Checking out the BG Climate Normals

Andy Long

Spring, 2020

- Let's load in the data, and see what we've got. These were obtained from NOAA, and I stashed a copy so that I would have some control over it:

In[1049]:=
```
BGNormals = Import[
   "http://ceadserv1.nku.edu/longa//classes/mat375/days/docs/Fletcher/data/
     BowlingGreenDailyClimateNormals.csv", "CSV"];
BGheader = BGNormals[[1]]
BGNormalData = BGNormals[[2 ;;]];
```

Out[1050]=
```
{STATION, STATION_NAME, ELEVATION, LATITUDE, LONGITUDE, DATE,
 DLY-GRDD-BASE40, Completeness Flag, DLY-GRDD-BASE45, Completeness Flag,
 DLY-GRDD-BASE50, Completeness Flag, DLY-GRDD-BASE55, Completeness Flag,
 DLY-GRDD-BASE57, Completeness Flag, DLY-GRDD-BASE60, Completeness Flag,
 DLY-GRDD-BASE65, Completeness Flag, DLY-GRDD-BASE70, Completeness Flag,
 DLY-GRDD-BASE72, Completeness Flag, DLY-GRDD-TB4886, Completeness Flag,
 DLY-GRDD-TB5086, Completeness Flag, DLY-CLDD-BASE45, Completeness Flag,
 DLY-CLDD-BASE50, Completeness Flag, DLY-CLDD-BASE55, Completeness Flag,
 DLY-CLDD-BASE57, Completeness Flag, DLY-CLDD-BASE60, Completeness Flag,
 DLY-CLDD-NORMAL, Completeness Flag, DLY-CLDD-BASE70, Completeness Flag,
 DLY-CLDD-BASE72, Completeness Flag, DLY-HTDD-BASE40, Completeness Flag,
 DLY-HTDD-BASE45, Completeness Flag, DLY-HTDD-BASE50, Completeness Flag,
 DLY-HTDD-BASE55, Completeness Flag, DLY-HTDD-BASE57, Completeness Flag,
 DLY-HTDD-BASE60, Completeness Flag, DLY-HTDD-NORMAL, Completeness Flag,
 DLY-TAVG-NORMAL, Completeness Flag, DLY-DUTR-NORMAL, Completeness Flag,
 DLY-TMAX-NORMAL, Completeness Flag, DLY-TMIN-NORMAL, Completeness Flag,
 DLY-TAVG-STDDEV, Completeness Flag, DLY-DUTR-STDDEV, Completeness Flag,
 DLY-TMAX-STDDEV, Completeness Flag, DLY-TMIN-STDDEV, Completeness Flag}
```

Now I want to pick off the important variables: max, min,

and dtr.

In[1052]:=
```
(* Dumb counting method...:) *)
navg = 13 * 4 + 6 + 1;
ndtr = navg + 2;
nmax = navg + 4;
nmin = navg + 6;
navgstddev = navg + 8;
ndtrstddev = navg + 10;
nmaxstddev = navg + 12;
nminstddev = navg + 14;
indices = {navg, ndtr, nmax, nmin, navgstddev, ndtrstddev, nmaxstddev, nminstddev};
vnames = BGheader[[indices]]
```

Out[1061]=
```
{DLY-TAVG-NORMAL, DLY-DUTR-NORMAL, DLY-TMAX-NORMAL, DLY-TMIN-NORMAL,
 DLY-TAVG-STDDEV, DLY-DUTR-STDDEV, DLY-TMAX-STDDEV, DLY-TMIN-STDDEV}
```

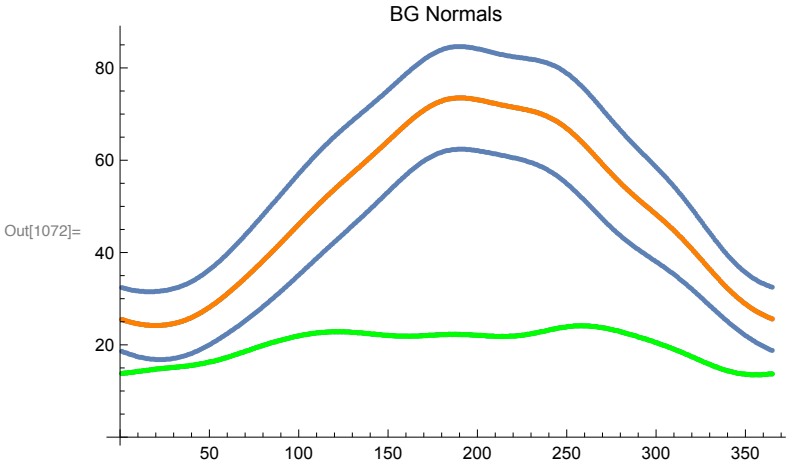
In[1062]:=
```
BGNormalTemps = BGNormalData[[All, indices]];
BGNormalTemps[[1, All]]
{avg, dtr, max, min, avgstddev, dtrstddev, maxstddev, minstddev} =
  Transpose[BGNormalTemps];
```

