Keeling Analysis: non-linear regression

Andy Long
Spring, 2020

> Throughout the non-linear version of R^2 is useless. I propose a “better” version, which suggests that the non-linear model with variable amplitude has the highest adjusted R^2 of 0.998882 (which just barely beats the R^2 of the linear model: R^2 of 0.998863).
>
> That model suggests that the amplitude is, indeed, varying (based on the parameters of the linear function for amplitude being distinctly non-zero, per the confidence intervals or p-values.
>
> The period is, interestingly enough, not 1, based on the 95% confidence intervals (they do not contain 1 as a possibility).

Here’s one way you might Import the CO2 data: straight off the web.

In[546]:=

```
dat = Import["ftp://aftp.cmdl.noaa.gov/products/trends/co2/co2_mm_mlo.txt", "Table"];
DeleteCases[dat, {__String}, {-1}];
dat[[96, 1]]
dat[[71]]
(* First data line in the file: *)
Table[dat[[73, i]], {i, 1, 7}]
dim = Dimensions[dat][[1]]
DecimalDate = Table[dat[[i, 3]] - 1958, {i, 74, dim}] ;
CO2data = Table[dat[[i, 5]], {i, 74, dim}] ;
KeelingData = Transpose[{DecimalDate, CO2data}];
p1 = ListPlot[KeelingData]
```

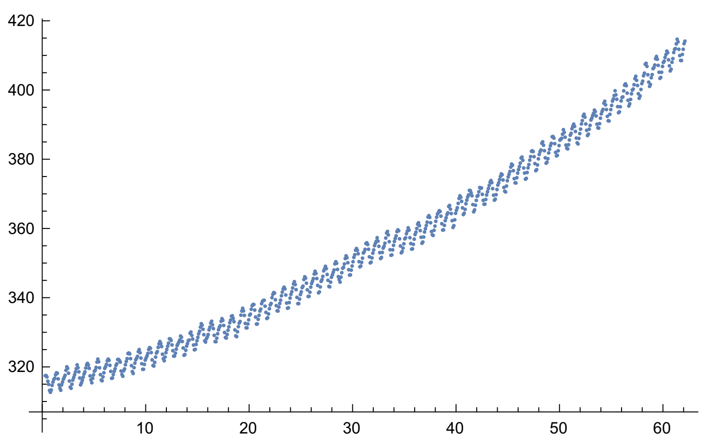
Out[548]= 1960

Out[549]= {#, decimal, average, interpolated, trend, #days}

Out[550]= {1958, 3, 1958.21, 315.71, 315.71, 314.62, -1}

Out[551]= 816

Out[555]=



We'll do some linear regression to get starting parameters:

In[556]:=
```
Clear[a, b, c]
lm = LinearModelFit[
  KeelingData, {x, x^2, Sin[2 Pi x], Cos[2 Pi x]}, x]

{astart, bstart, cstart, d, e} = lm["BestFitParameters"]
ampstart = Sqrt[d^2 + e^2]
x0start = -ArcTan[e / d] / (2 Pi)
lm["ParameterTable"]
lm["ANOVATable"]
lm["ParameterConfidenceIntervalTable"]
Show[p1, Plot[lm[x], {x, 0, Max[DecimalDate]}]]
```

Out[557]= FittedModel[$314.354 + 0.769266\,x + 0.0128702\,x^2 - 0.998694 \cos[2\pi x] + 2.64516 \sin[2\pi x]$]

Out[558]= {314.354, 0.769266, 0.0128702, 2.64516, -0.998694}

Out[559]= 2.82741

Out[560]= 0.0574564

General: Exp[-3456.16] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[-968.572] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[-1020.21] is too small to represent as a normalized machine number; precision may be lost.

General: Further output of General::munfl will be suppressed during this calculation.

Out[561]=

	Estimate	Standard Error	t-Statistic	P-Value
1	314.354	0.107553	2922.77	0.
x	0.769266	0.00795453	96.7079	0.
x^2	0.0128702	0.000123457	104.249	0.
Sin[2 π x]	2.64516	0.0499185	52.9895	9.60773×10^{-254}
Cos[2 π x]	-0.998694	0.0498633	-20.0286	1.34401×10^{-71}

General: Exp[-2497.] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[-1018.99] is too small to represent as a normalized machine number; precision may be lost.

