Looking at the Weather

In this notebook we will take a look at how *Mathematica* can be used to gather some real-world data and then fit curves to it.

Getting some temperature data

One of the many data types *Mathematica* has access to is weather information. Here we take the average daily temperature measured at the weather station KDUA, conveniently located at Eaker airport, from 2003 to August 2015: (note - this will take a minute or so to evaluate as your computer will need to download the appropriate information)

```
rawdata = WeatherData[ "KDUA", "MeanTemperature",
   {{2003, 1, 1}, {2015, 8, 1}, "Day"}, "DateNonMetricValue"];
```

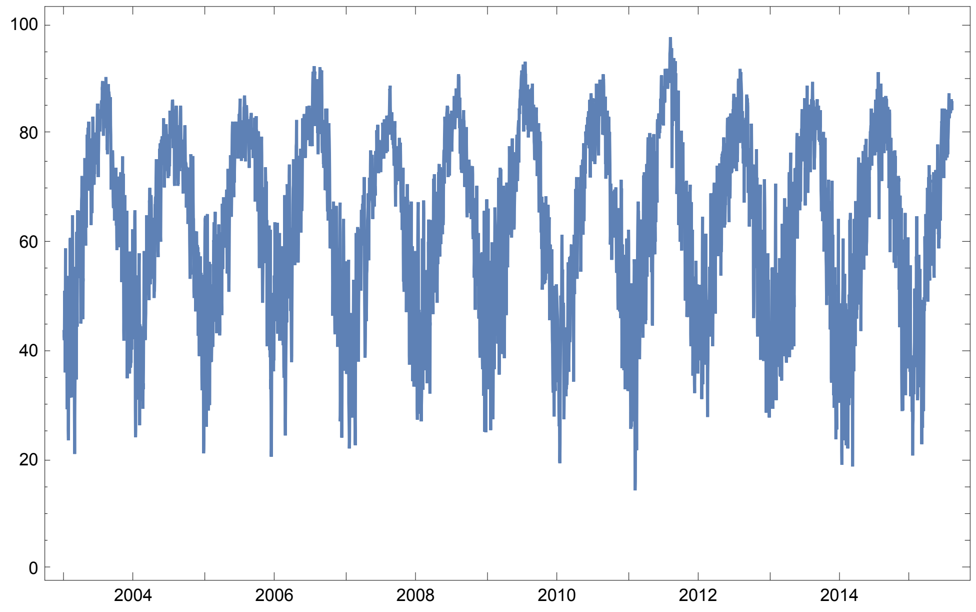
The raw data format is not terribly friendly, as we can see by looking at the first 5 data points:

```
rawdata["Path"][[1 ;; 5]]
```

```
{{3 250 368 000, 43.7}, {3 250 454 400, 42.69},
 {3 250 540 800, 42.1}, {3 250 627 200, 48.79}, {3 250 713 600, 51.21}}
```

The first number is the elapsed seconds since the start of the year 1900. Fortunately *Mathematica*'s DateListPlot can graph this for us without needing to worry about the data format:

```
DateListPlot[ rawdata, ImageSize → 500]
```

