Single population growth models

Objective

We are given a table with the population census at different time intervals between a date a and a date b, and want to get an expression for the population. This allows us to:

- compute a value for the population at any time between the date a and the date b (*interpolation*),
- predict a value for the population at a date before a or after b (*extrapolation*).

PROCEEDINGS

OF THE

NATIONAL ACADEMY OF SCIENCES

Volume 6 JUNE 15, 1920 Number 6

ON THE RATE OF GROWTH OF THE POPULATION OF THE UNITED STATES SINCE 1790 AND ITS MATHEMATICAL REPRESENTATION[1]

BY RAYMOND PEARL AND LOWELL J. REED

DEPARTMENT OF BIOMETRY AND VITAL STATISTICS, JOHNS HOPKINS UNIVERSITY

Read before the Academy, April 26, 1920

Showing the Dates of the Taking of the Census and the Recorded Populations from 1790 to 1910

Date of Census		Recorded Population (Revised Figures from Statistical Abst., 1918)
Year	Month and Day	
1790	First Monday in August	3,929,214
1800	First Monday in August	5,308,483
1810	First Monday in August	7,239,881
1820	First Monday in August	9,638,453
1830	June 1	12,866,020
1840	June 1	17,069,453
1850	June 1	23,191,876
1860	June 1	31,443,321
1870	June 1	38,558,371
1880	June 1	50,155,783
1890	June 1	62,947,714
1900	June 1	75,994,575
1910	April 15	91,972,266

The US population from 1790 to 1910

Year	Population (millions)
1790	3.929
1800	5.308
1810	7.240
1820	9.638
1830	12.866
1840	17.069
1850	23.192

Year	Population (millions)
1860	31.443
1870	38.558
1880	50.156
1890	62.948
1900	75.995
1910	91.972

PLOT THE DATA !!! (here, to 1910)

Using MatLab (or Octave), create two vectors using commands such as

```
t=1790:10:1910;
```

Format is

Vector=Initial value:Step:Final value

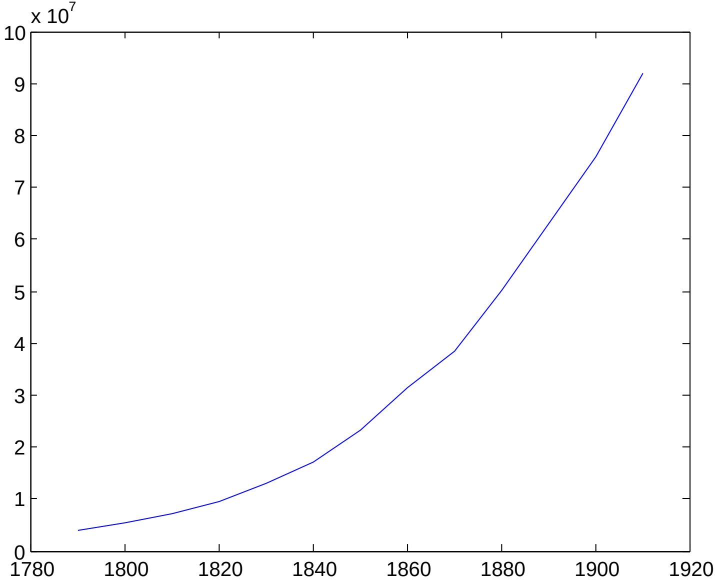
(semicolumn hides result of the command.)

```
P=[3929214,5308483,7239881,9638453,12866020,...
17069453,23191876,31443321,38558371,50155783,...
62947714,75994575,91972266];
```

Here, elements were just listed (... indicates that the line continues below).

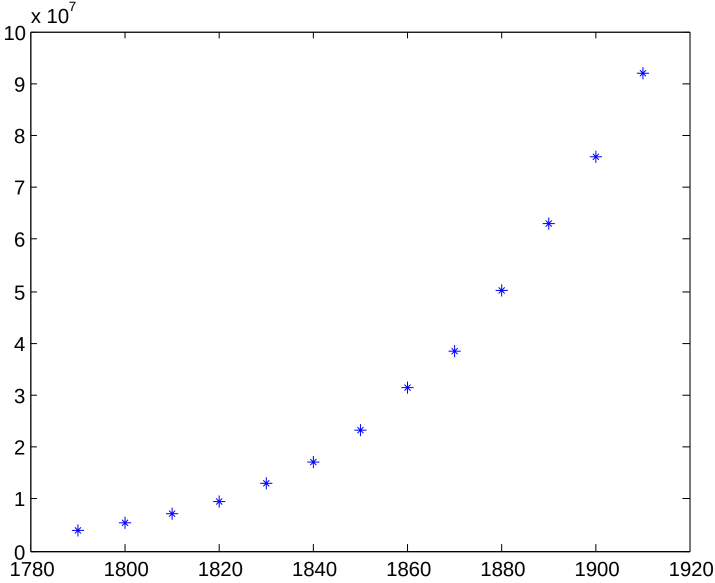
Then plot using

```
plot(t,P);
```



To get points instead of a line

```
plot(t,P,'*');
```



First idea

The curve looks like a piece of a parabola. So let us fit a curve of the form

$$P(t) = a + bt + ct^2.$$

To do this, we want to minimize

$$S = \sum_{k=1}^{13} \left(P(t_k) - P_k\right)^2,$$

where t_k are the known dates, P_k are the known populations, and $P(t_k) = a + bt_k + ct_k^2$.

