COVID-19 Death Number Gompertz Curve Modeling and Forecasting of UK-2020

kobayashikorio@gmail.com
2020/4/15

元データのアーカイブ場所

多くのデータサイトはJohns Hopkins大学からデータを引用している。
2019 Novel Coronavirus COVID-19 (2019-nCoV) Data Repository by Johns Hopkins CSSE
https://github.com/CSSEGISandData/COVID-19

J.Hopkins大学のデータがJSONに変換された例がある。
JSON time-series of coronavirus cases (confirmed, deaths and recovered) per country - updated daily
https://github.com/pomber/covid19

このサイトのJSONファイルは以下から読み取ることができる。
https://pomber.github.io/covid19/timeseries.json

死亡者数方程式の導出

死亡者数の時間的な変化を与える方程式を考える。死亡数をy として、ある時間 dt において、この、時間あたりの死亡者数の変化 dy が、dy/dt ＝ Ａｙ＾ｅxp(－Ｂt)　と表わされるとする。すなわち、yの増加率は、直前の y に比例して増加するパラメータと時間tに伴い指数的に減少するパラメータに関係する。この方程式の解として、y= k b^Exp[-c t]　が導かれる。この曲線はゴンペルツ曲線 （Gompertz curve）と呼ばれる。微分方程式に示すところによれば、指数関数部分は、単に時間tで決定される事になる。

JSONファイルの読み込み

In[161]:=
```
country = "United Kingdom";
population = 66 181 585;
urlCovid19 = "https://pomber.github.io/covid19/timeseries.json";
data = Map[Association, Association[Import[urlCovid19]][country], {1}];
ddata = {DateObject[#["date"]], #["deaths"]} & /@ data;
firstday = ddata[[1, 1]];
Last[ddata]
```

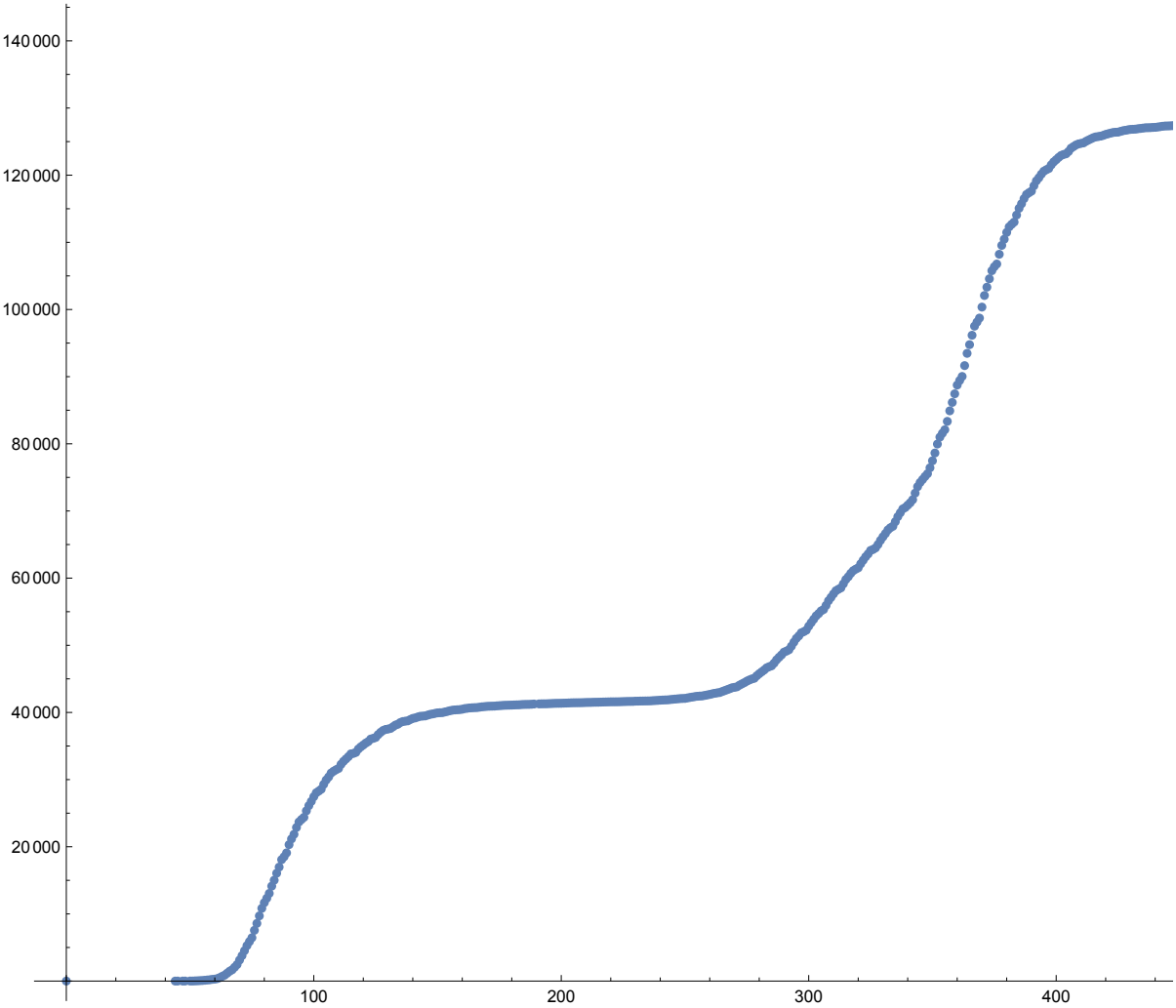
Out[ ]=
```
{Sat 9 Oct 2021, 138 101}
```

```
entity = "UnitedKingdom";
Normal[Entity["Country", entity]["Population"]]
```