Out[1063]=
```
{25.5, 13.8, 32.4, 18.6, 11, 6.4, 11.2, 11.9}
```

Averages and DTR, presented as data in the data set, are actually just derived from max and min normals, as these commands show:

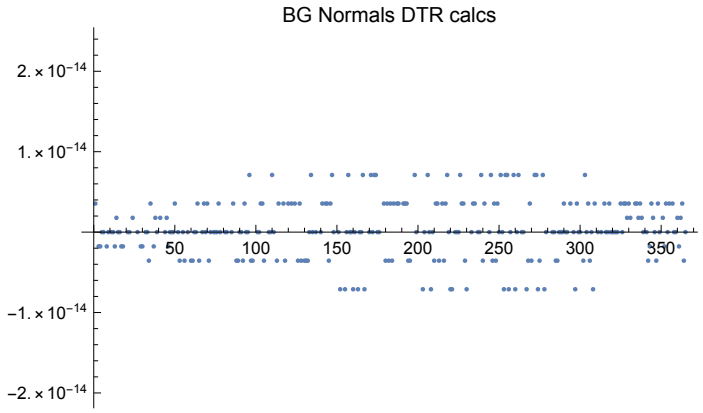
In[1065]:=
```
days = Table[i, {i, 1, 365}];
p1 = ListPlot[Transpose[{days, min}], PlotLabel → "BG Normals"];
p2 = ListPlot[Transpose[{days, avg}], PlotLabel → "BG Normal Mean"];
p3 = ListPlot[Transpose[{days, max}], PlotLabel → "BG Normal Mean"];
p4 = ListPlot[Transpose[{days, dtr}], PlotLabel → "BG Normals"];
p5 = ListPlot[Transpose[{days, max - min}],
   PlotLabel → "BG Normals", PlotStyle → Green];
p6 = ListPlot[Transpose[{days, (max + min) / 2}],
   PlotLabel → "BG Normals", PlotStyle → Orange];
Show[p1, p2, p3, p4, p5, p6, PlotRange → All]
```

Out[1072]=

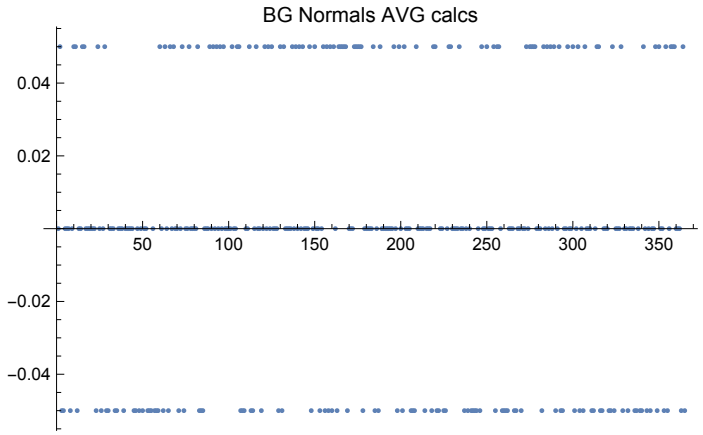


In[1073]:=
```
ListPlot[Transpose[{days, dtr - (max - min)}], PlotLabel → "BG Normals DTR calcs"]
ListPlot[Transpose[{days, avg - (max + min) / 2}], PlotLabel → "BG Normals AVG calcs"]
```

Out[1073]=



Out[1074]=



- Now let's do some regression:

Regressions for Minimum Normals

We can do the regressions either as linear regressions, or as non-linear regressions -- and we'll get exactly the same results! It's because of the magic of the trig identity, which allows us to break the Sin[2π(t-t0)] term into Sine and Cosine terms.

If we do them as linear regressions, the R-squared terms are more useful.