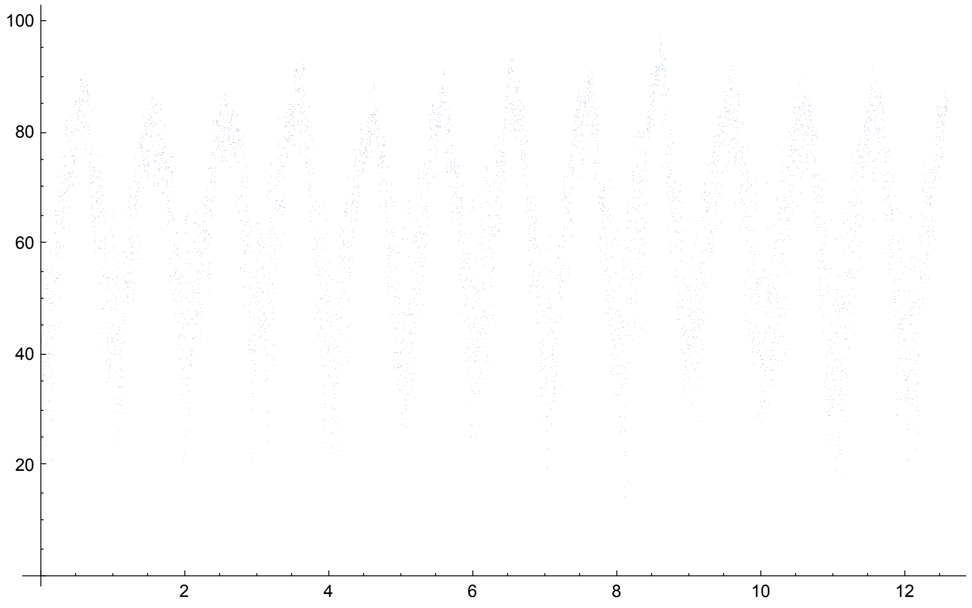


For the purposes of fitting curves to the data this is not the best format for the data. The following command converts the data to the format {"years since 1/1/2003", average temperature} with all of the units stripped off (it also ignores any data values that have the header "Missing", which appear in the data to indicate gaps):

```
data2 = Map[
   {N[DateDifference[ {2003, 1, 1}, #[[1]], "Year"] // QuantityMagnitude], #[[2]]} &,
   Select[rawdata["Path"], FreeQ[#, Missing] &]];
```

Now we have a raw list of points which we can graph and can do analysis on:

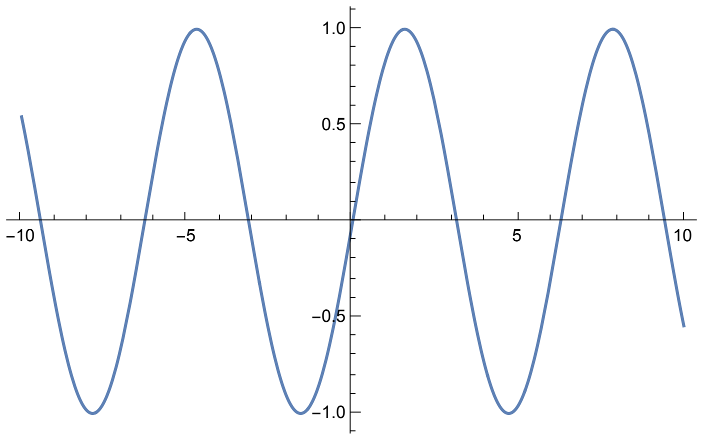
```
ListPlot[data2, PlotStyle → PointSize[Tiny], ImageSize → 500]
```



Some basic waves

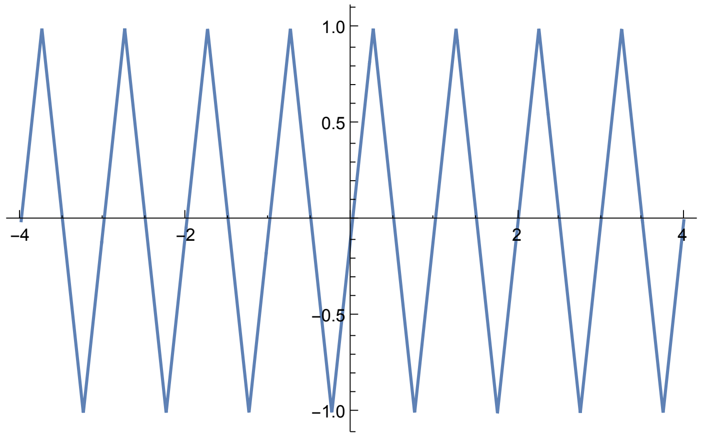
A standard “smooth” wave is reprented by the sine function:

```
Plot[ Sin[x], {x, 0 - 10, 10}]
```



Another common wave is a “triangle” wave, which is piecewise linear and continuous but not differentiable:

```
Plot[TriangleWave[x], {x, -4, 4}]
```