初回報告日からの経過日数を求めたのち報告のなかった日をデータから削除。

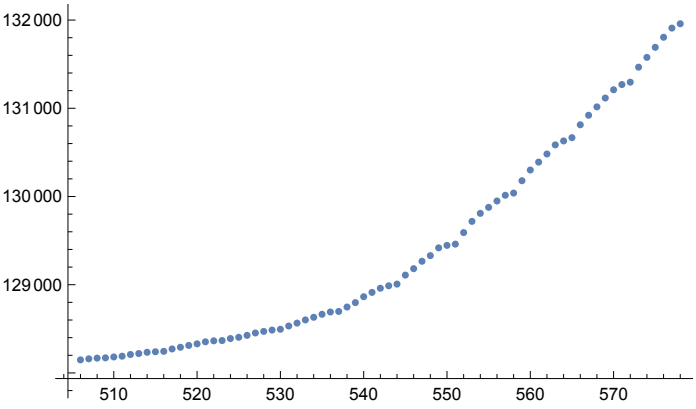
In[167]:=
```
i4 = Map[{QuantityMagnitude[#[[1]] - firstday], #[[2]]} &, ddata];
raw = Map[First, Gather[i4, #1[[2]] == #2[[2]] &]];
ListPlot[raw]
```

Out[ ]=



In[ ]:=
```
ListPlot[raw[[460 ;;]]]
```

Out[ ]=

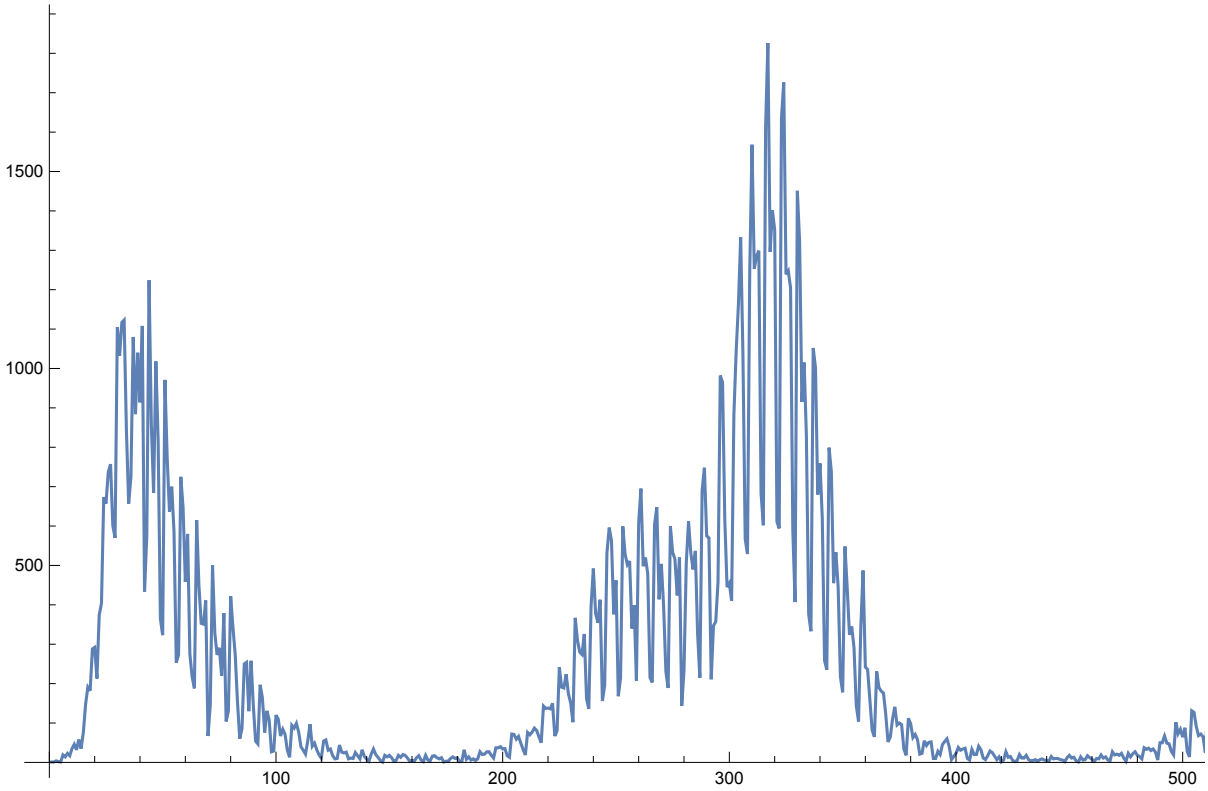


単位期間あたりのDeath-Numberの変化

各報告とその一つ前の報告について、Death-Numberの変化をΔN、報告間隔をΔDとした時の、ΔN/ΔDをプロットする。

```
In[ ]:= raw

In[ ]:= i5 = BlockMap[#[[2]] - #[[1]] &, i4, 2, 1];
        i6 = Map[(#[[2]] / #[[1]]) &, i5];
        i7 = First[Transpose[Drop[i4, 1]]];

In[ ]:= {Dimensions@raw, Dimensions@delta}
Out[ ]= {{491, 2}, {}}

In[ ]:= delta = Apply[Subtract, Map[Reverse,
            BlockMap[Transpose, Select[raw, 30 < #[[1]] ≤ 650 &], 2, 1], {2}], {2}];