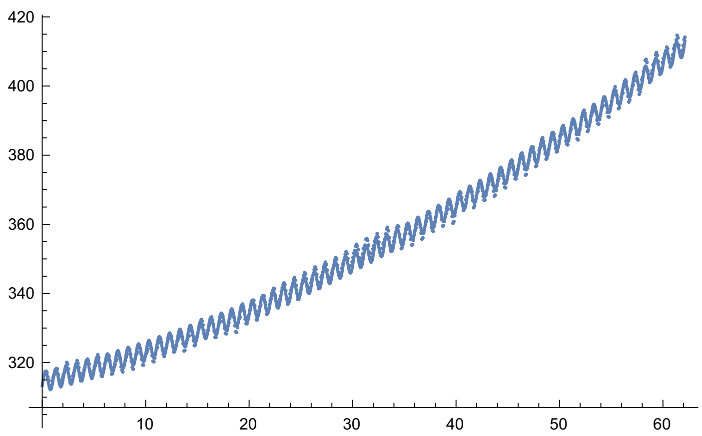
Out[562]=

	DF	SS	MS	F-Statistic	P-Value
x	1	586413.	586413.	634219.	0.
x^2	1	10013.2	10013.2	10829.5	0.
Sin[2 π x]	1	2597.5	2597.5	2809.26	8.33073×10^{-254}
Cos[2 π x]	1	370.908	370.908	401.146	1.34401×10^{-71}
Error	738	682.371	0.924622		
Total	742	600076.			

Out[563]=

	Estimate	Standard Error	Confidence Interval
1	314.354	0.107553	{314.143, 314.565}
x	0.769266	0.00795453	{0.75365, 0.784882}
x^2	0.0128702	0.000123457	{0.0126278, 0.0131126}
Sin[2 π x]	2.64516	0.0499185	{2.54716, 2.74316}
Cos[2 π x]	-0.998694	0.0498633	{-1.09658, -0.900803}

Out[564]=



Examining the RSquared as calculated by the non-linear regression function NonlinearModelFit, and that produced by LinearModelFit:

In[565]:=

```
SSTuncorrected = (CO2data).(CO2data)
fits = Table[lm[x], {x, DecimalDate}];
SSModel = (fits).(fits)
SSModel / SSTuncorrected
SSTotal = (CO2data - Mean[CO2data]).(CO2data - Mean[CO2data])
SSReg = (fits - Mean[CO2data]).(fits - Mean[CO2data])
SSReg / SSTotal
lm["RSquared"]
```

Out[565]= 9.42395×10^{7}

Out[567]= 9.42388×10^{7}

Out[568]= 0.999993

Out[569]= 600 076.

Out[570]= 599 394.

Out[571]= 0.998863

Out[572]= 0.998863

Now we'll do non-linear regression to get the parameters:

In[573]:=
```
Clear[a, b, c, amp, x0]
{astart, bstart, cstart, ampstart, x0start}
nlm = NonlinearModelFit[
   KeelingData,
   a + b x + c x^2 + amp Sin[2 Pi (x - x0) / Tval],
   {{a , astart}, {b, bstart}, {c, cstart},
    {amp, ampstart}, {x0, x0start}, {Tval, 1}},
   x]
{aend, bend, cend, ampend, x0end, Tvalend} == nlm["BestFitParameters"]
nlm["ParameterTable"]
nlm["ANOVATable"]
nlm["ParameterConfidenceIntervalTable"]
```

Out[574]= {314.354, 0.769266, 0.0128702, 2.82741, 0.0574564}

Out[575]= FittedModel[$314.346 + 0.769893\,x + 0.0128605\,x^2 + 2.83231\,\mathrm{Sin}[6.28659\,(-0.0748356 + x)]$]

Out[576]= {a → 257.331, b → 56.1589, c → 0.0163744, amp → 2.83231, x0 → 0.0578403, Tval → 0.999457} == {a → 314.346, b → 0.769893, c → 0.0128605, amp → 2.83231, x0 → 0.0748356, Tval → 0.999458}

General: Exp[−3457.18] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[−973.114] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[−1023.65] is too small to represent as a normalized machine number; precision may be lost.

General: Further output of General::munfl will be suppressed during this calculation.

Out[577]=

	Estimate	Standard Error	t-Statistic	P-Value
a	314.346	0.106794	2943.48	0.
b	0.769893	0.00789852	97.473	0.
c	0.0128605	0.000122588	104.908	0.
amp	2.83231	0.0495328	57.1804	3.55524×10^{-273}
x0	0.0748356	0.00559651	13.3718	1.14158×10^{-36}
Tval	0.999458	0.000155523	6426.41	0.

