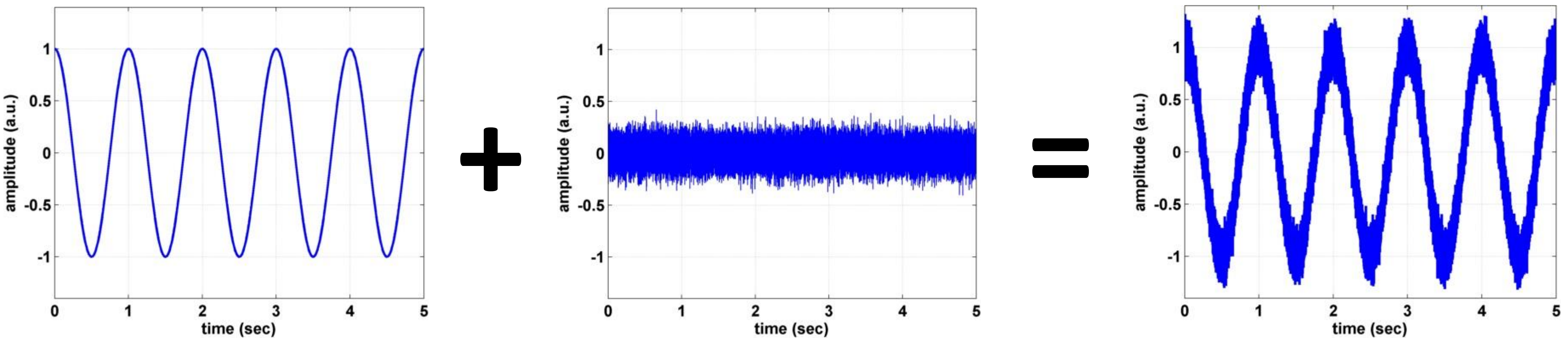
***Nyquist Frequency** = 250 Hz*

(NB: well above the highest freq band)

Noise & Error propagation

***Noise** is a general term for alterations that a signal may suffer because of :*

- *Inaccuracies of measurement equipment*
- *Interference from artefact sources*
- *Modelling errors*

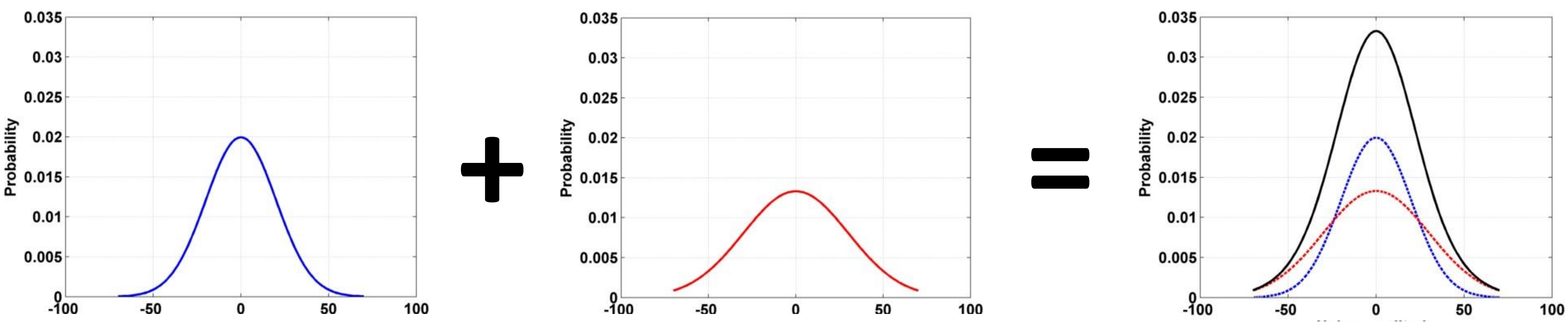



```
lx = length(signal);
noise = 0.1*randn(1,lx);
Figure; plot(t,noise);

nSignal = noise+signal;
plot(t,nSignal,'b','linewidth',2)
grid on;box on;
xlabel('time (sec)');ylabel('amplitude (a.u.)')
```

Noise & Error propagation

Any transformation of the data will be affected by noise, and may amplify it

For example: Subtracting/adding data sets with equal variance doubles the variance




```
Nvalues = -70:70;
S_mean = 0;
S_sd1 = 20;S_sd2 = 30;
Signal1 = normpdf(Nvalues,S_mean,S_sd1);
Signal2 = normpdf(Nvalues,S_mean,S_sd2);

figure;
bar([1 2 3],[var(Signal1),var (Signal2),var (Signal1+Signal2)])
set(gca,'XTickLabel',{'Noise1','Noise2','N1+N2'})
Title('Standard deviation')
```

Noise & Error propagation

If the operation is more complex (e.g. derivative), the effect of noise will probably be more complex.

```
subplot(1,2,1)
plot(diff(signal));
title('clean signal');

subplot(1,2,2)
plot(diff(nSignal));
title('noisy signal')
```

Data Quality: Signal-to-Noise Ratio (SNR)

Signal-to-Noise ratio: compare the level of “signal” to the level of “noise” .

Common definition for SNR:
Divide power (variance) of signal by power (variance) of noise

$$SNR = \frac{P_{Signal}}{P_{Noise}}$$

Other definitions possible:

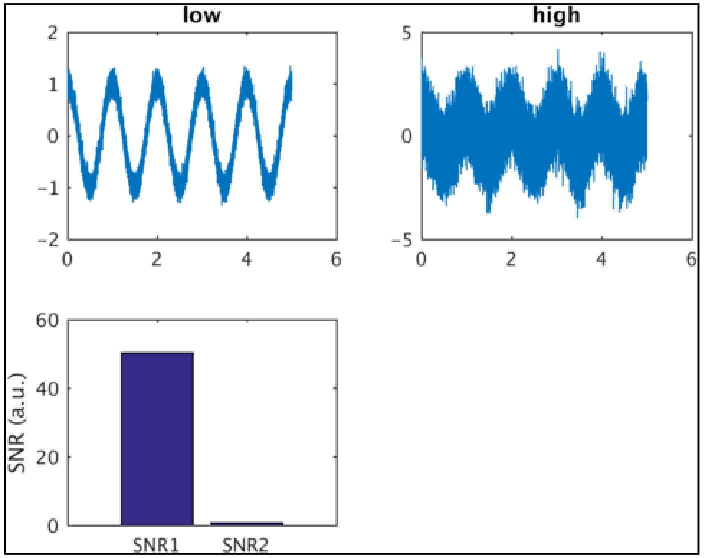
Divide amplitude of signal by standard deviation of noise

Divide root-mean-square (RMS) of signal by RMS of noise

Decibels: $SNR_{dB} = 10log_{10}\frac{P_{Signal}}{P_{Noise}} = P_{Signal,dB} - P_{Noise,dB}$

Data Quality: Signal-to-Noise Ratio (SNR)

$$SNR = \frac{P(var)Signal}{P(var)Noise}$$




```
lx = length(signal);
noise_low  = 0.1*randn(1,lx);
noise_high = 0.8*randn(1,lx);

SNR1 = var(signal)/var(noise_low);
SNR2 = var(signal)/var(noise_high);

figure;
subplot(2,2,1);plot(t,signal+noise_low);title('low');
subplot(2,2,2);plot(t,signal+noise_high);title('high');

subplot(2,2,3);bar([SNR1,SNR2]);set(gca,'XTickLabel',{'SNR1','SNR2'}); ylabel('SNR (a.u.)')
```

The End…