In[ ]:= ListLinePlot[Map[Last, delta]]
```

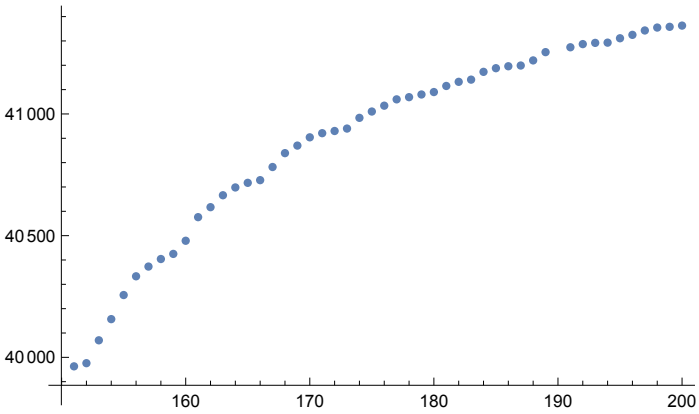
Out[ ]=



グラフを部分的に拡大すると、数百人程度が死亡する、比較的小さな事象がいくつも重なっているのがわかる。今の所、比較的小さな事象の生起割合は７日周期を持っており、医療施設の運用が週を元に実施されているであろうことと合致している。

In[ ]:=
```
targetPeriod = Select[raw, 150 < #[[1]] ≤ 200 &];
ListPlot[targetPeriod]
```

Out[ ]=



解析1回目

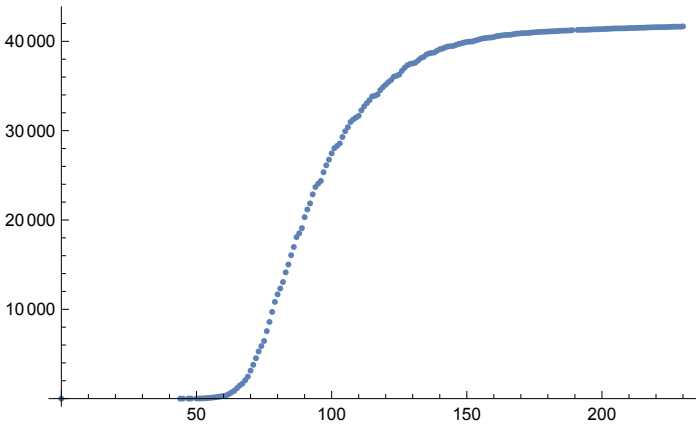
時系列の分割

報告開始日から、ある期日までの結果は、最初のウィルス株によるもの、それ以降の結果は、次のウィルス株によるものと考えて、報告データを切り分ける。この切りわけ日の設定は、報告数の時系列が作る曲線の変曲点からヒューリスティックに決める。

1. 第一回は230日で切り分ける。

In[ ]:=
```
rawModified[1] = raw;
targetPeriod = Select[rawModified[1], #[[1]] ≤ 230 &];
ListPlot[targetPeriod]
```

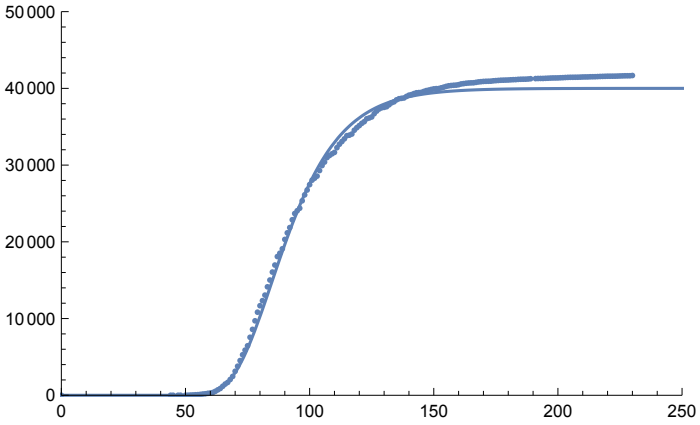
Out[ ]=



2. 分割した時系列のうち、最初の時系列をGompertz曲線にマッチングさせるために、ヒューリスティックに初期値を与える。初期値を与えるために、報告値と比較しながら、初期値を調整する。