This first regression just fits
a + b Cos[2 Pi x] + c Sin[2Pi x]

In[1075]:=
```
pointOfYear = days / 365;
data = Transpose[{pointOfYear, min}];
minplot = ListPlot[data, PlotLabel → "Minimum -- BG Normals"];
minlm = LinearModelFit[Cases[data, {_?NumericQ..}], {Cos[2 Pi x], Sin[2 Pi x]}, x]
minfits = minlm["FitResiduals"];
minlm["ParameterTable"]
minlm["RSquared"]
minlm["AdjustedRSquared"]
minlm["ParameterConfidenceIntervals"]
Show[minplot, Plot[minlm[x], {x, 0, 1}]]
Histogram[minfits]
```

Out[1078]= FittedModel[39.7551 − 20.8182 Cos[2 π x] − 7.81826 Sin[2 π x]]

Out[1080]=

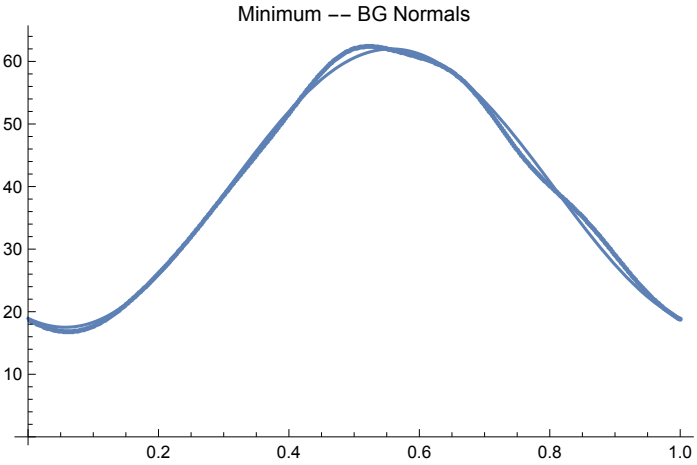
	Estimate	Standard Error	t-Statistic	P-Value
1	39.7551	0.0429371	925.891	0.
Cos[2 π x]	−20.8182	0.0607222	−342.844	0.
Sin[2 π x]	−7.81826	0.0607222	−128.755	2.09402×10^{-304}

Out[1081]= 0.997308

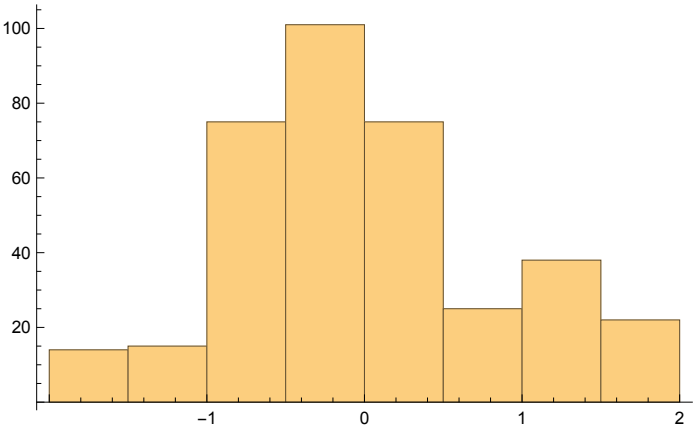
Out[1082]= 0.997293

Out[1083]= {{39.6706, 39.8395}, {−20.9376, −20.6988}, {−7.93768, −7.69885}}

Out[1084]=



Out[1085]=



This second regression adds a linear term (d x):
a + d x + b Cos[2 Pi x] + c Sin[2Pi x]
and we discover that the d is significantly different from 0, and positive -- 1.9, which, on an annual basis would represent quite an amazing increase in minimum temperature!

In[1086]:=
```
minlm = LinearModelFit[Cases[data, {_?NumericQ..}], {x, Cos[2 Pi x], Sin[2 Pi x]}, x]
minfits = minlm["FitResiduals"];
minlm["ParameterTable"]
minlm["RSquared"]
minlm["AdjustedRSquared"]
minlm["ParameterConfidenceIntervals"]
Show[minplot, Plot[minlm[x], {x, 0, 1}]]
Histogram[minfits]
```

Out[1086]= FittedModel[38.8011 + 1.90273 x − 20.8234 Cos[2 πx] − 7.21262 Sin[2 πx]]

