Phys 238: Uncertainties resulting from Random, Independent errors

In[72]:=
```
Clear["Global`*"]
```

Individual Random Numbers

In[73]:=
```
RandomReal[{5 , 7}] (* Generates a random number between 5 and 7. *)
```

Out[73]=
```
5.20891
```

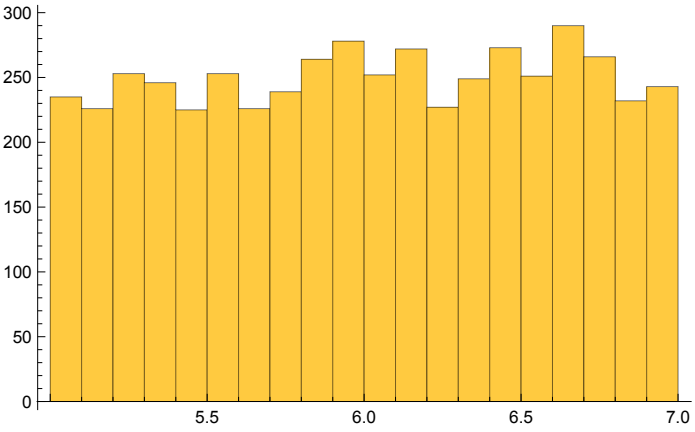
In[74]:=
```
RandomReal[{5, 7}, 10] (* Generate a bunch (10) of them *)
```

Out[74]=
```
{5.06997, 5.86994, 5.60286, 5.67317,
 5.31659, 6.96606, 5.6477, 6.87101, 5.27136, 5.4901}
```

In[75]:=
```
vals = RandomReal[{5, 7}, 5000]; (* A big bunch *)
```

In[76]:=
```
Histogram[vals]
```

Out[76]=



Repeating a single measurement that results from the sum of many individual random numbers should give a normal distribution.

Define a function that represents one data point that is a result of many independent random numbers. In this example, “many” = 200.

In[77]:=
```
onepoint := Mean[RandomReal[{5, 7}, 200]]
```

This returns a different measurement each time you try it.

In[78]:=
```
onepoint
```

Out[78]=
```
6.0662
```

In[79]:=
```
onepoint
```

Out[79]=
```
6.01492
```

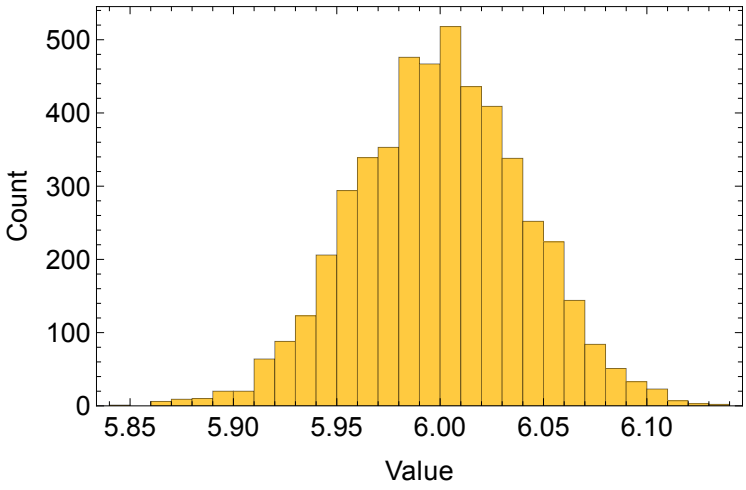
Now imaging doing 5000 such measurements.

In[80]:=
```
data = Table[onepoint, {i, 1, 5000}];
```

Display a histogram with useful plot labels, and in a larger size (scaled to 60% of the screen size here) . Give this graph a name so we can use it again later.

In[128]:=
```
histplot = Histogram[data, Frame → True,
  FrameLabel → {"Value", "Count"}, LabelStyle → Larger, ImageSize → Scaled[0.6]]
```

Out[128]=



In[82]:=
```
μ = Mean[data]
σ = StandardDeviation[data]
δμ = σ / Sqrt[Length[data]]
```

Out[82]=
```
5.99995
```

Out[83]=
```
0.0408734
```

Out[84]=
```
0.000578037
```

How many are within ± 1 σ of the mean? Use the 'Select' function to select data points within σ of μ.

In[85]:=
```
nin1 = Length[Select[data, μ - σ ≤ # ≤ μ + σ &]] / Length[data];
StringForm["``, or ``%", nin1, NumberForm[100.0 * nin1, 5]]
```

Out[86]=
$\frac{3387}{5000}$, or 67.74%

Now try to model this distribution by a Gaussian. Getting a manual list of the histogram numbers is a bit tedious. The HistogramList[ ] function actually gives two separate lists, first is a list of bin boundaries, and second is the list of counts in each bin.