In[ ]:=
```
Claear[c];
start = 55;
model[1] = n b^Exp[-c (t - start)];

initialguess = {n → 40 000, c → 0.065, b → 0.001};
Show[
 Plot[model[1] /. initialguess, {t, 0, 1000}, PlotRange → {{0, 250}, {0, 50 000}}],
 ListPlot[targetPeriod]
]
```

Out[ ]=



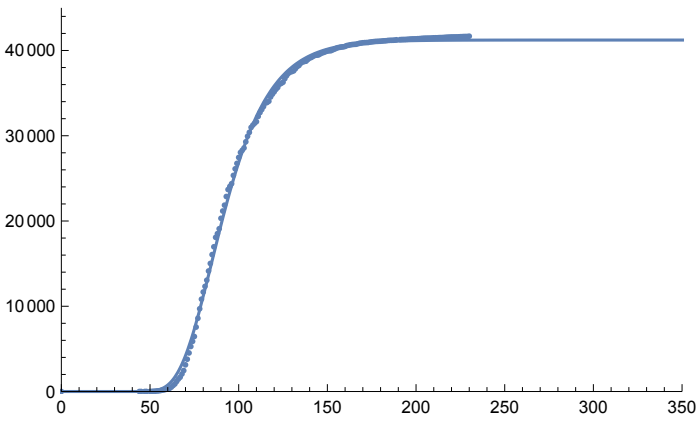
3. Fitting関数を用い、初期値を与えて、最初の時系列をGompertz曲線にFittingさせる。

In[ ]:=
```
ans[1] = FindFit[targetPeriod, model[1],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[ ]=
```
{n → 41 225., c → 0.0558033, b → 0.00512729}
```

In[ ]:=
```
Show[
 Plot[model[1] /. ans[1], {t, 0, 400}, PlotRange → {{0, 350}, {0, 45 000}}],
 ListPlot[targetPeriod]
]
```

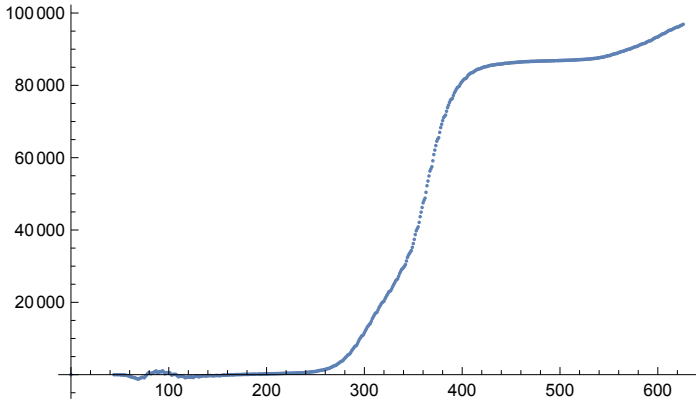
Out[ ]=



4. 時系列にfittingさせたGompertz曲線を用いて、報告時系列から、分割された時系列から推定した、推定死亡数時系列を差し引く。差し引いた時系列をモディファイド時系列と呼ぶ。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[1]];
ListPlot[
 rawModified[2] = Transpose@{pdays, death - (model[1] /. ans[1] /. t → pdays)}]
```

Out[ ]=



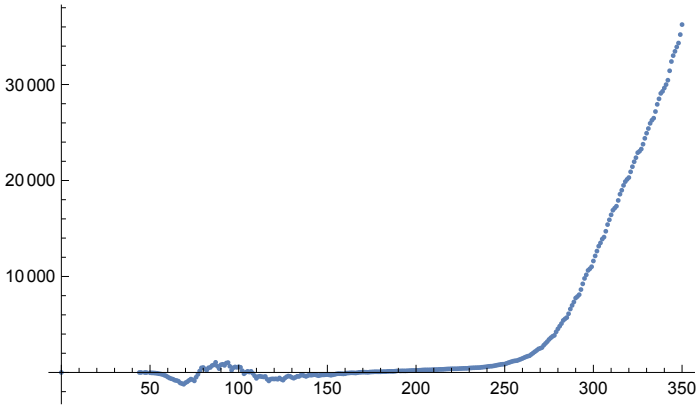
2回目

In[ ]:=

1. 350日で切り分ける。

In[ ]:=
```
targetPeriod = Select[rawModified[2], #[[1]] ≤ 350 &];
ListPlot[targetPeriod]
```

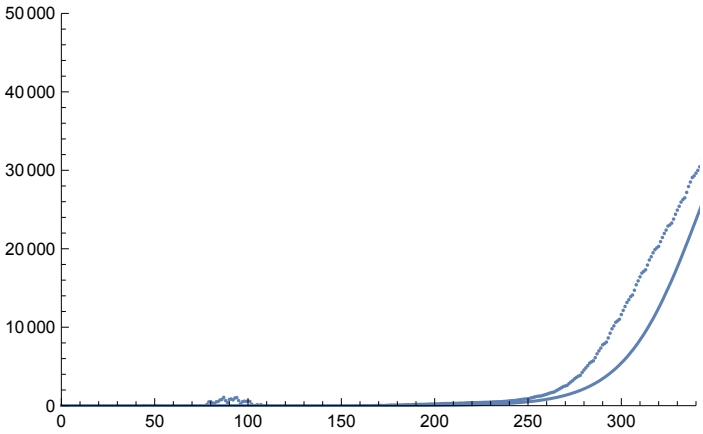
Out[ ]=



2. 分割した時系列のうち、時系列をLogistic曲線にマッチングさせるために、ヒューリスティックに初期値を与える。初期値を与えるために、報告値と比較しながら、初期値を調整する。

