# Phys 238 : The Simple Pendulum

## Reading .CSV Files With *Mathematica*

## Getting Started

*Mathematica* remembers definitions, even if you delete them from the notebook. This can sometimes be confusing, so it's often a good idea to simply clear everything and start anew. This is especially true if you use the Evaluation -> Evaluate Notebook menu item.

```
In[3]:= Clear["Global`*"] (* Clear all variable and function definitions *)
```

## Reading a CSV file

### Preliminaries

It is not always obvious where Mathematica will look for data files or export graphs. It is often useful to keep your data in the same directory as the notebook. The following command sets the default directory to be the same as the directory for this notebook.

```
In[4]:= SetDirectory[NotebookDirectory[]]
Out[4]= /Users/doughera/notes/Physics-Labs-git/intermediate/Phys238/pendulum
```

### Initial Import

Mathematica has many different ways to read and interpret .csv (Comma Separated Values) files. One high-level way is to use what Mathematica calls a “Dataset”, which is a structured object with its own special query language. (In Python, an analogous approach is to use the Pandas library.) Unfortunately, Mathematica's Datasets do not work well with poorly structured CSV files, and multidimensional slices of Datasets can be challenging to use with some other Mathematica functions. Both issues can arise in data we will encounter this semester. Accordingly, this notebook will use a more basic, but more flexible, approach.

First, import all the data as a “CSV” file. Use a semicolon at the end of the statement so that the entire data file isn't printed out.

```
In[5]:= fulldata = Import["pendulum-20250129.csv", "CSV"];
```

‘fulldata’ is now an array containing all the contents of the file. Before doing anything further, it is worthwhile to look at the first few elements of the data. Use the double bracket notation to extract individual elements of the array. For example, the first row is given by fulldata[[ 1 ]]. Note that the row is itself a list (in curly braces) of 3 elements, which are the titles for the columns of data. There are 3 columns, Time, GateState, and Period. GateState is either ‘1’ or ‘0’, depending on whether the photo-gate was blocked or not.

```
In[6]:= fulldata⟦1⟧
Out[6]= {Latest: Time (s), Latest: GateState, Latest: Period (s)}
```

You can see more rows by replacing the single number ‘1’ by a list (in curly braces) of the rows you want to see. For example, the first 6 rows are

```
In[7]:= fulldata⟦{1, 2, 3, 4, 5, 6}⟧
Out[7]= {{Latest: Time (s), Latest: GateState, Latest: Period (s)},
         {0.139623, 1, Missing[NotAvailable]},
         {0.222001, 0, Missing[NotAvailable]}, {1.14797, 1, Missing[NotAvailable]},
         {1.23138, 0, Missing[NotAvailable]}, {2.07431, 1, 1.93468}}
```

This is a nested list of lists. There are 6 rows, each of which is a list of 3 items.  A useful way to display such a list of lists is as a Table, using the TableForm[ ] command.

```
In[8]:= TableForm[ fulldata⟦{1, 2, 3, 4, 5, 6}⟧ ]
Out[8]//TableForm=
        Latest: Time (s)    Latest: GateState    Latest: Period (s)
        0.139623            1                    Missing[NotAvailable]
        0.222001            0                    Missing[NotAvailable]
        1.14797             1                    Missing[NotAvailable]
        1.23138             0                    Missing[NotAvailable]
        2.07431             1                    1.93468
```

Finally, a quick shorthand for the 1-6 range is to use the Range function:

```
In[9]:= Range[6]
Out[9]= {1, 2, 3, 4, 5, 6}
```

So the first 10 rows are

```
In[10]:= TableForm[ fulldata⟦Range[10]⟧ ]
Out[10]//TableForm=
        Latest: Time (s)    Latest: GateState    Latest: Period (s)
        0.139623            1                    Missing[NotAvailable]
        0.222001            0                    Missing[NotAvailable]
        1.14797             1                    Missing[NotAvailable]
        1.23138             0                    Missing[NotAvailable]
        2.07431             1                    1.93468
        2.15672             0                    Missing[NotAvailable]
        3.08287             1                    Missing[NotAvailable]
        3.16634             0                    Missing[NotAvailable]
        4.00907             1                    1.93477
```