Fitting waves to the temperatures

The standard sine wave has an amplitude of 1, a period of 2π, no shift to the left or right, and is centered on the $x-$axis. To find a sine-like wave which best fits the data we use the general formula $a \sin(b\,x\ + c) + d$, where a represents the amplitude, $\frac{2\pi}{b}$ the period, c controls shifts to the left or right, and d represents the center of the wave. To find the best fitting curve of this form to our data we use the NonlinearModelFit command (we need to use NonlinearModelFit as this is non-linear regression). As the sine wave has lots of symmetries to it there will be many optimal solutions, so it is best to give some initial seed values for a, b, c, and d to get a good fit. Looking at the original graph we can estimate the amplitude of the graph to be about 20, the period to be roughly 1 year (so b should be near 2π), and the average temperature seems to be around 63.5 degrees. Using these as seed values gives us this command:

```
sinewave =
 NonlinearModelFit[data2, a * Sin[b x + c] + d, {{a, 20}, {b, 2 Pi}, c, {d, 63.5}}, x]
```

```
FittedModel[ 63.6332 - 20.6964 Sin[1.30227 + 6.28271 x] ]
```

The FittedModel result contains lots of information beyond just the formula, but we can use it to get the formula directly:

```
sinewave[x]
```

```
63.6332 - 20.6964 Sin[1.30227 + 6.28271 x]
```

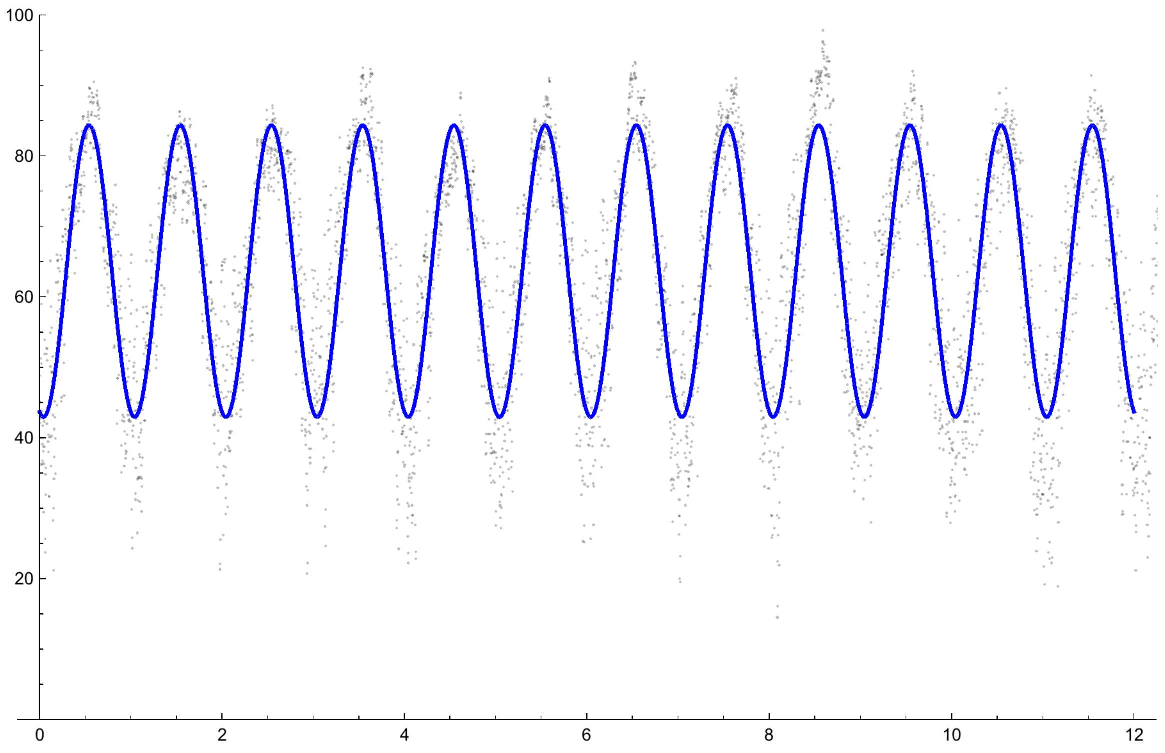
Some of the additional information embedded in the fitted model are confidence intervals for each parameter (by default the 95% condifence interval - this can be controlled by one of the options in NonlinearModelFit_:

```
sinewave["ParameterConfidenceIntervalTable"]
```

	Estimate	Standard Error	Confidence Interval
a	−20.6964	0.164077	{−21.0181, −20.3747}
b	6.28271	0.00220719	{6.27838, 6.28703}
c	1.30227	0.0160367	{1.27083, 1.33371}
d	63.6332	0.116492	{63.4048, 63.8615}