In[ ]:=
```
start = 250;
inter = 340;
model[2] = n / (1 + c Exp[-a n (t - start)]);;
initialguess = {n → 50 000, c → 100.0, a → 0.000001};
Show[
 Plot[model[2] /. initialguess,
  {t, 0, 500}, PlotRange → {{0, inter}, {0, 50 000}}],
 ListPlot[rawModified[2]]
]
```

Out[ ]=



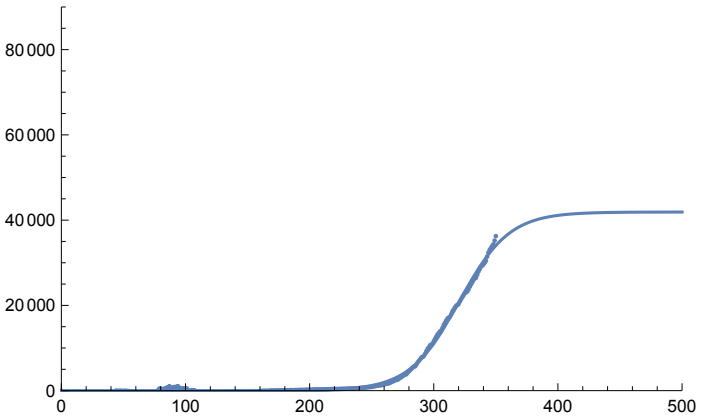
3. Fitting関数を用い、初期値を与えて、最初の時系列をLogistic曲線にFittingさせる。

In[ ]:=
```
ans[2] = FindFit[targetPeriod, model[2],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[ ]= $\left\{n \to 41\,899.8,\ c \to 33.7363,\ a \to 1.18914 \times 10^{-6}\right\}$

In[ ]:=
```
upto = 500;
Show[
 Plot[model[2] /. ans[2], {t, 0, upto}, PlotRange → {{0, 500}, {0, 90 000}}],
 ListPlot[targetPeriod]
]
```

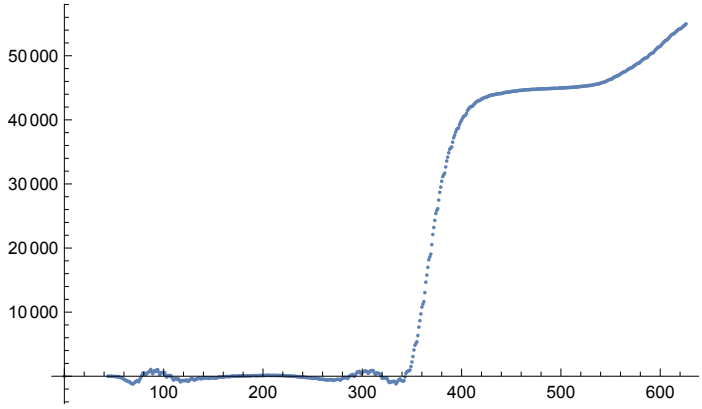
Out[ ]=



4. 時系列にfittingさせたLogistic曲線を用いて、報告時系列から、分割された時系列から推定した、推定死亡数時系列を差し引く。差し引いた時系列をモディファイド時系列と呼ぶ。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[2]];
ListPlot[
 rawModified[3] = Transpose@{pdays, death - (model[2] /. ans[2] /. t → pdays)}]
```

Out[ ]=



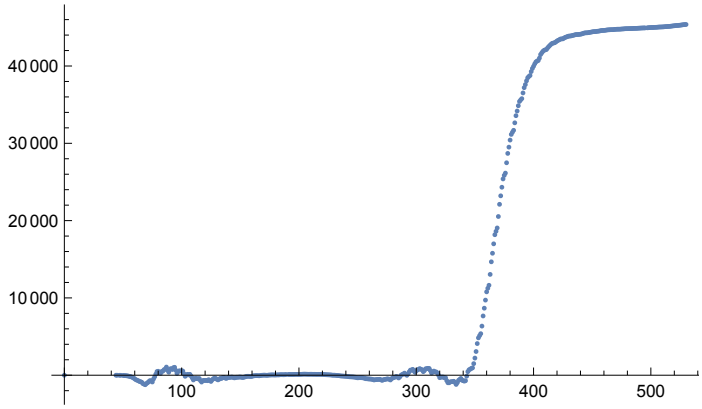
3回目

In[ ]:=

1. 530日で切り分ける。

In[ ]:=
```
targetPeriod = Select[rawModified[3], #[[1]] ≤ 530 &];
ListPlot[targetPeriod]
```

