COVID-19 Death Number Gompertz Curve Modeling and Forecasting of UK-2020 to 2022

kobayashikorio@gmail.com
2020/4/15
2021/1/24

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。すなわち、自然免疫の否定である。

JSONファイルの読み込み

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

上記を最初に一度だけ実行し、人口数を得る。

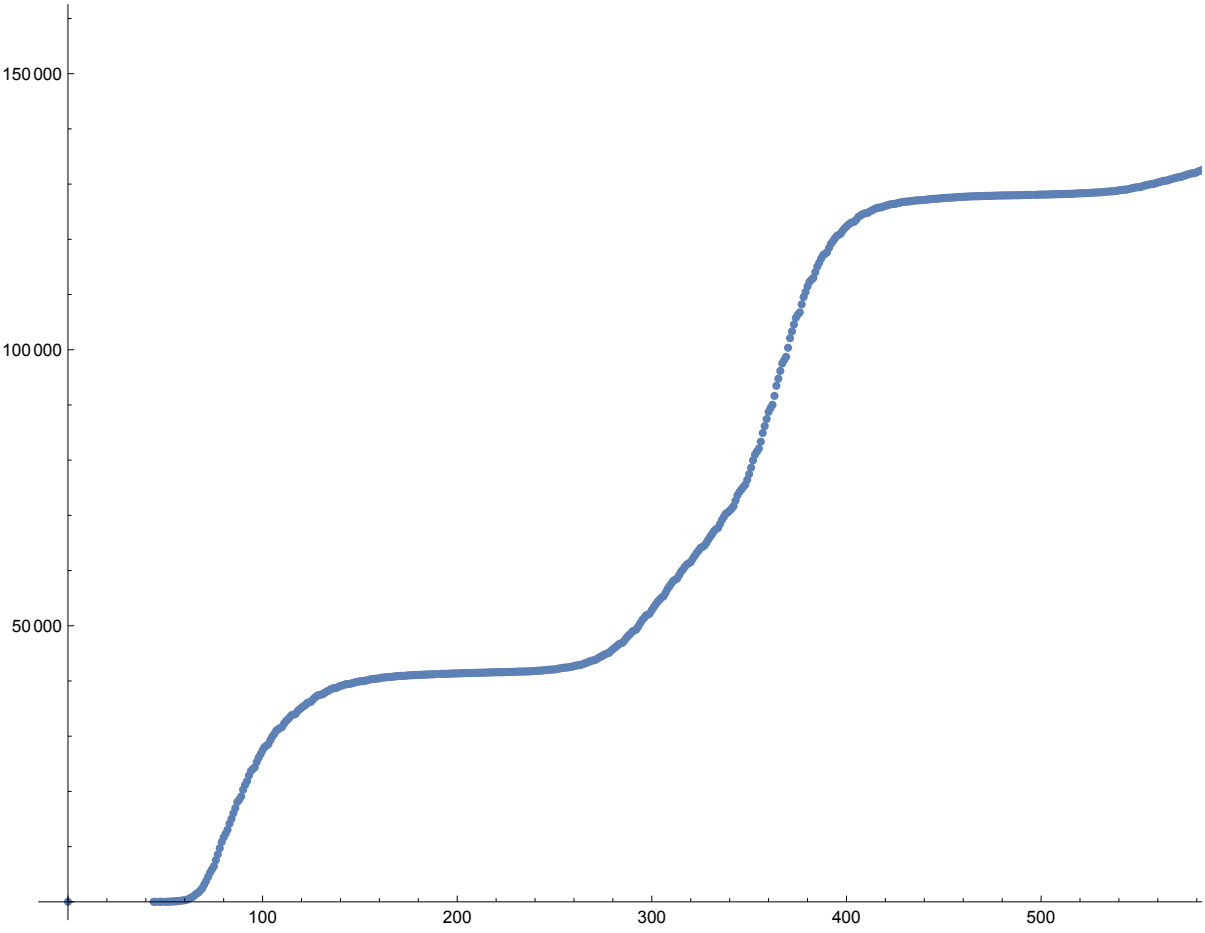
In[]:=
```
country = "United Kingdom";
population = 66 181 585;
urlCovid19 = "https://pomber.github.io/covid19/timeseries.json";
data = Map[Association, Association[Import[urlCovid19]][country], {1}];
ddata = {DateObject[#["date"]], #["deaths"]} & /@ data;
firstday = ddata[[1, 1]];
Last[ddata]
```