We proceed as in the notes (but note that the role of a, b, c is reversed):

$$S = S(a, b, c) = \sum_{k=1}^{13} \left(a + bt_k + ct_k^2 - P_k\right)^2$$

is maximal if (necessary condition) $\partial S/\partial a = \partial S/\partial b = \partial S/\partial c = 0$, with

$$\frac{\partial S}{\partial a} = 2\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k)$$

$$\frac{\partial S}{\partial b} = 2\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k)t_k$$

$$\frac{\partial S}{\partial c} = 2\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k)t_k^2$$

So we want

$$2\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k) = 0$$

$$2\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k)t_k = 0$$

$$2\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k)t_k^2 = 0,$$

that is

$$\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k) = 0$$

$$\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k)t_k = 0$$

$$\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k)t_k^2 = 0.$$

Rearranging the system

$$\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k) = 0$$

$$\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k)t_k = 0$$

$$\sum_{k=1}^{13}(a + bt_k + ct_k^2 - P_k)t_k^2 = 0,$$

we get

$$\sum_{k=1}^{13}(a + bt_k + ct_k^2) = \sum_{k=1}^{13} P_k$$

$$\sum_{k=1}^{13}(at_k + bt_k^2 + ct_k^3) = \sum_{k=1}^{13} P_k t_k$$

$$\sum_{k=1}^{13}(at_k^2 + bt_k^3 + ct_k^4) = \sum_{k=1}^{13} P_k t_k^2.$$

$$\sum_{k=1}^{13}(a + bt_k + ct_k^2) = \sum_{k=1}^{13} P_k$$
$$\sum_{k=1}^{13}(at_k + bt_k^2 + ct_k^3) = \sum_{k=1}^{13} P_k t_k$$
$$\sum_{k=1}^{13}(at_k^2 + bt_k^3 + ct_k^4) = \sum_{k=1}^{13} P_k t_k^2,$$

after a bit of tidying up, takes the form

$$\left(\sum_{k=1}^{13} 1\right) a + \left(\sum_{k=1}^{13} t_k\right) b + \left(\sum_{k=1}^{13} t_k^2\right) c = \sum_{k=1}^{13} P_k$$
$$\left(\sum_{k=1}^{13} t_k\right) a + \left(\sum_{k=1}^{13} t_k^2\right) b + \left(\sum_{k=1}^{13} t_k^3\right) c = \sum_{k=1}^{13} P_k t_k$$
$$\left(\sum_{k=1}^{13} t_k^2\right) a + \left(\sum_{k=1}^{13} t_k^3\right) b + \left(\sum_{k=1}^{13} t_k^4\right) c = \sum_{k=1}^{13} P_k t_k^2.$$

So the aim is to solve the linear system

$$\begin{pmatrix} 13 & \sum_{k=1}^{13} t_k & \sum_{k=1}^{13} t_k^2 \\ \sum_{k=1}^{13} t_k & \sum_{k=1}^{13} t_k^2 & \sum_{k=1}^{13} t_k^3 \\ \sum_{k=1}^{13} t_k^2 & \sum_{k=1}^{13} t_k^3 & \sum_{k=1}^{13} t_k^4 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum_{k=1}^{13} P_k \\ \sum_{k=1}^{13} P_k t_k \\ \sum_{k=1}^{13} P_k t_k^2 \end{pmatrix}$$

With MatLab (or Octave), getting the values is easy.

- To apply an operation to every element in a vector or matrix, prefix the operation with a dot, hence

  ```
  t.^2;
  ```

  gives, for example, the vector with every element t_k squared.
- Also, the function `sum` gives the sum of the entries of a vector or matrix.
- When entering a matrix or vector, separate entries on the same row by `,` and create a new row by using `;`.

Thus, to set up the problem in the form of solving $Ax = b$, we need to do the following:

```
format long g;
A=[13,sum(t),sum(t.^2);sum(t),sum(t.^2),sum(t.^3);...
sum(t.^2),sum(t.^3),sum(t.^4)];
b=[sum(P);sum(P.*t);sum(P.*(t.^2))];
```

The `format long g` command is used to force the display of digits (normally, what is shown is in "scientific" notation, not very informative here).

Then, solve the system using

```
A\b
```

We get the following output:

```
>> A\b
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.118391e-020.

ans =

        22233186177.8195
        -24720291.325476
        6872.99686313725
```

(note that here, Octave gives a solution that is not as good as this one, provided by MatLab).