Out[ ]=

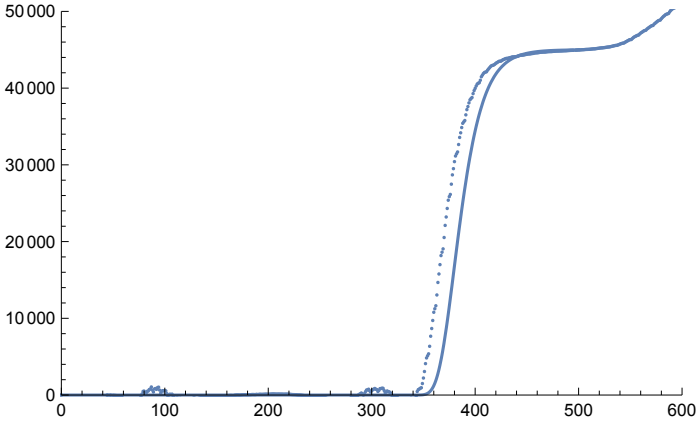


2. 分割した時系列のうち、時系列をGompertz曲線にマッチングさせるために、ヒューリスティックに初期値を与える。初期値を与えるために、報告値と比較しながら、初期値を調整する。

In[ ]:=
```
start = 350;
inter = 600;
model[3] = n b^Exp[-c (t - start)];
initialguess = {n → 45 000, c → 0.065, b → 0.001};
Show[
 Plot[model[3] /. initialguess,
  {t, 0, 500}, PlotRange → {{0, inter}, {0, 50 000}}],
 ListPlot[rawModified[3]]
]
```

··· General: $0.001^{7.58422\times10^{9}}$ は正規化された機械数として表すには小さすぎます．精度が失われる可能性があります．

Out[ ]=



3. Fitting関数を用い、初期値を与えて、最初の時系列をGompertz曲線にFittingさせる。

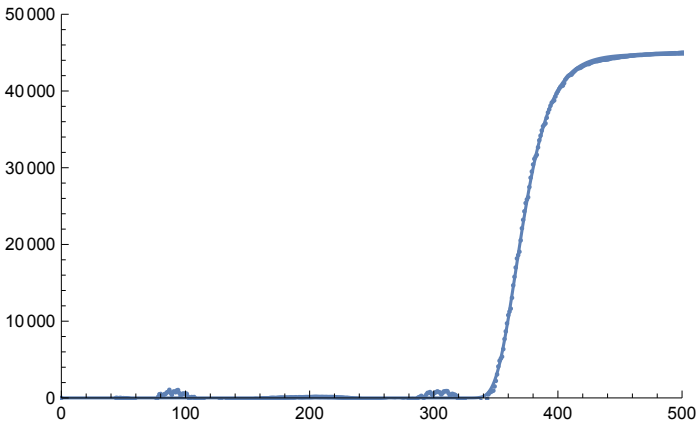
In[ ]:=
```
ans[3] = FindFit[targetPeriod, model[3],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[ ]=
```
{n → 44 807.6, c → 0.0645583, b → 0.0607658}
```

In[ ]:=
```
upto = 560;
Show[
 Plot[model[3] /. ans[3], {t, 0, upto}, PlotRange → {{0, 500}, {0, 50 000}}],
 ListPlot[targetPeriod]
]
```

··· General:
$0.0607658^{6.49746\times10^{9}}$ は正規化された機械数として表すには小さすぎます．精度が失われる可能性があります．

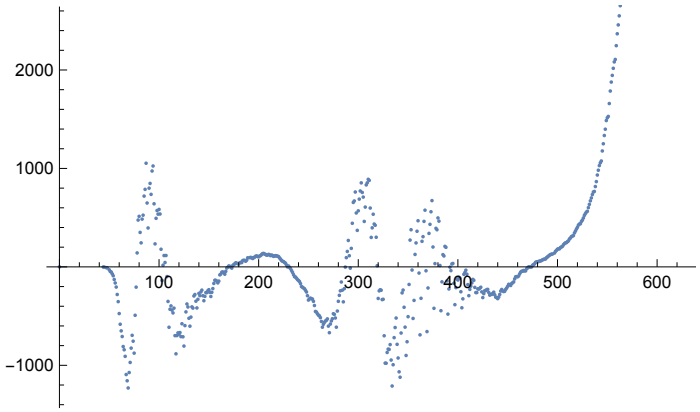
Out[ ]=



4. 時系列にfittingさせたLogistic曲線を用いて、報告時系列から、分割された時系列から推定した、推定死亡数時系列を差し引く。差し引いた時系列をモディファイド時系列と呼ぶ。