We can of course graph the model againt our data:

```
Plot[ sinewave[x], {x, 0, 12},  PlotStyle → Directive[Thick, Blue],
 Prolog → {Opacity[.25], PointSize[.002], Point[data2]},
 PlotRange → {0, 100}, ImageSize → 600]
```



From the graph we can see our model works pretty well in the fall and spring but doesn't capture the temperatre well either at the peak of summer or the depth of winter. We can try to remedy this somewhat by using a triangular wave instead of a sine wave - the key difference would be that a period of 1 for a triangle wave corresponds to $b = 1$ rather than $b = 2\pi$:

```
trianglewave = NonlinearModelFit[ data2,
   a * TriangleWave[b x + c] + d, {{a, 20}, {b, 1}, c, {d, 63.5}}, x] // Quiet
```

```
FittedModel[ 63.6378 + 25.1357 TriangleWave[0.70288 + 1.00053 x] ]
```

Again we can get the formula directly as well as some confidence intervals:

```
trianglewave[x]
```

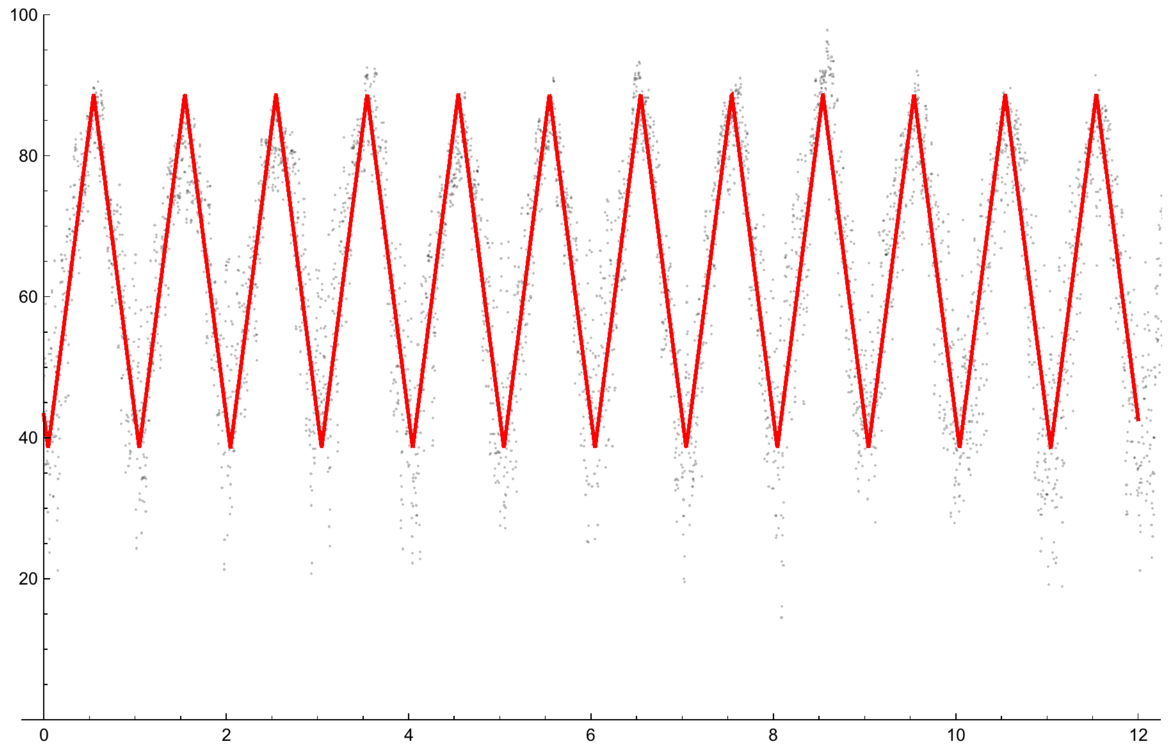
```
63.6378 + 25.1357 TriangleWave[0.70288 + 1.00053 x]
```

```
trianglewave["ParameterConfidenceIntervalTable"]
```

	Estimate	Standard Error	Confidence Interval
a	25.1357	0.204685	{24.7344, 25.5369}
b	1.00053	0.000324771	{0.999894, 1.00117}
c	0.70288	0.00235993	{0.698254, 0.707507}
d	63.6378	0.118923	{63.4046, 63.8709}