Out[1088]=

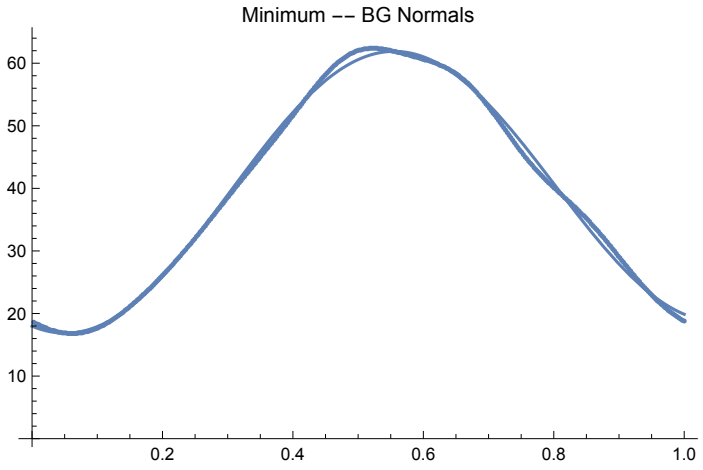
	Estimate	Standard Error	t-Statistic	P-Value
1	38.8011	0.114996	337.412	0.
x	1.90273	0.21577	8.81833	5.02075×10^{-17}
Cos[2 πx]	−20.8234	0.0551584	−377.521	0.
Sin[2 πx]	−7.21262	0.0880855	−81.8821	3.10506×10^{-235}

Out[1089]= 0.997785

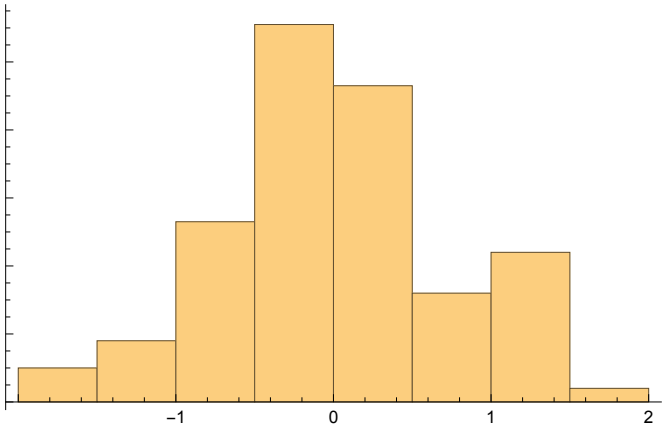
Out[1090]= 0.997767

Out[1091]= {{38.5749, 39.0272}, {1.47841, 2.32706}, {−20.9319, −20.715}, {−7.38585, −7.0394}}

Out[1092]=



Out[1093]=



Now I'll do the regression using non-linear regression. In a way, it's cleaner: I don't need to deduce the amplitude and phase shift -- I'm staring at them! The results are identical, however -- and that's one of the cool things.

Again, the first model is -- well, there it is in the command for NonlinearModelFit! I like the way we write the regression model with NonlinearModelFit.

In[1094]:=
```
minlm = NonlinearModelFit[data, {a + amp Sin[2 Pi (x - x0)]}, {a, amp, {x0, .3}}, x]
minfits = minlm["FitResiduals"];
minlm["ParameterTable"]
minlm["RSquared"]
minlm["AdjustedRSquared"]
minlm["ParameterConfidenceIntervals"]
Show[minplot, Plot[minlm[x], {x, 0, 1}]]
Histogram[minfits]
```

Out[1094]=
```
FittedModel[ 39.7551 + 22.2379 Sin[2 π (-0.307177 + x)] ]
```

Out[1096]=

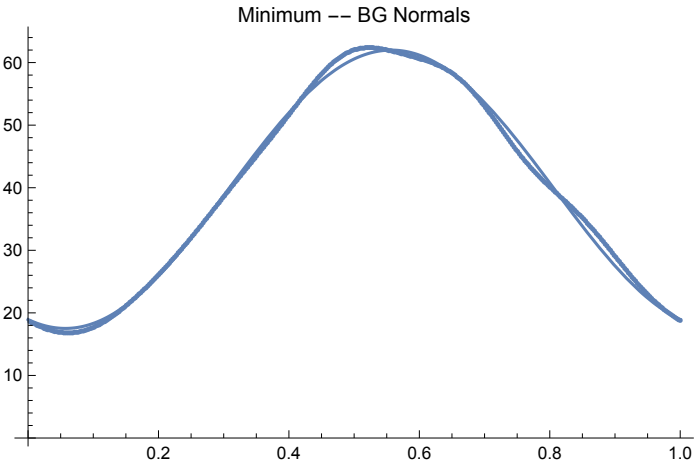
	Estimate	Standard Error	t-Statistic	P-Value
a	39.7551	0.0429371	925.891	0.
amp	22.2379	0.0607222	366.224	0.
x0	0.307177	0.000434584	706.829	0.