## Extracting the Relevant Data

The data consists of 3 columns. We are only interested in the period, which is in the third column. For illustrative purposes, work with a short subset of the full data.

```
In[11]:= short = fulldata⟦Range[11]⟧;
         TableForm[short]

Out[12]//TableForm=
         Latest: Time (s)    Latest: GateState    Latest: Period (s)
         0.139623            1                    Missing[NotAvailable]
         0.222001            0                    Missing[NotAvailable]
         1.14797             1                    Missing[NotAvailable]
         1.23138             0                    Missing[NotAvailable]
         2.07431             1                    1.93468
         2.15672             0                    Missing[NotAvailable]
         3.08287             1                    Missing[NotAvailable]
         3.16634             0                    Missing[NotAvailable]
         4.00907             1                    1.93477
         4.09148             0                    Missing[NotAvailable]
```

To extract a full row, you use the [[ ]] notation. To extract a specific element within that row, you have to specify both the row and column. For example, to extract the 3rd item of row 6, you would write

```
In[13]:= short⟦6, 3⟧
Out[13]=
         1.93468
```

You can replace the ‘6’ by a list (in curly braces) of rows to extract:

```
In[14]:= short⟦{6, 7, 8, 9, 10}, 3⟧
Out[14]=
         {1.93468, Missing[NotAvailable],
          Missing[NotAvailable], Missing[NotAvailable], 1.93477}
```

The special entry “All” will extract all rows:

```
In[15]:= short⟦All, 3⟧
Out[15]=
         {Latest: Period (s), Missing[NotAvailable], Missing[NotAvailable],
          Missing[NotAvailable], Missing[NotAvailable], 1.93468, Missing[NotAvailable],
          Missing[NotAvailable], Missing[NotAvailable], 1.93477, Missing[NotAvailable]}
```

It is not useful for this data set, but you can also select multiple columns in a similar way:

```
In[•]:= short⟦6, {1, 3}⟧ (* Extract row 6, columns 1 and 3 *)
Out[•]=
         {2.07431, 1.93468}
```

```
In[•]:= short⟦6, All⟧ (* Extract row 6, All columns *)
Out[•]=
         {2.07431, 1, 1.93468}
```

So here are the values from column 3 of our shortened data set, shown in Table format:

```
In[16]:= col3 = short⟦All, 3⟧;
```

```
In[17]:= TableForm[ col3 ]
Out[17]//TableForm=
         Latest: Period (s)
         Missing[NotAvailable]
         Missing[NotAvailable]
         Missing[NotAvailable]
         Missing[NotAvailable]
         1.93468
         Missing[NotAvailable]
         Missing[NotAvailable]
         Missing[NotAvailable]
         1.93477
         Missing[NotAvailable]
```

## Using the Select Function

The next challenge is to select the elements from the 3rd column that correspond to period values. Mathematica has a function ‘NumberQ’ that tests if an element is a number. Consider row 2 and row 6:

```
In[18]:= col3⟦2⟧
Out[18]=
         Missing[NotAvailable]

In[19]:= NumberQ[col3⟦2⟧]
Out[19]=
         False

In[20]:= col3⟦6⟧
Out[20]=
         1.93468

In[21]:= NumberQ[col3⟦6⟧]
Out[21]=
         True
```

Next, we need to Select the elements from column 3 that are numbers. Mathematica’s Select[ ] function does this.

```
In[22]:= Select[col3, NumberQ[#] &]
Out[22]=
         {1.93468, 1.93477}
```

The Select function takes two arguments: Select[data, criterion]. It steps through each individual element of ‘data’ and applies ‘criterion’ to each row. Specifically, it sets the special variable ‘#’ equal to each element in turn, and returns only those elements for which ‘NumberQ’ returns true. You have to include the ‘&’ sign so to apply the NumberQ function to the ‘#’ data element. (See the help for Function for more details, but it is rather dense reading.)