Out[578]=

	DF	SS	MS
Model	6	9.42388×10^{7}	1.57065×10^{7}
Error	737	671.546	0.911189
Uncorrected Total	743	9.42395×10^{7}	
Corrected Total	742	600076.	

Out[579]=

	Estimate	Standard Error	Confidence Interval
a	314.346	0.106794	{314.136, 314.556}
b	0.769893	0.00789852	{0.754386, 0.785399}
c	0.0128605	0.000122588	{0.0126198, 0.0131012}
amp	2.83231	0.0495328	{2.73507, 2.92955}
x0	0.0748356	0.00559651	{0.0638486, 0.0858226}
Tval	0.999458	0.000155523	{0.999153, 0.999763}

What we notice is that the Tval CI does not contain 1. Hence 1 is not an acceptable value (at 95% confidence).

Examining the RSquared as calculated by the non-linear regression function NonlinearModelFit:

In[580]:=
```
fits = Table[nlm[x], {x, DecimalDate}];
SSModel = (fits).(fits)
SSModel / SSTuncorrected
nlm["RSquared"]
nlm["AdjustedRSquared"]

(* This seems like the fairer comparison to the linear regression R^2.
   However it doesn't take into account the additional parameters
   (six for the nlm fit, and only five for the linear model):
  the more parameters you use, the better fit they should provide. *)
SSReg = (fits - Mean[CO2data]).(fits - Mean[CO2data])
ourSQ = SSReg / SSTotal

(* https://en.wikipedia.org/wiki/Coefficient_of_determination#Adjusted_R2
    Rsqadj=1-(1-Rsq)* (n-1)/(n-p-1)
*)
n = Length[KeelingData]
p = 6
Rsqadj = 1 - (1 - ourSQ) * (n - 1) / (n - p - 1)

ListPlot[Transpose[{DecimalDate, nlm["FitResiduals"]}]]
Sqrt[Mean[nlm["FitResiduals"]^2]]
```

Out[581]= 9.42388×10^7

Out[582]= 0.999993

Out[583]= 0.999993

Out[584]= 0.999993

Out[585]= 599 405.

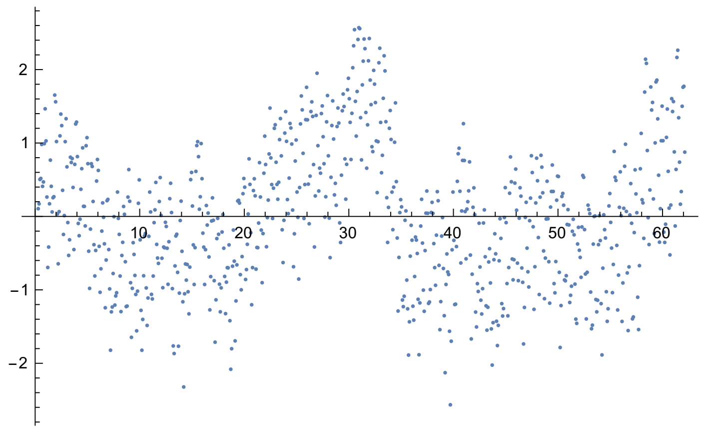
Out[586]= 0.998881

Out[587]= 743

Out[588]= 6

Out[589]= 0.998872

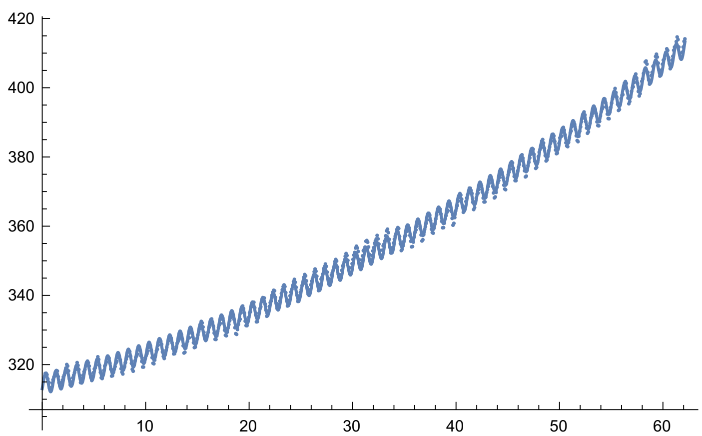
Out[590]=



Out[591]= 0.9507

In[592]:=
```
Show[p1, Plot[nlm[x], {x, 0, Max[DecimalDate]}]]
```

Out[592]=



Not much change. Now let’s allow the amplitude of the sinusoidal variation be a linear function, and see if it’s significantly changing over time:

In[593]:=
```
Clear[a, b, c, amp, x0, m]
{astart, bstart, cstart, ampstart, x0start}
nlmFinal = NonlinearModelFit[
  KeelingData,
  a + b x + c x ^ 2 + (amp + m x) Sin[2 Pi (x - x0) / Tval],
  {{a , astart}, {b, bstart}, {c, cstart},
   {amp, ampstart}, {x0, x0start}, {m, 0}, {Tval, 1}},
  x]
{aend, bend, cend, ampend, x0end, mend, Tvalend} = nlmFinal["BestFitParameters"]
nlmFinal["ParameterTable"]
nlmFinal["ANOVATable"]
nlmFinal["ParameterConfidenceIntervalTable"]
```

Out[594]= {314.354, 0.769266, 0.0128702, 2.82741, 0.0574564}

Out[595]= FittedModel[$314.35 + 0.769565\,x + 0.012866\,x^2 + (2.59801 + 0.007526\,x)\,\mathrm{Sin}[6.2866\,(-0.0748921 + x)]$]

Out[596]= {a → 314.35, b → 0.769565, c → 0.012866,
 amp → 2.59801, x0 → 0.0748921, m → 0.007526, Tval → 0.999457}

General: Exp[−3456.15] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[−974.866] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[−1025.94] is too small to represent as a normalized machine number; precision may be lost.

General: Further output of General::munfl will be suppressed during this calculation.

Out[597]=

	Estimate	Standard Error	t-Statistic	P-Value
a	314.35	0.106339	2956.12	0.
b	0.769565	0.00786512	97.8453	0.
c	0.012866	0.000122073	105.396	0.
amp	2.59801	0.0990808	26.2211	1.5704×10^{-107}
x0	0.0748921	0.00580865	12.8932	1.9346×10^{-34}
m	0.007526	0.00276033	2.72649	0.00655357
Tval	0.999457	0.00015517	6441.05	0.

Out[598]=

	DF	SS	MS
Model	7	9.42388×10^7	1.34627×10^7
Error	736	664.832	0.903304
Uncorrected Total	743	9.42395×10^7	
Corrected Total	742	600076.	

Out[599]=

	Estimate	Standard Error	Confidence Interval
a	314.35	0.106339	{314.141, 314.559}
b	0.769565	0.00786512	{0.754124, 0.785006}
c	0.012866	0.000122073	{0.0126263, 0.0131056}
amp	2.59801	0.0990808	{2.40349, 2.79252}
x0	0.0748921	0.00580865	{0.0634886, 0.0862956}
m	0.007526	0.00276033	{0.00210694, 0.0129451}
Tval	0.999457	0.00015517	{0.999152, 0.999761}

Examining the RSquared as calculated by the non-linear regression function NonlinearModelFit:

In[600]:=
```
fits = Table[nlmFinal[x], {x, DecimalDate}];
SSModel = (fits).(fits)
SSModel / SSTuncorrected
nlmFinal["RSquared"]
nlmFinal["AdjustedRSquared"]

(* This seems like the fairer comparison to the linear regression R^2.
  However it doesn't take into account the additional parameters
  (seven, versus six for the nlm fit, and only five for the linear model):
 the more parameters you use, the better fit they should provide. *)
SSReg = (fits - Mean[CO2data]).(fits - Mean[CO2data])
ourSQ = SSReg / SSTotal
(* https://en.wikipedia.org/wiki/Coefficient_of_determination#Adjusted_R2
   Rsqadj=1-(1-Rsq)* (n-1)/(n-p-1)
*)
n = Length[KeelingData]
p = 7
Rsqadj = 1 - (1 - ourSQ) * (n - 1) / (n - p - 1)
```

Out[601]= 9.42388×10^{7}

Out[602]= 0.999993

Out[603]= 0.999993

Out[604]= 0.999993

Out[605]= 599 412.