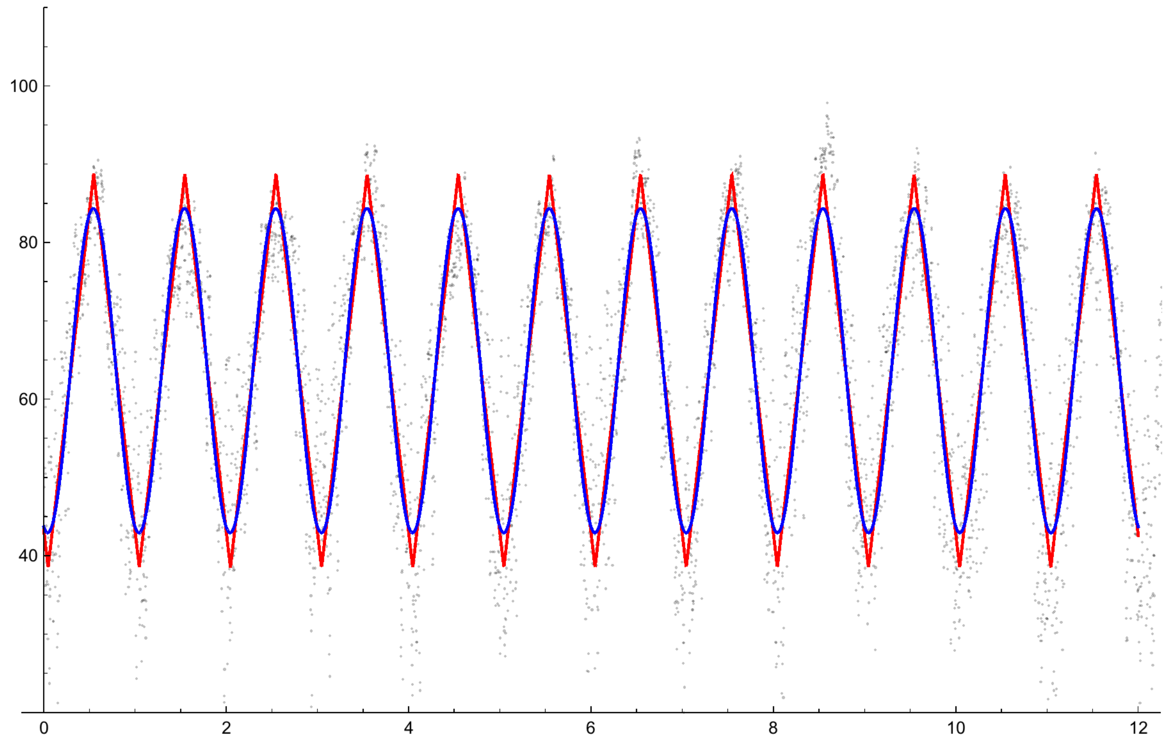
Graphing the triangle wave against the data looks like this:

```
Plot[ trianglewave[x], {x, 0, 12},  PlotStyle → Directive[Thick, Red],
 Prolog → {Opacity[.25], PointSize[.002], Point[data2]},
 PlotRange → {0, 100}, ImageSize → 600]
```



Viewing the two curves together we have:

```
Plot[{trianglewave[x], sinewave[x]}, {x, 0, 12},  PlotStyle → {Red, Blue},
 Prolog → {Opacity[.25], PointSize[.002], Point[data2]},
 ImageSize → 600, PlotRange → {20, 110}]
```



From the graphs we can see the triangle wave fits the peaks and valleys a bit better, but it is unclear which is the better overall fit. Formtunately the adjusted R^2 values are included as part of each FittedModel:

```
sinewave["AdjustedRSquared"]
```

```
0.985735
```