```
In[•]:= {pdays, death} = Transpose[rawModified[3]];
        ListPlot[
         rawModified[4] = Transpose@{pdays, death - (model[3] /. ans[3] /. t → pdays)}]
```

Out[•]=



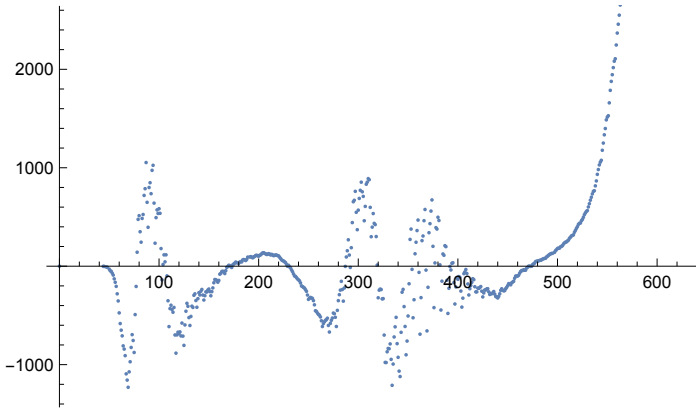
4回目

最終はLogistic曲線が選択される。ただし、2回目はLogistic曲線でなければ、適切なモディファイド曲線が得られない。

1. 全てのデータ。

```
In[•]:= targetPeriod = Select[rawModified[4], #[[1]] ≤ 700 &];
        ListPlot[targetPeriod]
```

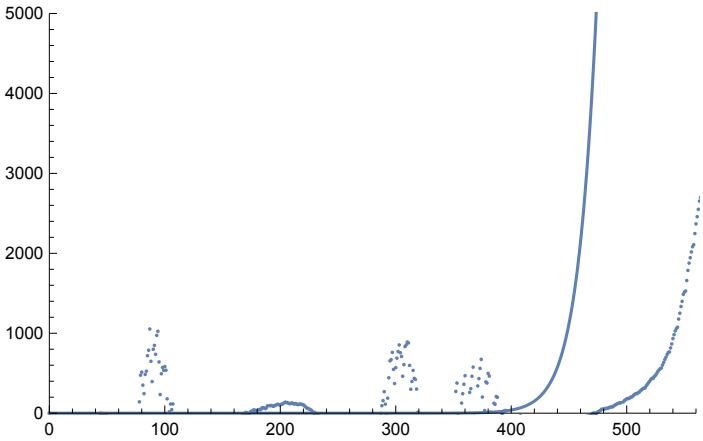
Out[•]=



2. 分割した時系列のうち、時系列をLogistic曲線にマッチングさせるために、ヒューリスティックに初期値を与える。初期値を与えるために、報告値と比較しながら、初期値を調整する。

```
In[•]:= start = 460;
        inter = 560;
        model[4] = n / (1 + c Exp[-a n (t - start)]);
        initialguess = {n → 45 000, c → 20, a → 0.0000015};
        (*model[4]=n b^Exp[-c( t-start)];
        initialguess={n→50000,c→0.03,b→0.001};*)
```

In[]:=
```
initialguess = {n → 45 000, c → 20, a → 0.0000015};
Show[
 Plot[model[4] /. initialguess, {t, 0, 560}, PlotRange → {{0, inter}, {0, 5000}}],
 ListPlot[rawModified[4]]
]
```

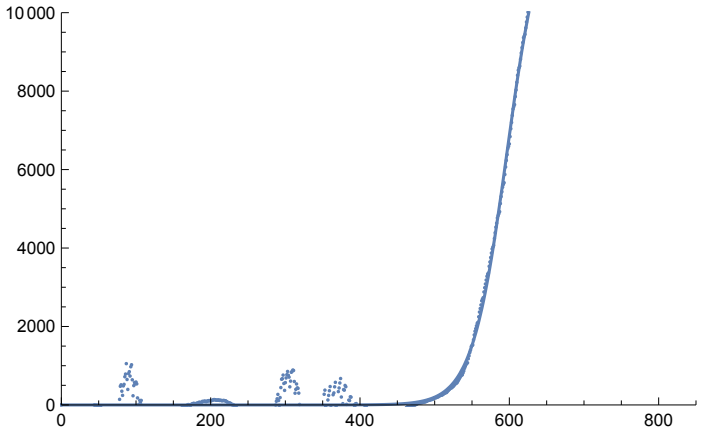
Out[]=


3. Fitting関数を用い、初期値を与えて、最初の時系列をLogistic曲線にFittingさせる。

In[]:=
```
ans[4] = FindFit[targetPeriod, model[4],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[]=
$$\{n \to 12\,833.9,\ c \to 356.478,\ a \to 3.34904 \times 10^{-6}\}$$

In[]:=
```
upto = 850;
Show[
 Plot[model[4] /. ans[4], {t, 0, upto}, PlotRange → {{0, upto}, {0, 10 000}}],
 ListPlot[rawModified[4]]
]
```

Out[]=


全ての時系列へのフィッティング

In[•]:=
```
percent =
 100 Divide[(model[1] /. ans[1]) + (model[2] /. ans[2]) + (model[3] /. ans[3]) +
     (model[4] /. ans[4]) /. t → Infinity, population]
```