Out[]= { Sat 22 Jan 2022 , 154 298 }

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

Out[]=



1回目 Gompertz曲線

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

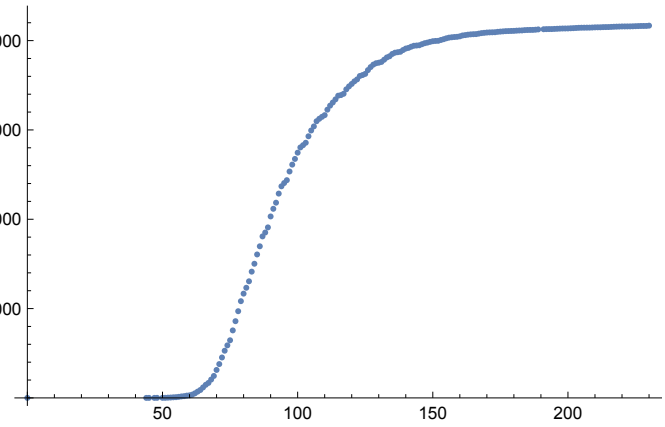
は、次のウィルス株によるものと考えて、報告データを切り分ける。この切りわけ日の設定は、報告数の時系列が作る曲線の変曲点からヒューリスティックに決める。

1. 第一回は230日で切り分ける。

この場合、最初から230日目までが、Fitting対象である、targetPeriodである。

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

Out[ ]=

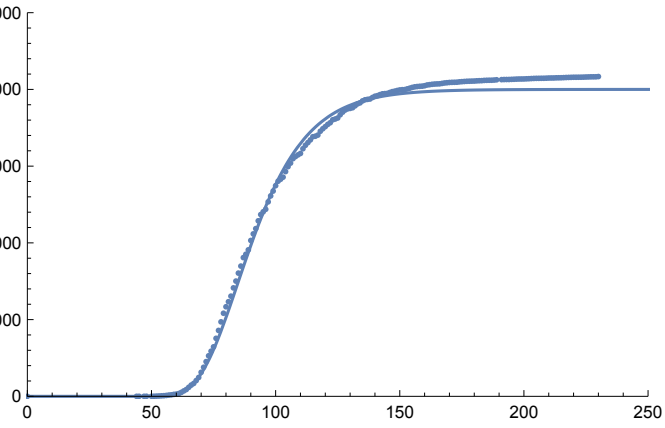


2. 分割した時系列のうち、最初の時系列をGompertz曲線にマッチングさせるために、ヒューリスティックに初期値を与える。初期値を与えるために、報告値と比較しながら、初期値を調整する。

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

initialguess = {n → 40000, c → 0.065, b → 0.001};
Show[
 Plot[model[1] /. initialguess, {t, 0, 1000}, PlotRange → {{0, 250}, {0, 50000}}],
 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。