So one way to get all the period values out of the full data set is

```
In[23]:= data = Select[fulldata[[All, 3]], NumberQ[#] &];
```

Looking at the first 20 periods:

```
In[24]:= data[[Range[20]]]
Out[24]=
{1.93468, 1.93477, 1.93487, 1.93471, 1.9345, 1.93472,
 1.93484, 1.93484, 1.93457, 1.93453, 1.9349, 1.93494, 1.93465,
 1.93491, 1.93465, 1.93485, 1.93472, 1.93458, 1.93455, 1.93494}
```

### Saving the extracted data

For a future homework assignment, you will need the period values. It is handy to export the values now with the ‘Export’ command. This will create a data file with all the period values in a single column, suitable for easy import into another program.

```
In[25]:= Export["pendulum-20250129-period.csv", data]
Out[25]=
pendulum-20250129-period.csv
```

### Summary of Import:

For this data set, here are the relevant commands,

```
In[26]:= SetDirectory[NotebookDirectory[]]
Out[26]=
/Users/doughera/notes/Physics-Labs-git/intermediate/Phys238/pendulum

In[27]:= fulldata = Import["pendulum-20250129.csv", "CSV"];

In[28]:= data = Select[fulldata[[All, 3]], NumberQ[#] &];
```

(Optional) Save the period data for later importing.

```
In[29]:= Export["pendulum-20250129-period.csv", data]
Out[29]=
pendulum-20250129-period.csv
```

## Statistical Analysis

Here is how to calculate various important statistics for the data set. Most of the commands should be self-evident.

```
In[30]:= npts = Length[data]
Out[30]=
516

In[31]:= T = Mean[data]
Out[31]=
1.93466
```

```
In[32]:= σT = StandardDeviation[data]
Out[32]=
         0.000168487
```

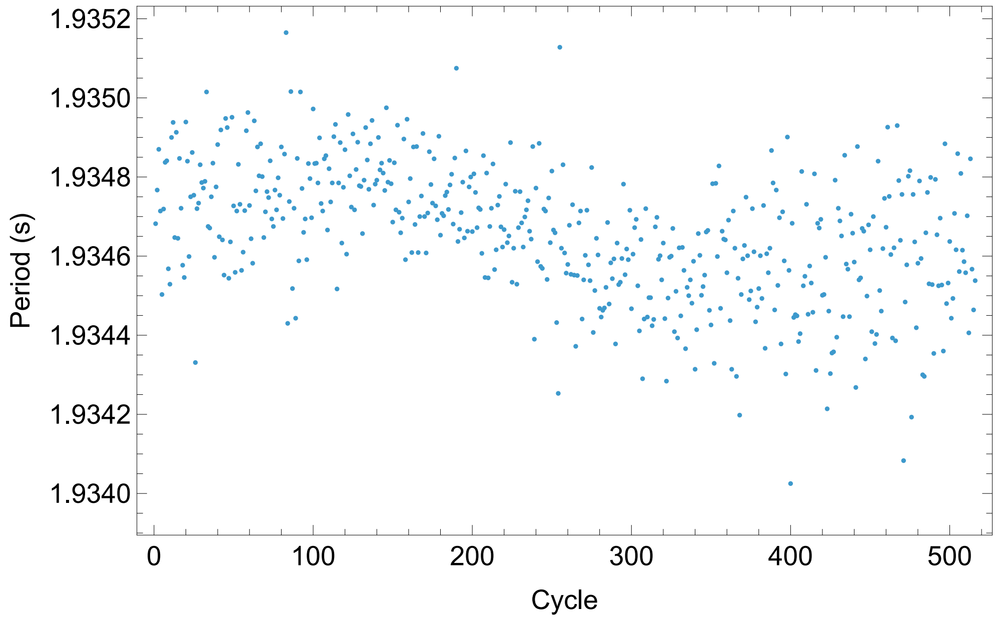
```
In[33]:= δT = σT / Sqrt[npts]
Out[33]=
```
$7.41724 \times 10^{-6}$

Mathematica can express this as a value with uncertainty using the Around[ ] function:

```
In[34]:= Around[T, δT]
Out[34]=
         1.93465(6 ± 7)
```