Out[•]= 0.212697

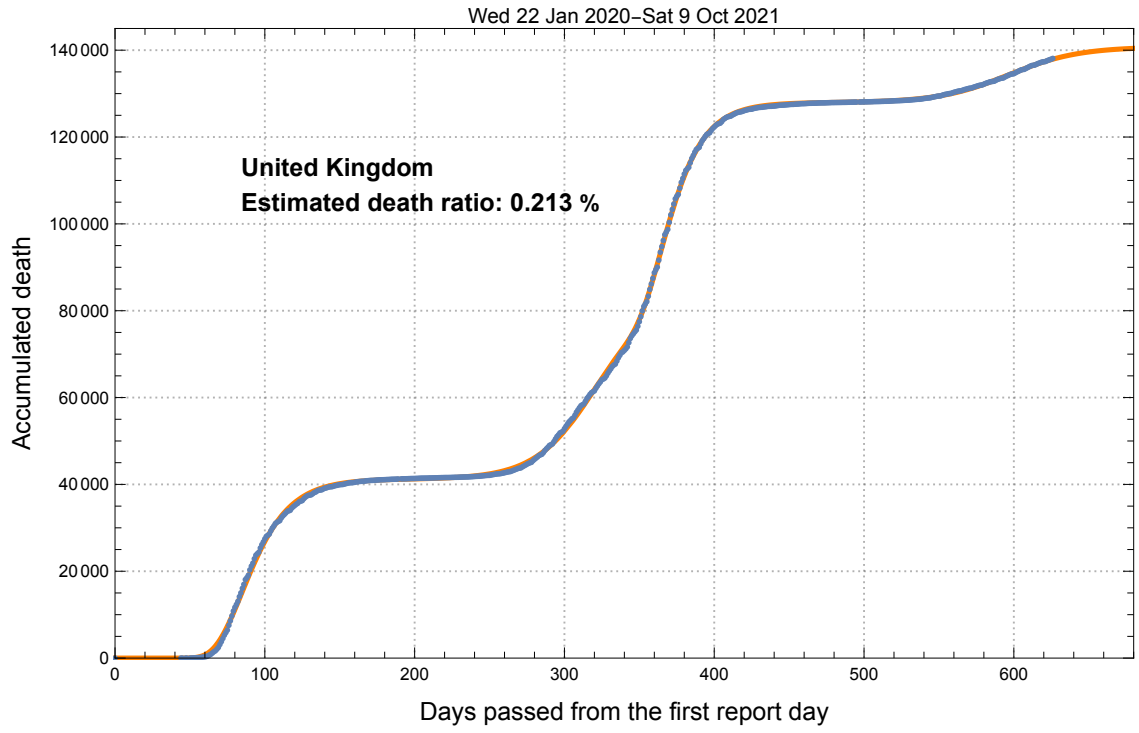
In[•]:=
```
percent * population / 100
```

Out[•]= 140 766.

In[ ]:=
```
fdate = First[ddata][[1]];
ldate = Last[ddata][[1]];
upto = 680;

Show[
 Plot[(model[1] /. ans[1]) + (model[2] /. ans[2]) + (model[3] /. ans[3]) +
    (model[4] /. ans[4]), {t, 0, upto}, PlotRange → {{0, upto}, {0, 145 000}},
  PlotTheme → "Grid", PlotStyle → {Orange, Thickness[0.005]}],
 ListPlot[raw, PlotLegends →
   Placed[Text[Style[country <> "\n" <> "Estimated death ratio: " <>
       ToString[DecimalForm[percent, 3]] <> " %", 13, Bold]], {0.3, 0.75}]],

 ImageMargins → 20,
 Frame → True,
 PlotLabel → DateString[fdate] <> "-" <> DateString[ldate],
 FrameLabel → {Style["Days passed from the first report day", 13],
   Style["Accumulated death", 13]}, LabelStyle → {Black}]
```

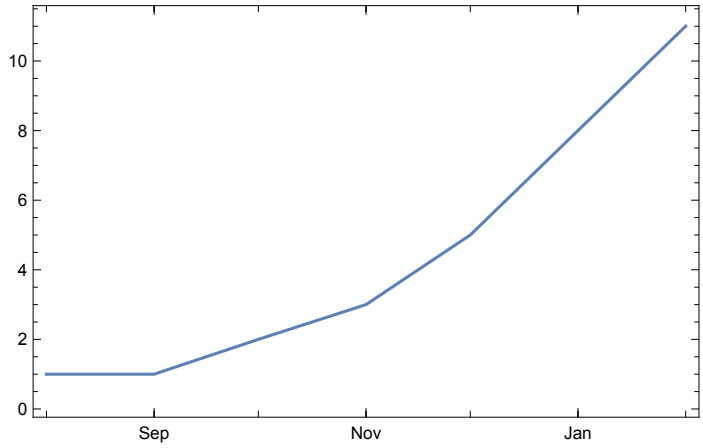
General:
0.0607658 $^{6.49643\times10^{9}}$ は正規化された機械数として表すには小さすぎます．精度が失われる可能性があります．

Out[ ]=



```
DateListPlot[{1, 1, 2, 3, 5, 8, 11}, {2000, 8}, Ticks → None]
```

Out[ ]=



In[ ]:=
```
data = {{{2006, 10, 1}, 8}, {{2006, 10, 10}, 10}, {{2006, 10, 20}, 12},
    {{2006, 10, 24}, 14}, {{2006, 11, 5}, 15}, {{2006, 11, 15}, 20}};
```

In[ ]:=
```
DateListPlot[data, {2006, 6}, Ticks → None]
```

Out[ ]=