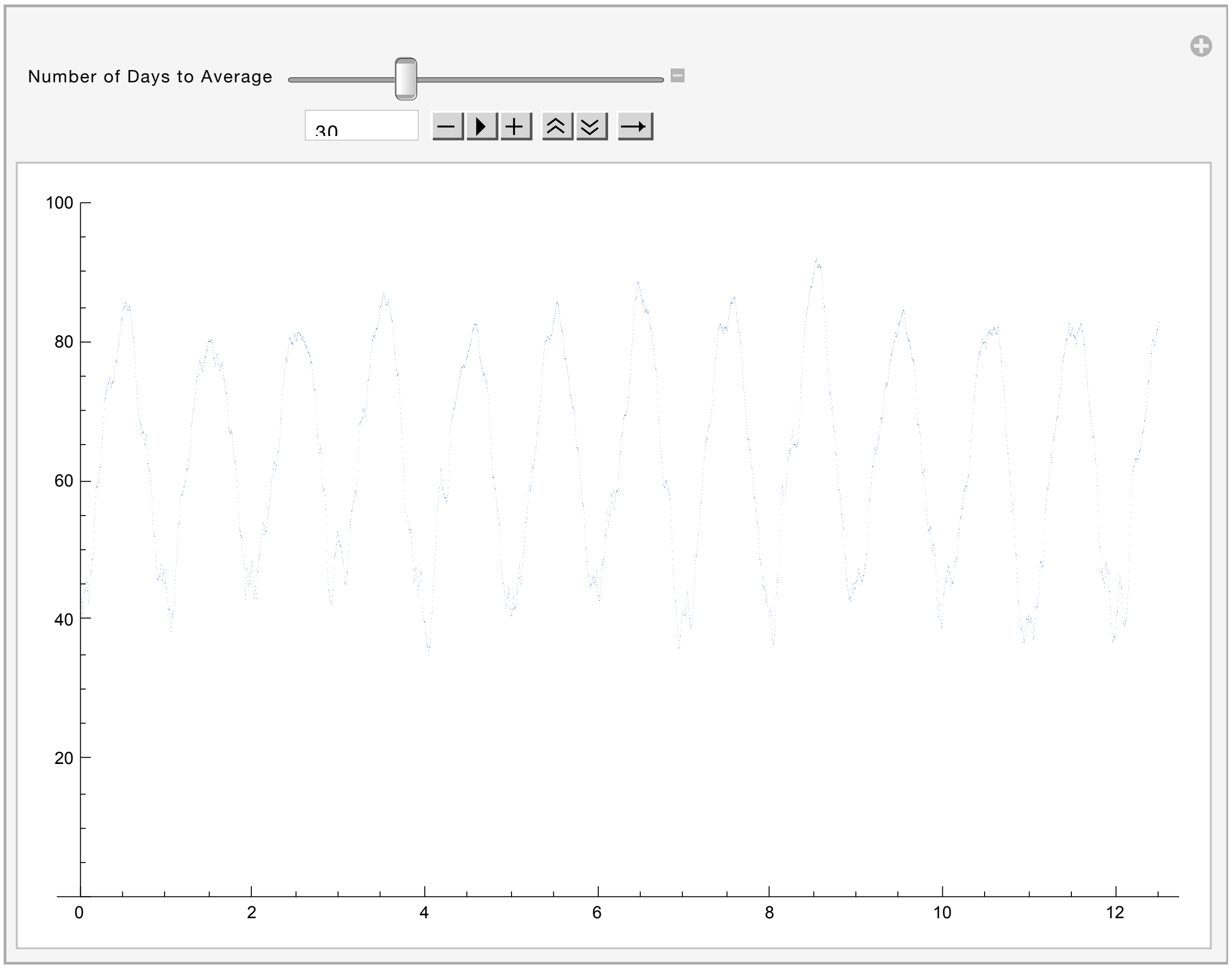
```
trianglewave["AdjustedRSquared"]
```

```
0.985132
```

So we can see that both curves do a good overall job, with the sine wave being just the tiniest bit better.

One of the things that makes the analysis of temperature data difficult is the data is very “noisy” - there is a lot of variation from day to day. Another type of data with this issue is stock prices, and one way that can be handled is to look at “moving averages” - looking at 10 day averages or 30 day averages rather than day-to-day averages. We can easily see what that idea looks like for our temperature data using a Manipulate command:

```
Manipulate[data3 = Table[
   {data2[[k, 1]], Mean[ data2[[k ;; k + j - 1, 2]]]}, {k, 1, Length[data2] - j + 1}];
 ListPlot[data3, ImageSize → 600, PlotRange → 100, PlotStyle → PointSize[Tiny]],
 {{j, 1, "Number of Days to Average"}, 1, 100, 1}, SaveDefinitions → True]
```



Author: Dr. Christopher Moretti
Revision Date: August 3, 2015