Thus

$P(t) = 22233186177.8195 - 24720291.325476t + 6872.99686313725t^2$

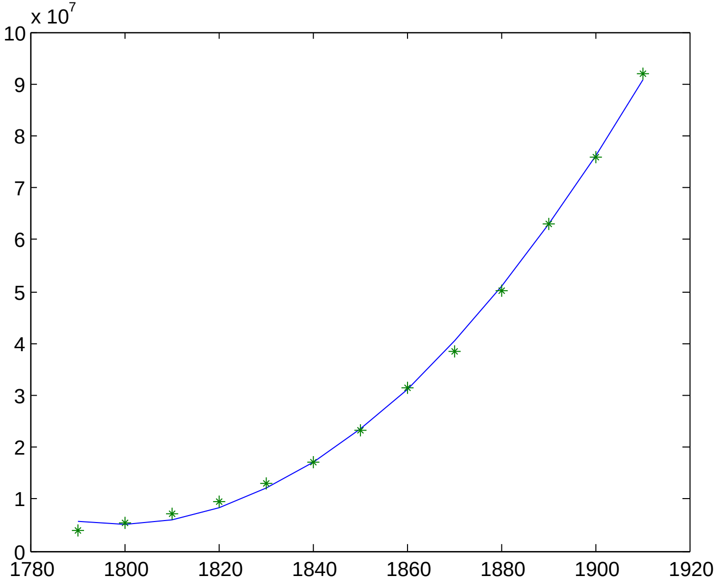
To see what this looks like,

```
plot(t,22233186177.8195-24720291.325476.*t...
+6872.99686313725.*t.^2);
```

(note the dots before multiplication and power, since we apply this function to every entry of t). In fact, to compare with original data:

```
plot(t,22233186177.8195-24720291.325476.*t...
+6872.99686313725.*t.^2,t,P,'*');
```

Our first guess, in pictures



Now we want to generate the table of values, to compare with the true values and thus compute the error. To do this, we can proceed directly:

```
computedP=22233186177.8195-24720291.325476.*t...
+6872.99686313725.*t.^2;
```

We get

```
computedP =

 Columns 1 through 4:

     5633954.39552689      5171628.52739334      6083902.03188705      8370774.90901184

 Columns 5 through 8:

     12032247.1587601      17068318.7811356      23478989.7761383      31264260.1437798

 Columns 9 through 12:

      40424129.884037      50958598.9969215      62867667.4824371      76151335.3405762

 Column 13:

     90809602.5713463
```

We can also create an *inline* function

```
f=inline('22233186177.8195-24720291.325476.*t+6872.99686313725.*t.^2')
f =

     Inline function:
     f(t) = 22233186177.8195-24720291.325476.*t+6872.99686313725.*t.^2
```

This function can then easily be used for a single value

```
octave:24> f(1880)
ans =      50958598.9969215
```

as well as for vectors..

(Recall that t has the dates; t in the definition of the function is a dummy variable, we could have used another letter-.)

```
octave:25> f(t)
```

```
ans =

 Columns 1 through 4:

     5633954.39552689      5171628.52739334      6083902.03188705      8370774.90901184

 Columns 5 through 8:

     12032247.1587601      17068318.7811356      23478989.7761383      31264260.1437798

 Columns 9 through 12:

      40424129.884037      50958598.9969215      62867667.4824371      76151335.3405762
12186176863781.4
 Column 13:

     90809602.5713463
```

Form the vector of errors, and compute sum of errors squared:

```
octave:26> E=f(t)-P;
octave:27> sum(E.^2)
ans =      12186176863781.4
```

Quite a large error (12,186,176,863,781.4), which is normal since we have used actual numbers, not thousands or millions of individuals, and we are taking the square of the error.

To present things legibly, one way is to put everything in a matrix..

```
M=[P;f(t);E;E./P];
```

This matrix will have each type of information as a row, so to display it in the form of a table, show its transpose, which is achieved using the function `transpose` or the operator $'$.

```
M'
ans =
 3929214       5633954.39552689     1704740.39552689     0.433862954658
 5308483       5171628.52739334    -136854.472606659   -0.0257803354756
 7239881       6083902.03188705    -1155978.96811295    -0.15966822771
 9638453       8370774.90901184    -1267678.09098816    -0.131522983095
12866020       12032247.1587601    -833772.841239929   -0.0648042550252
17069453       17068318.7811356    -1134.21886444092  -6.644728828e-05
23191876       23478989.7761383     287113.776138306    0.0123799289086
31443321       31264260.1437798    -179060.856220245  -0.00569471832254
38558371        40424129.884037     1865758.88403702    0.0483879073635
50155783       50958598.9969215     802815.996921539    0.0160064492846
62947714       62867667.4824371    -80046.5175628662  -0.00127163502018
75994575       76151335.3405762     156760.340576172   0.00206278330494
91972266       90809602.5713463    -1162663.42865372    -0.012641456813
```

Now for the big question...

How does our formula do for present times?

```
f(2006)
ans =      301468584.066013
```

Actually, quite well: 301,468,584, compared to the 298,444,215 July 2006 estimate, overestimates the population by 3,024,369, a relative error of approximately 1%.