Out[1097]=
```
0.999635
```

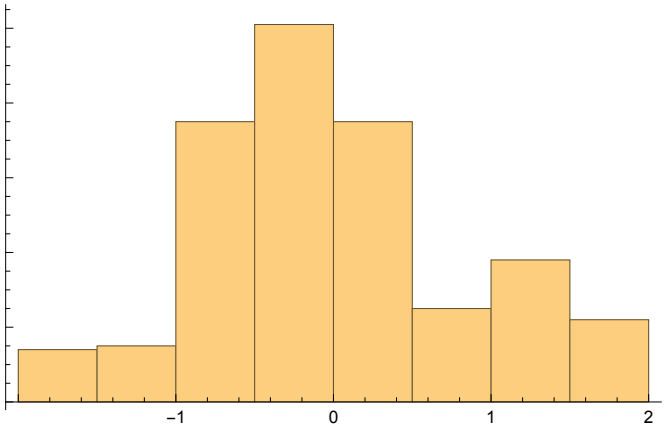
Out[1098]=
```
0.999632
```

Out[1099]=
```
{{39.6706, 39.8395}, {22.1185, 22.3573}, {0.306322, 0.308031}}
```

Out[1100]=



Out[1101]=



Our model has an amplitude of 22.24 (a total variation of twice that), and the curve hits its mean at .3063 of the year; that means it bottoms out at .3063-.25 of the year, or .0563*365.25 -- about the 21st of January.

As before, we now add the linear term, only doing so with NonlinearModelFit.

In[1102]:=
```
minlm = NonlinearModelFit[data,
  {a + b x + amp Sin[2 Pi (x - x0)]}, {a, b, amp, {x0, .3}}, x]
minfits = minlm["FitResiduals"];
minlm["ParameterTable"]
minlm["RSquared"]
minlm["AdjustedRSquared"]
minlm["ParameterConfidenceIntervals"]
Show[minplot, Plot[minlm[x], {x, 0, 1}]]
Histogram[minfits]
```

Out[1102]= FittedModel[38.8011 + 1.90273 x + 22.0372 Sin[2 π (−0.303068 + x)]]

Out[1104]=

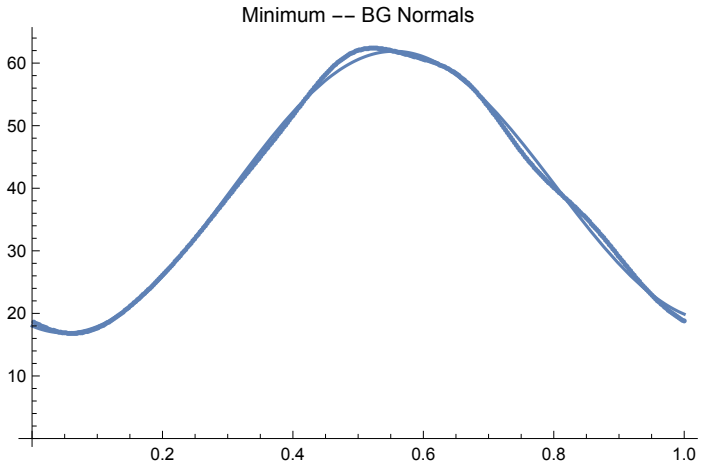
	Estimate	Standard Error	t-Statistic	P-Value
a	38.8011	0.114996	337.412	0.
b	1.90273	0.21577	8.81833	5.02075×10^{-17}
amp	22.0372	0.0593514	371.301	0.
x0	0.303068	0.000616165	491.862	0.

Out[1105]= 0.9997

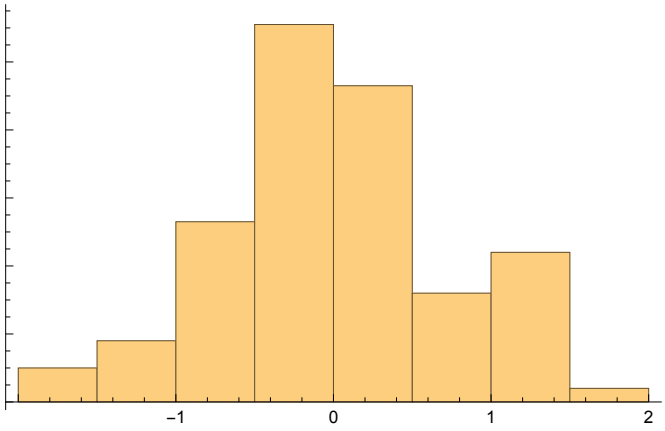
Out[1106]= 0.999696

Out[1107]= {{38.5749, 39.0272}, {1.47841, 2.32706}, {21.9205, 22.1539}, {0.301856, 0.30428}}

Out[1108]=



Out[1109]=



Regressions for Maximum Normals

I'm now going to reproduce the same type of results for the maxima, rather than the minima.