Out[606]= 0.998892

Out[607]= 743

Out[608]= 7

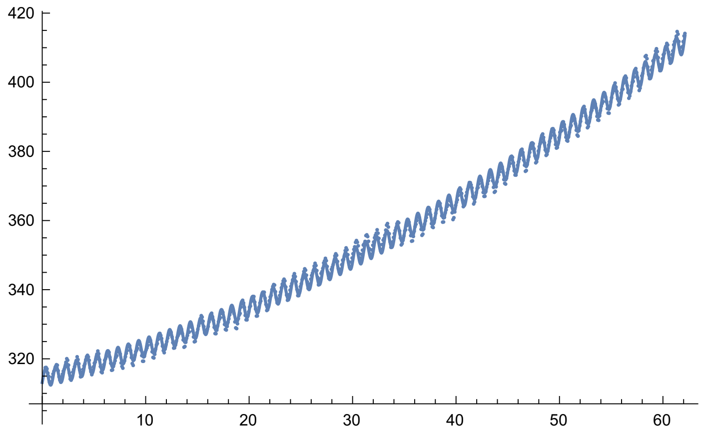
Out[609]= 0.998882

In[610]:=
```
Show[p1, Plot[nlmFinal[x], {x, 0, Max[DecimalDate]}]]

phase = x0 /. x0end
phaseInDaysFromNewYear = phase * 365.25
periodFinal = 1 + period /. periodend
periodFinal = periodFinal * 365.25

ListPlot[Transpose[{DecimalDate, nlmFinal["FitResiduals"]}]]
Sqrt[Mean[nlmFinal["FitResiduals"] ^ 2]]
```

Out[610]=



Out[611]= 0.0748921

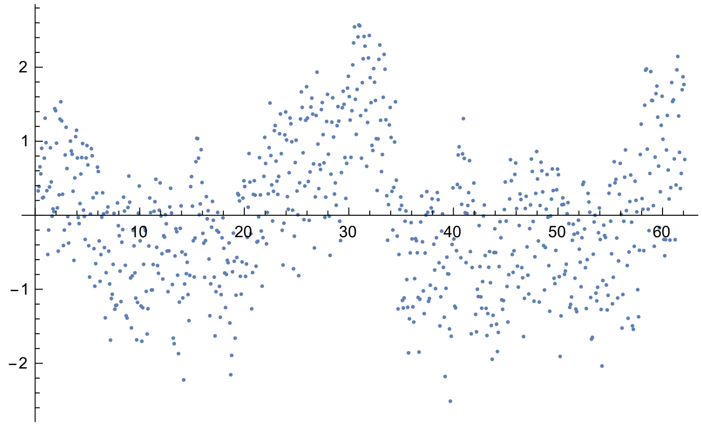
Out[612]= 27.3543

ReplaceAll: {periodend} is neither a list of replacement rules nor a valid dispatch table, and so cannot be used for replacing.

Out[613]= 1 + period /. periodend

Out[614]= 365.25 (1 + period /. periodend)

Out[615]=

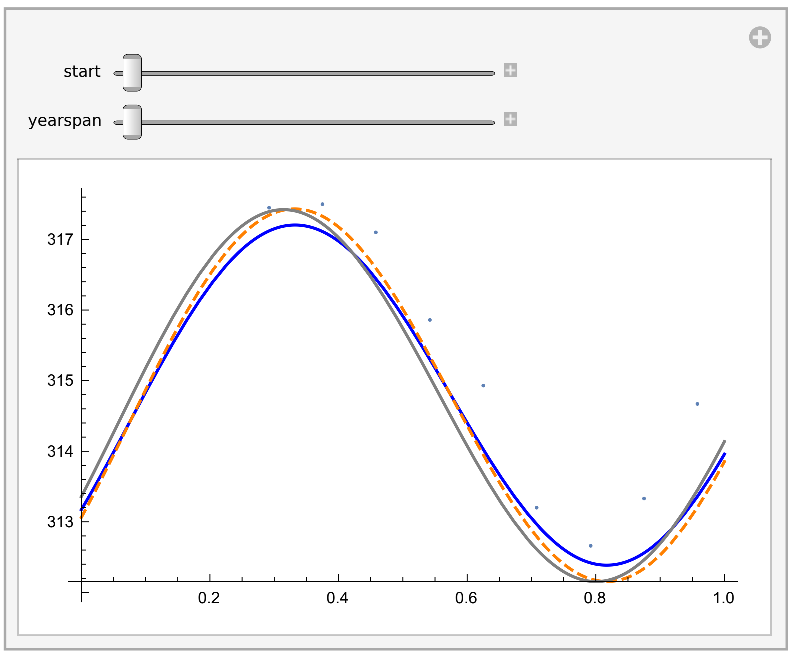


Out[616]= 0.945935

Let’s compare the models: A manipulate command let’s us see how the models are doing in their fit to the data:

In[617]:=
```
Manipulate[
 Module[{},
  Show[
   Plot[
    {nlmFinal[x], nlm[x], lm[x]},
    {x, start, start + yearspan},
    PlotStyle → {Blue, {Orange, Dashed}, Gray}]
   , p1]
 ],
 {start, 0, Max[DecimalDate]},
 {yearspan, 1, 10}
]
```

Out[617]=



I didn’t ask you for the exponential model, but Flerlage, et al. did a nice job in their report, and concluded that this was the best model:

"Mathematica found that the best power model fit for the data is fit by the nonlinear model: CO2=257.184 +56.2922 e^ (0.0163503 * (years since 1958)) + 2.83537 * sin(2π ((years since 1958)-1.05748)])"

In[618]:=
```
Clear[a, b, c, amp, x0]
{astart, bstart, cstart, ampstart, x0start} =
 {257.184, 56.2922, 0.0163503, 2.83537, .05748}
nlmExpo = NonlinearModelFit[
  KeelingData,
  a + b Exp[c x] + amp Sin[2 Pi (x - x0)],
  {{a , astart}, {b, bstart}, {c, cstart},
   {amp, ampstart}, {x0, x0start}},
  x]
{aend, bend, cend, ampend, x0end} = nlmExpo["BestFitParameters"]
nlmExpo["ParameterTable"]
nlmExpo["ANOVATable"]
nlmExpo["ParameterConfidenceIntervalTable"]
```

Out[619]= {257.184, 56.2922, 0.0163503, 2.83537, 0.05748}

Out[620]= FittedModel[$257.331 + 56.1589\, e^{0.0163744\, x} + 2.83231\, \mathrm{Sin}[2\, \pi\, (-0.0578403 + x)]$]

Out[621]= {a → 257.331, b → 56.1589, c → 0.0163744, amp → 2.83231, x0 → 0.0578403}

General: Exp[−1707.61] is too small to represent as a normalized machine number; precision may be lost.

General: Exp[−1008.42] is too small to represent as a normalized machine number; precision may be lost.

Out[622]=

	Estimate	Standard Error	t-Statistic	P-Value
a	257.331	0.945811	272.074	0.
b	56.1589	0.867885	64.7078	2.06302×10^{-306}
c	0.0163744	0.000159775	102.484	0.
amp	2.83231	0.049994	56.6529	6.0335×10^{-271}
x0	0.0578403	0.00280815	20.5973	8.00031×10^{-75}

Out[623]=

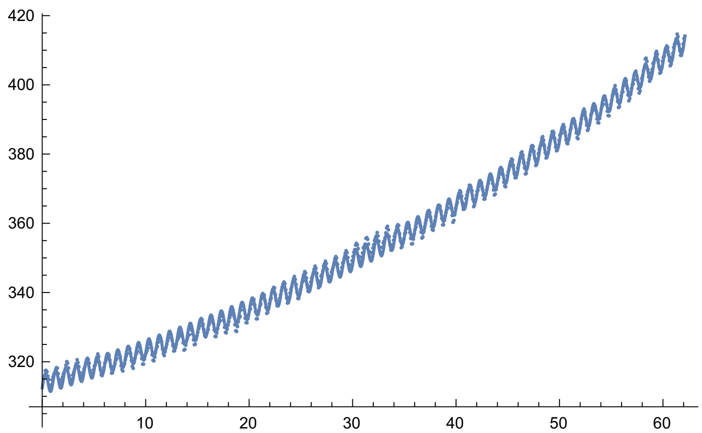
	DF	SS	MS
Model	5	9.42388×10^{7}	1.88478×10^{7}
Error	738	684.912	0.928065
Uncorrected Total	743	9.42395×10^{7}	
Corrected Total	742	600076.	