Here is a plot of the data, with sensible labels. Save the plot in the ‘Tplot’ variable to use later.

```
In[36]:= Tplot = ListPlot[data, LabelStyle → Larger, Frame → True,
           FrameLabel → {"Cycle", "Period (s)"}, ImageSize → Scaled[0.8]]
Out[36]=
```

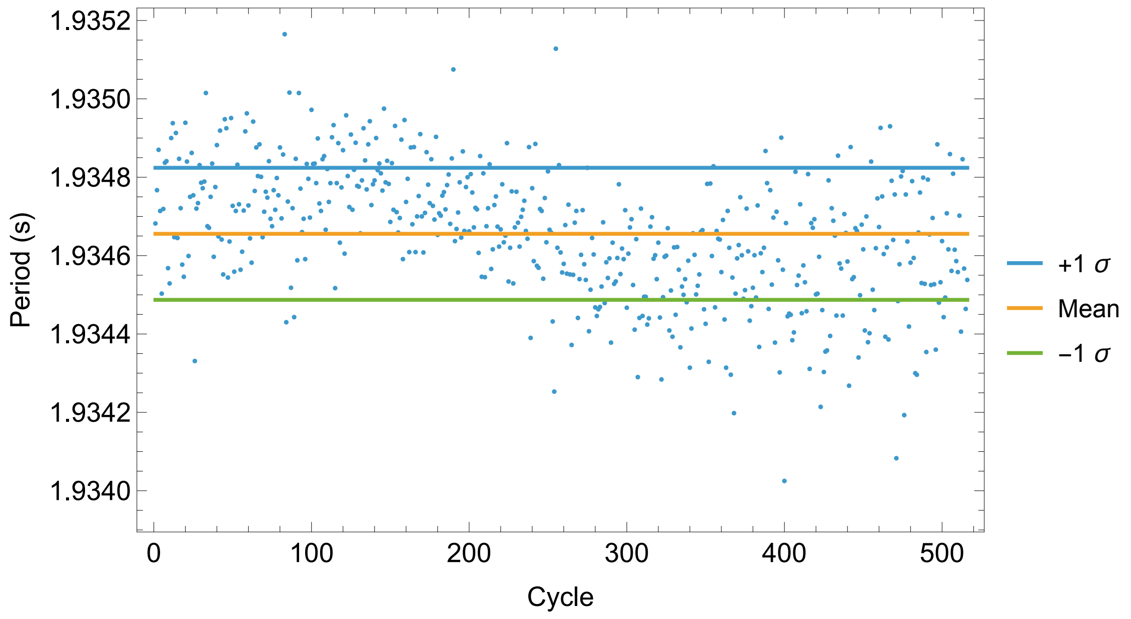


## Further explorations

We can illustrate the ±1 $\sigma$ range on the plot. Note that the ‘Show’ command combines graphics objects, in this case the data plot we made above, and a new plot showing the ±1 $\sigma$ range.

```
In[37]:= TCompPlot = Show[{
             Tplot,
              Plot[{T + σT, T, T - σT}, {cycle, 1, Length[data]},
                PlotLegends → {"+1 σ", "Mean", "-1 σ"}]
           }]
```

Out[37]=



```
In[38]:= Export["pendulum-period-plot.pdf", TCompPlot]
```

Out[38]=

```
pendulum-period-plot.pdf
```

We can also count the numbers with a particular range with the Select[ ] function. This function steps through the ‘data’ array and assigns the special symbol ‘#’ to the current element of the array. (The ‘&’ must also be included at the end of the comparison.)

```
In[39]:= within1 = Select[data, T - σT ≤ # ≤ T + σT &];
         100. * Length[within1] / npts
```

Out[40]=

```
67.8295
```

```
In[41]:= within2 = Select[data, T - 2 σT ≤ # ≤ T + 2 σT &];
         100. * Length[within2] / npts
```

Out[42]=

```
95.5426
```

```
In[43]:= within3 = Select[data, T - 3 σT ≤ # ≤ T + 3 σT &];
         100. * Length[within3] / npts
```

Out[44]=

```
99.4186
```