In[1110]:=
```
data = Transpose[{pointOfYear, max}];
maxplot = ListPlot[data, PlotLabel → "Maximum -- BG Normals"];
maxlm = LinearModelFit[data, {Cos[2 Pi x], Sin[2 Pi x]}, x]
maxfits = maxlm["FitResiduals"];
maxlm["ParameterTable"]
maxlm["RSquared"]
maxlm["AdjustedRSquared"]
maxlm["ParameterConfidenceIntervals"]
Show[maxplot, Plot[maxlm[x], {x, 0, 1}]]
Histogram[maxfits]
```

Out[1112]=
$$\text{FittedModel}\left[59.6011 - 24.9441\,\text{Cos}[2\,\pi x] - 8.63393\,\text{Sin}[2\,\pi x]\right]$$

Out[1114]=

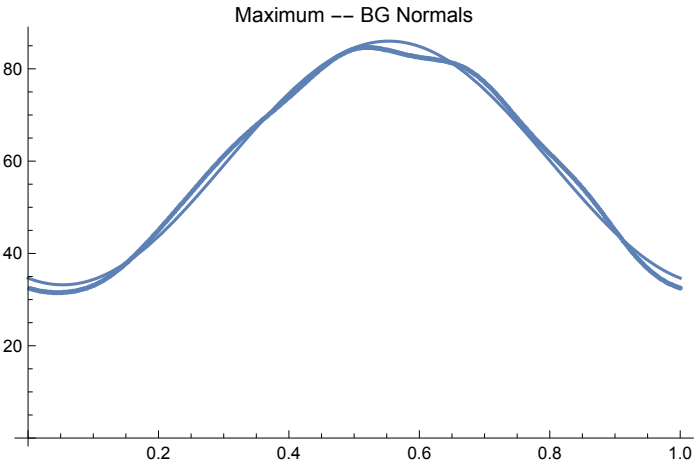
	Estimate	Standard Error	t-Statistic	P-Value
1	59.6011	0.0790277	754.18	0.
Cos[2 π x]	−24.9441	0.111762	−223.189	0.
Sin[2 π x]	−8.63393	0.111762	−77.2527	5.09779×10^{-227}

Out[1115]= `0.993552`

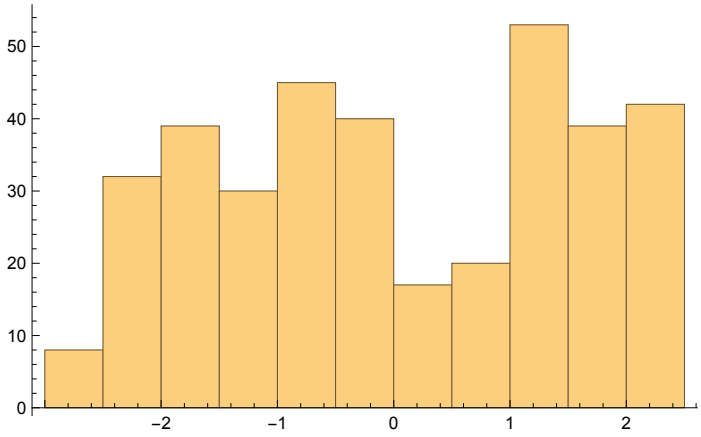
Out[1116]= `0.993517`

Out[1117]= `{{59.4457, 59.7565}, {-25.1639, -24.7243}, {-8.85371, -8.41414}}`

Out[1118]=



Out[1119]=



In[1120]:=
```
maxlm = LinearModelFit[data, {x, Cos[2 Pi x], Sin[2 Pi x]}, x]
maxfits = maxlm["FitResiduals"];
maxlm["ParameterTable"]
maxlm["RSquared"]
maxlm["AdjustedRSquared"]
maxlm["ParameterConfidenceIntervals"]
Show[maxplot, Plot[maxlm[x], {x, 0, 1}]]
Histogram[maxfits]
```