In[87]:=
```
{bins, counts} = N[HistogramList[data]]
```

Out[87]=
```
{{5.84, 5.85, 5.86, 5.87, 5.88, 5.89, 5.9, 5.91, 5.92,
  5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 6., 6.01, 6.02, 6.03,
  6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.1, 6.11, 6.12, 6.13, 6.14},
 {1., 0., 6., 9., 10., 20., 20., 64., 88., 123., 206., 294., 339., 353., 476.,
  467., 518., 436., 409., 338., 252., 224., 144., 84., 51., 33., 23., 7., 3., 2.}}
```

In[88]:=
```
bins
```

Out[88]=
```
{5.84, 5.85, 5.86, 5.87, 5.88, 5.89, 5.9, 5.91, 5.92,
 5.93, 5.94, 5.95, 5.96, 5.97, 5.98, 5.99, 6., 6.01, 6.02, 6.03,
 6.04, 6.05, 6.06, 6.07, 6.08, 6.09, 6.1, 6.11, 6.12, 6.13, 6.14}
```

In[89]:=
```
counts
```

Out[89]=
```
{1., 0., 6., 9., 10., 20., 20., 64., 88., 123., 206., 294., 339., 353., 476.,
 467., 518., 436., 409., 338., 252., 224., 144., 84., 51., 33., 23., 7., 3., 2.}
```

Because the ‘bins’ array specifies the boundaries, the length of the ‘bins’ array is actually one more than the length of the ‘counts’ array.

In[90]:=
```
Length[bins]
Length[counts]
```

Out[90]=
```
31
```

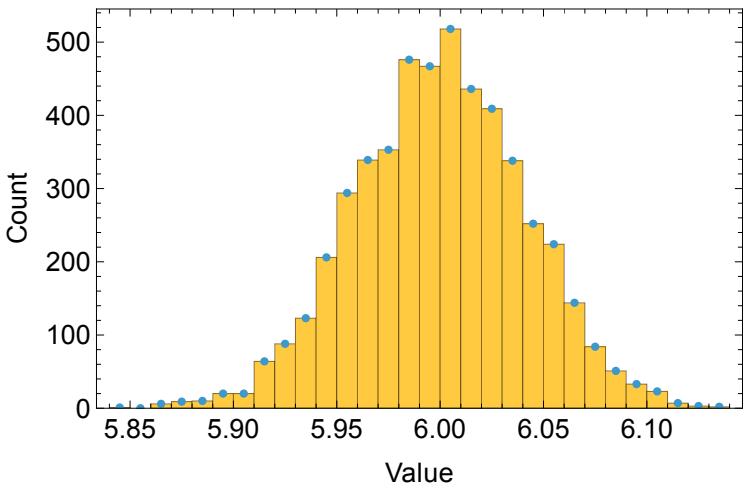
Out[91]=
```
30
```

Interleaving them is straightforward, if verbose. We will assign each height to the center point of each bin by constructing a new table.

In[92]:=
```
hist = Table[{(bins[[i]] + bins[[i + 1]]) / 2., counts[[i]]},
   {i, 1, Length[counts]}];
```

In[129]:=
```
(* Show those data points superimposed on our histogram
 to make sure everything looks right.  Note that the Show[ ]
 command takes a list of plots (in curly braces) to display. *)
Show[{histplot, ListPlot[hist]}]
```

Out[129]=



Enter an unnormalized gaussian (Eq. 5.20 in Taylor's text):

In[115]:=
```
gauss[A_, μ_, σ_, x_] := A/(σ Sqrt[2 π]) Exp[-(x - μ)^2/(2 σ^2)]
```

In[131]:=
```
Clear[A, μ, σ]
(* Fit the data to our gaussian *)
result = NonlinearModelFit[hist, gauss[A, μ, σ, x],
  {A, μ, σ}, x]
```

NonlinearModelFit: Failed to converge to the requested accuracy or precision within 100 iterations.