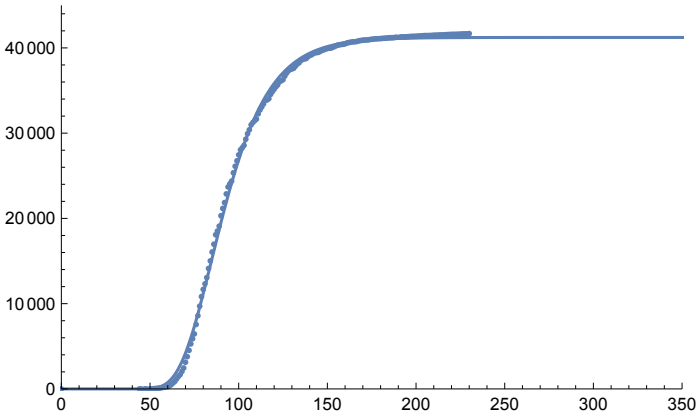
In[ ]:=
```
ans[1] = FindFit[targetPeriod, model[1],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

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

報告値の一部であるtargetPeriodとmodelをFittingした結果を確かめる。

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

Out[ ]=

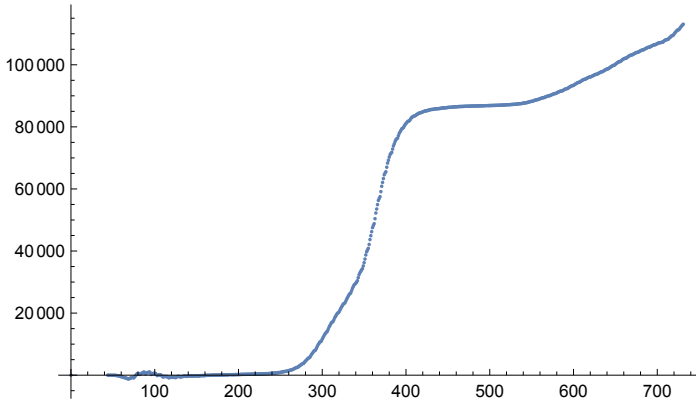


4. 時系列にfittingさせたGompertz曲線を用いて、報告時系列から、分割された時系列から推定した、推定死亡数時系列を差し引く。差し引いた時系列をモディファイド時系列と呼び次のステップで用いる。

報告時系列から、Fittingさせたmodel時系列を差し引くと、ゼロ近辺の値となる。

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

Out[ ]=



2回目Gompertz曲線

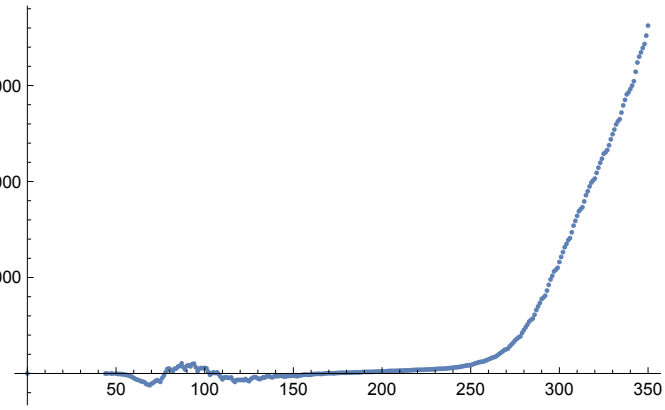
In[ ]:=

1. 350日で切り分ける。

350日以前の報告値がtargetPeriodである。

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

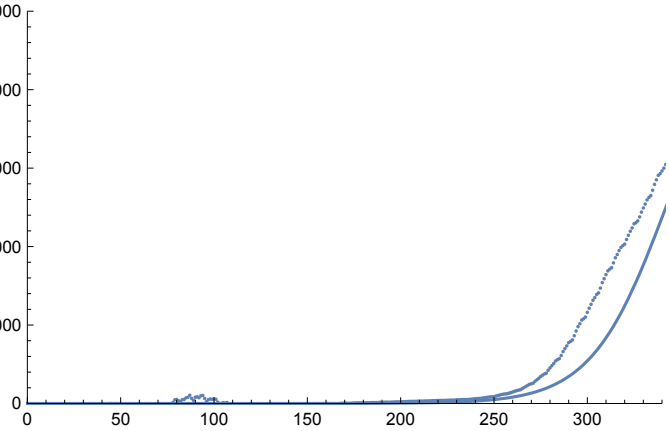
Out[ ]=



2. 分割した時系列のうち、時系列をGoempertz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[ ]=



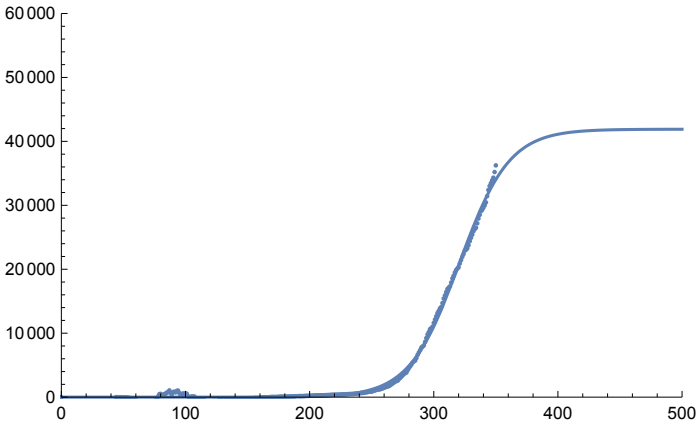