Out[624]=

	Estimate	Standard Error	Confidence Interval
a	257.331	0.945811	{255.474, 259.188}
b	56.1589	0.867885	{54.4551, 57.8627}
c	0.0163744	0.000159775	{0.0160608, 0.0166881}
amp	2.83231	0.049994	{2.73416, 2.93045}
x0	0.0578403	0.00280815	{0.0523274, 0.0633532}

In[625]:=
```
Show[p1, Plot[nlmExpo[x], {x, 0, Max[DecimalDate]}]]
phase = x0 /. x0end
phaseInDaysFromNewYear = phase * 365.25
```

Out[625]=



Out[626]= 0.0578403

Out[627]= 21.1262

Examining the RSquared as calculated by the non-linear regression function NonlinearModelFit:

In[628]:=
```
fits = Table[nlmExpo[x], {x, DecimalDate}];
SSModel = (fits).(fits)
SSModel / SSTuncorrected
nlmExpo["RSquared"]
(* This seems like the fairer comparison to the linear regression R^2.
 However it doesn't take into account the additional parameters:
  the more parameters you use, the better fit they should provide. *)
SSReg = (fits - Mean[CO2data]).(fits - Mean[CO2data])
ourSQ = SSReg / SSTotal
(* https://en.wikipedia.org/wiki/Coefficient_of_determination#Adjusted_R2
   Rsqadj=1-(1-Rsq)* (n-1)/(n-p-1)
*)
n = Length[KeelingData]
p = 5
Rsqadj = 1 - (1 - ourSQ) * (n - 1) / (n - p - 1)
```

Out[629]= 9.42388×10^{7}

Out[630]= 0.999993

Out[631]= 0.999993

Out[632]= 599 392.

Out[633]= 0.998859

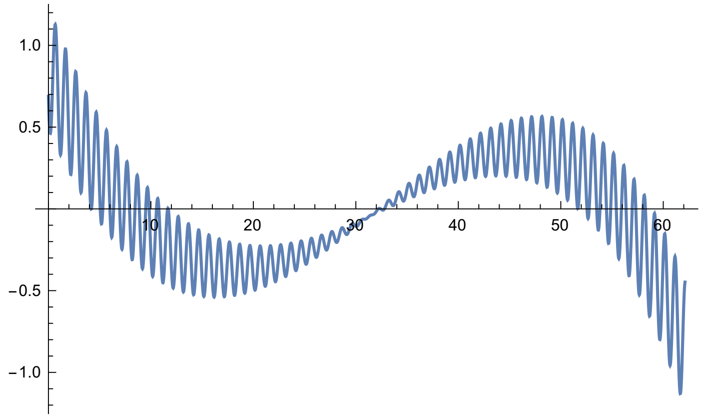
Out[634]= 743

Out[635]= 5

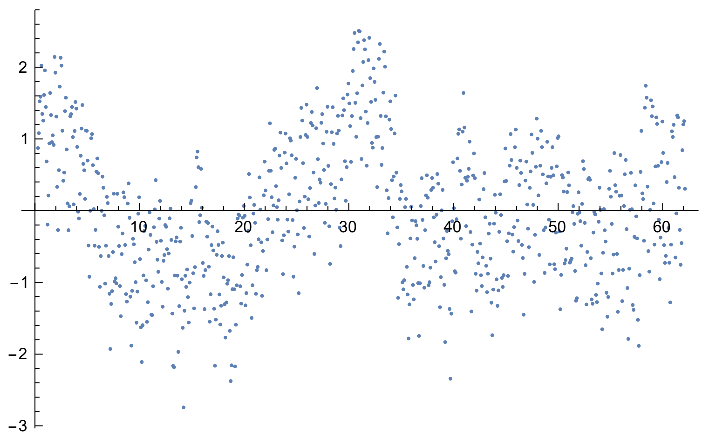
Out[636]= 0.998851

In[637]:=
```
Plot[nlmFinal[x] - nlmExpo[x], {x, 0, Max[DecimalDate]}]
ListPlot[Transpose[{DecimalDate, nlmExpo["FitResiduals"]}]]
```

Out[637]=



Out[638]=



In[639]:=
```
root = FindRoot[nlmExpo[x] == 450, {x, 60}]
1958 + x /. root
root = FindRoot[nlmFinal[x] == 450, {x, 60}]
1958 + x /. root
root = FindRoot[nlm[x] == 450, {x, 60}]
1958 + x /. root
root = FindRoot[lm[x] == 450, {x, 60}]
1958 + x /. root
```

Out[639]= {x → 75.618}

Out[640]= 2033.62

Out[641]= {x → 76.4477}

Out[642]= 2034.45

Out[643]= {x → 76.4324}

Out[644]= 2034.43

Out[645]= {x → 77.6616}

Out[646]= 2035.66