Out[1120]= FittedModel[58.9558 + 1.28703 x − 24.9476 Cos[2 π x] − 8.22426 Sin[2 π x]]

Out[1122]=

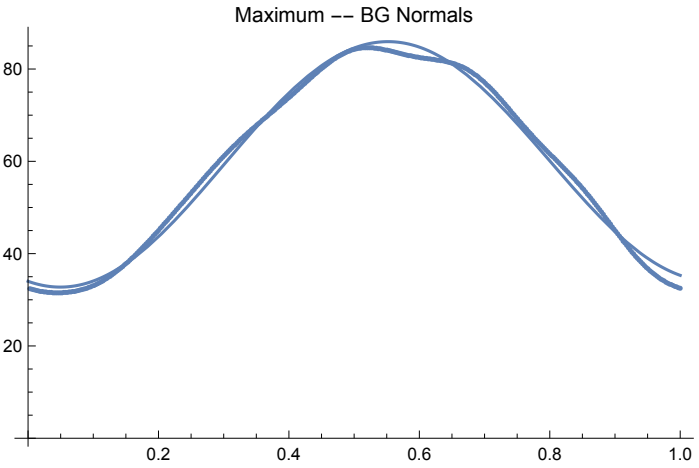
	Estimate	Standard Error	t-Statistic	P-Value
1	58.9558	0.230532	255.739	0.
x	1.28703	0.432552	2.97543	0.00312278
Cos[2 π x]	−24.9476	0.110576	−225.616	0.
Sin[2 π x]	−8.22426	0.176584	−46.5742	1.04268×10^{-154}

Out[1123]= 0.993707

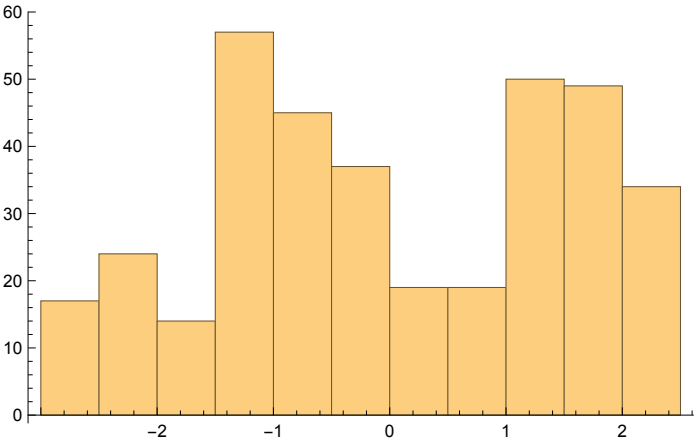
Out[1124]= 0.993654

Out[1125]= {{58.5025, 59.4092}, {0.436392, 2.13767}, {−25.1651, −24.7302}, {−8.57152, −7.877}}

Out[1126]=



Out[1127]=



So we notice that the linear term is 1.28 or so; it's significantly different from

zero. Maxima are increasing over the course of the year.

But not nearly as much as the minima, for which it was almost 2: minima were increasing twice as fast as maxima, according to these models.

In[1128]:=
```
maxlm = NonlinearModelFit[data, {a + amp Sin[2 Pi (x - x0)]}, {a, amp, {x0, .3}}, x]
maxfits = maxlm["FitResiduals"];
maxlm["ParameterTable"]
maxlm["RSquared"]
maxlm["AdjustedRSquared"]
maxlm["ParameterConfidenceIntervals"]
Show[maxplot, Plot[maxlm[x], {x, 0, 1}]]
Histogram[maxfits]
```

Out[1128]= FittedModel[59.6011 + 26.3961 Sin[2 π (−0.303034 + x)]]

Out[1130]=

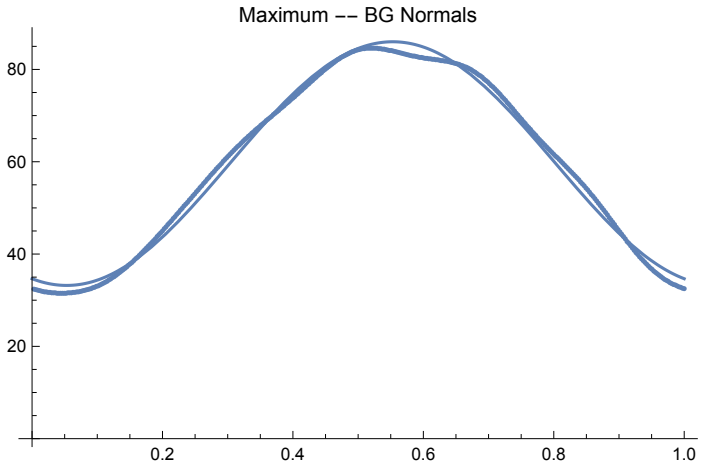
	Estimate	Standard Error	t-Statistic	P-Value
a	59.6011	0.0790277	754.18	0.
amp	26.3961	0.111762	236.181	0.
x0	0.303034	0.000673868	449.693	0.