3. Fitting関数を用い、初期値を与えて、target時系列にGompertz曲線をFittingさせる。

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

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

In[]:=
```
upto = 500;
Show[
 Plot[model[2] /. ans[2], {t, 0, upto}, PlotRange → {{0, 500}, {0, 60 000}}],
 ListPlot[targetPeriod]
]
```

Out[]=

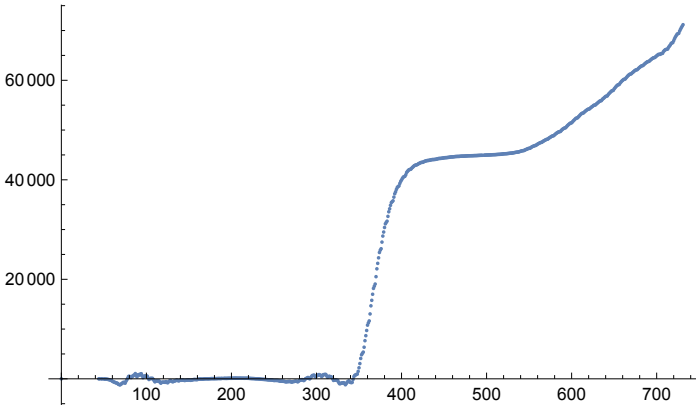


4. 時系列にfittingさせたGompertz曲線を用いて、報告時系列から、分割された時系列から推定した、推定死亡数時系列を差し引く。差し引いた時系列をモディファイド時系列と呼び次のステップで用いる。

モディファイド時系列をゼロに近づける時に、ゼロ周辺に誤差が残る。大きな誤差はこれ以降のFitting操作に影響を与え、適切にFittingが出来ない場面が生じる。

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

Out[]=

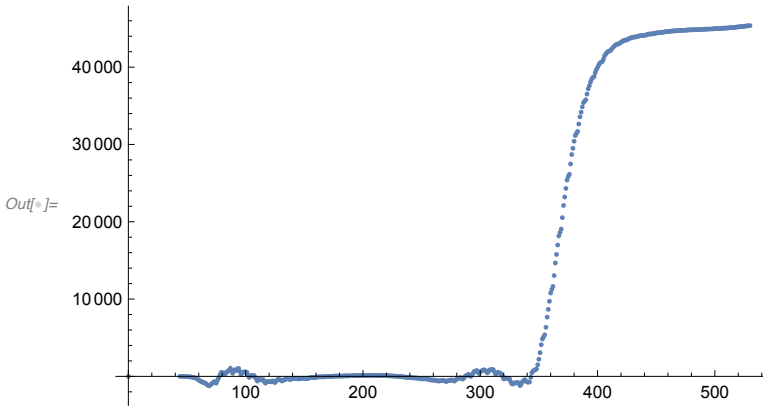


3回目 Gompertz 曲線

1. 530日で切り分ける。

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

Out[ ]=

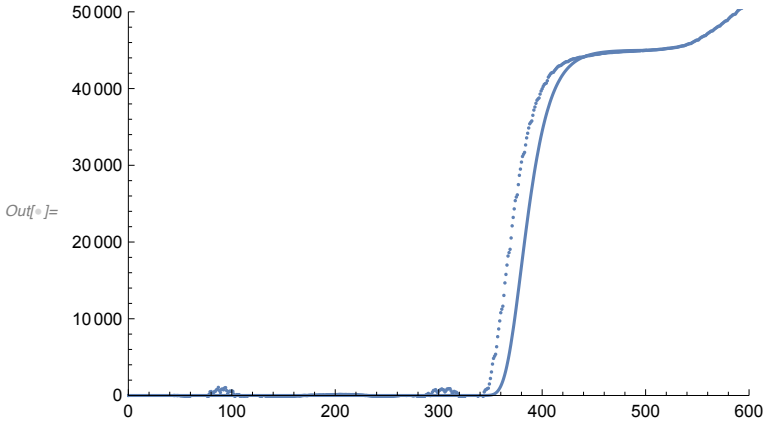


2. 分割した時系列のうち、時系列をGompertz曲線にマッチングさせるために、ヒューリスティックに初期値を与える。初期値を与えるために、報告値と比較しながら、初期値を調整する。