Out[132]=
```
FittedModel[3.66 ×10^3 ℯ^(-0.0496 (-≪19≫+x)^2)]
```

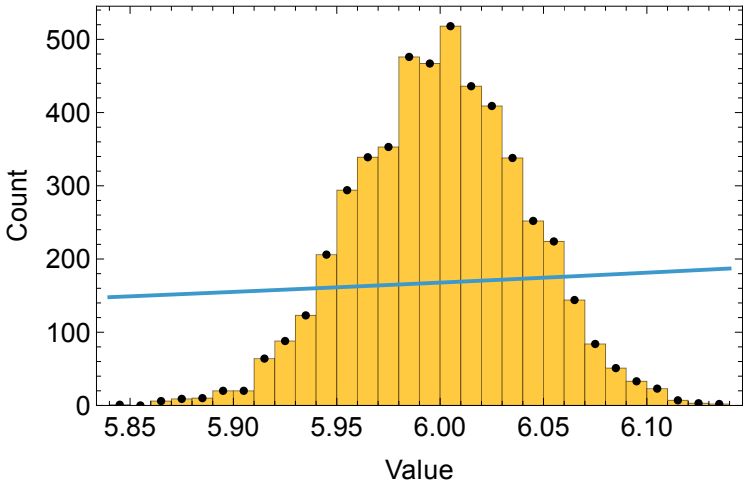
In[133]:=
```
result["BestFitParameters"]
```

Out[133]=
```
{A → 29 090.7, μ → 13.882, σ → 3.17514}
```

Show the histogram, the data used in the fit, and the best-fit gaussian all on the same graph. Add useful titles. This fit is very poor.

In[134]:=
```
randerrplot = Show[{
   histplot,
   ListPlot[hist, PlotStyle → Black], Plot[result[x], {x, Min[bins], Max[bins]}]
  }
 ]
```

Out[134]=



The initial fit attempt does always work well. It is useful to seed the process with guesses for the 'μ' and 'σ' parameters. These don't have to be exact, but it helps if they are close. Here, we can just use the calculated mean and standard deviation.

In[135]:=
```
Clear[A, μ, σ]
result = NonlinearModelFit[hist, gauss[A, μ, σ, x],
  {A,
   {μ, Mean[data]},
   {σ, StandardDeviation[data]}
  }, x]
```

Out[136]=

FittedModel[486. $e^{-296.\ (\ll 1 \gg)^2}$]

In[137]:=
```
result["BestFitParameters"]
```

Out[137]=

{A → 50.0578, μ → 6.00027, σ → 0.0411088}

In[138]:=
```
result["ParameterConfidenceIntervalTable"]
```

Out[138]=

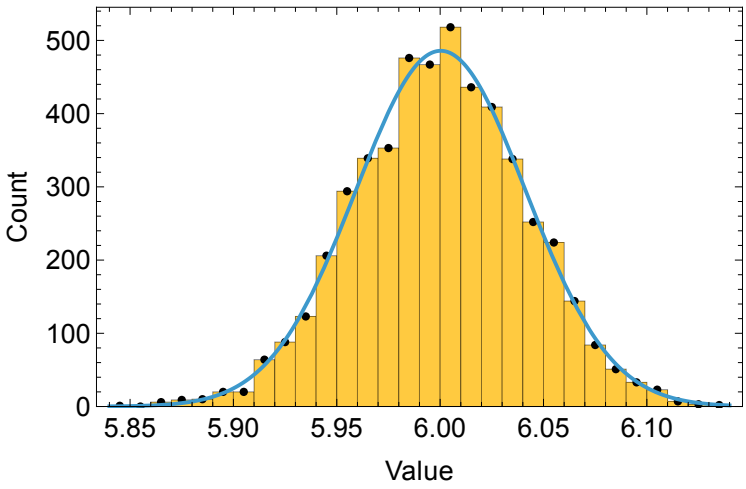
	Estimate	Standard Error	Confidence Interval
A	50.0578	0.76618	{48.4857, 51.6298 }
μ	6.00027	0.000726524	{5.99878, 6.00176 }
σ	0.0411088	0.000726595	{0.039618, 0.0425997 }

These are very close to the calculated mean and standard deviation:

In[139]:=
```
{Mean[data], StandardDeviation[data]}
```
Out[139]=
```
{5.99995, 0.0408734}
```

Show the histogram, the data used in the fit, and the best-fit gaussian all on the same graph. Add useful titles.

In[140]:=
```
randerrplot = Show[{
   histplot,
   ListPlot[hist, PlotStyle → Black], Plot[result[x], {x, Min[bins], Max[bins]}]
  }]
```
Out[140]=



In[141]:=
```
SetDirectory[NotebookDirectory[]] (* Save it in this directory *)
```
Out[141]=
```
/Users/doughera/238/2026/lectures-dev/Ch05
```

In[142]:=
```
Export["randerrplot.pdf", randerrplot]
```
Out[142]=
```
randerrplot.pdf
```