Out[1131]= 0.999421

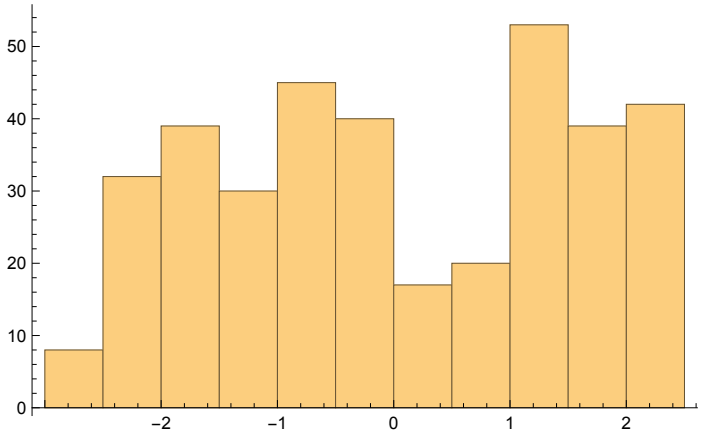
Out[1132]= 0.999416

Out[1133]= {{59.4457, 59.7565}, {26.1763, 26.6159}, {0.301709, 0.304359}}

Out[1134]=



Out[1135]=



Our model has an amplitude of 26.40 (a total variation of twice that), and the curve hits its mean at .3030 of the year; that means it bottoms out at .3030-.25 of the year, or .0530*365.25 -- about the 19th of January.

As before, we now add the linear term, only doing so with NonlinearModelFit.

In[1136]:=
```
maxlm = NonlinearModelFit[data,
  {a + b x + amp Sin[2 Pi (x - x0)]}, {a, b, amp, {x0, .3}}, x]
maxfits = maxlm["FitResiduals"];
maxlm["ParameterTable"]
maxlm["RSquared"]
maxlm["AdjustedRSquared"]
maxlm["ParameterConfidenceIntervals"]
Show[maxplot, Plot[maxlm[x], {x, 0, 1}]]
Histogram[maxfits]
```

Out[1136]= FittedModel[58.9558 + 1.28703 x + 26.2683 Sin[2 π (−0.300682 + x)]]

Out[1138]=

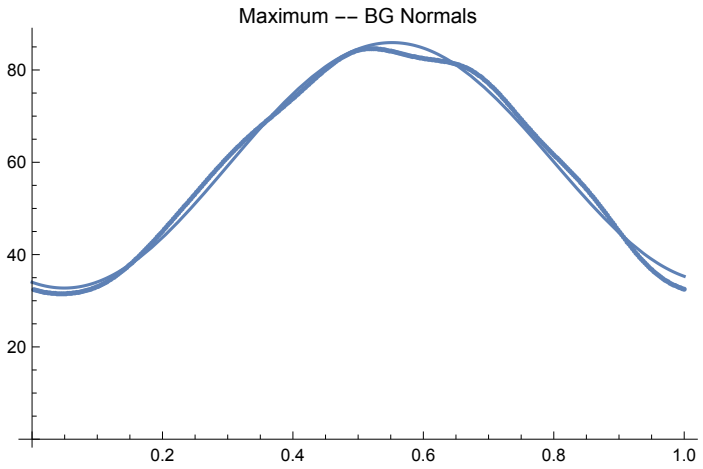
	Estimate	Standard Error	t-Statistic	P-Value
a	58.9558	0.230532	255.739	0.
b	1.28703	0.432552	2.97543	0.00312278
amp	26.2683	0.118271	222.103	0.
x0	0.300682	0.00103924	289.328	0.

Out[1139]= 0.999435

Out[1140]= 0.999428

Out[1141]= {{58.5025, 59.4092}, {0.436392, 2.13767}, {26.0357, 26.5009}, {0.298638, 0.302725}}

Out[1142]=



Out[1143]=