```
In[ ]:= start = 350;
        inter = 600;
        model[3] = n b^Exp[-c (t - start)];
        initialguess = {n → 45000, c → 0.065, b → 0.001};
        Show[
         Plot[model[3] /. initialguess,
          {t, 0, 500}, PlotRange → {{0, inter}, {0, 50000}}],
         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、targetPeriod時系列にGompertz曲線をFittingさせる。

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

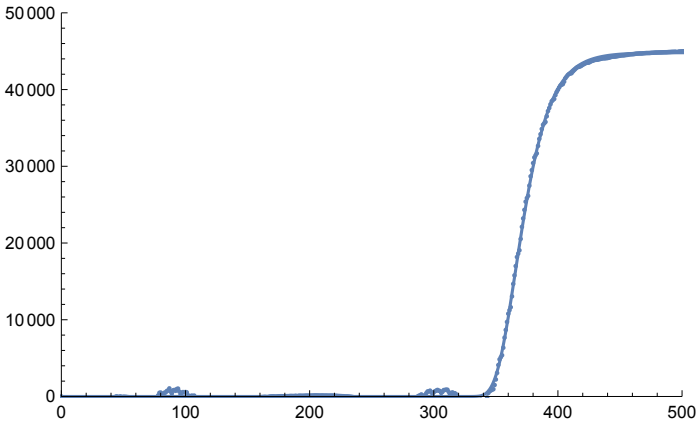
Out[ ]= {n → 44807.6, c → 0.0645583, b → 0.0607658}

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．

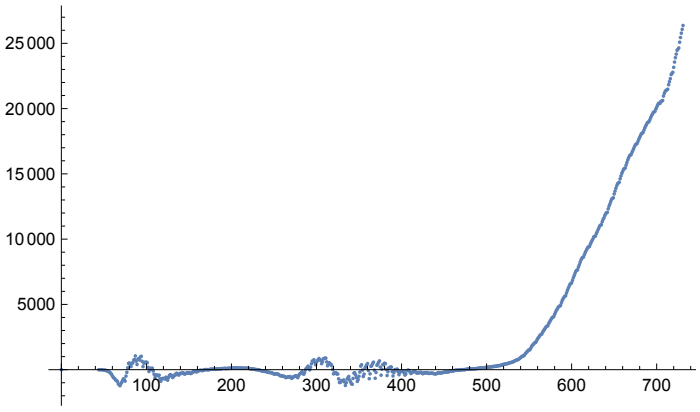
Out[ ]=



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び次のステップで用いる。

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

Out[ ]=

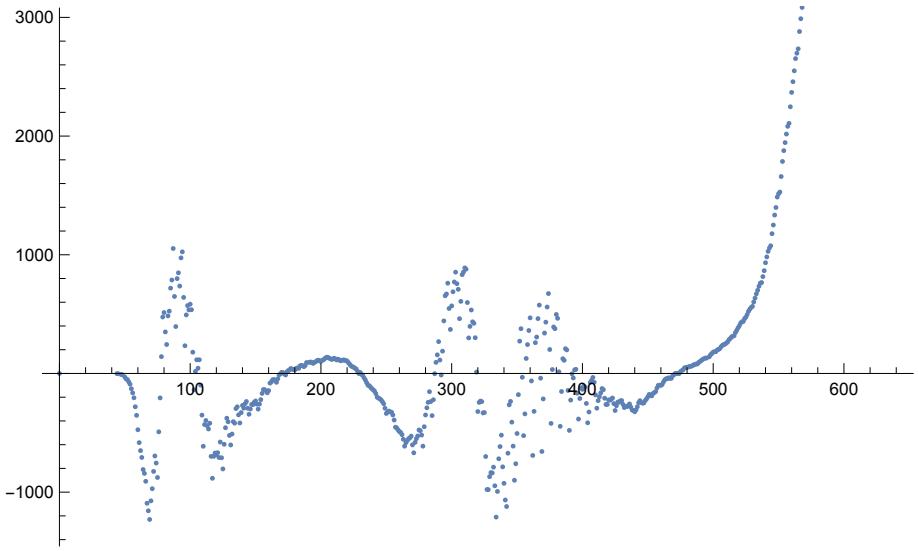


4回目 Gompertz 曲線

1. 次のターゲット期間を切り分ける

In[ ]:=
```
targetPeriod = Select[rawModified[4], #[[1]] ≤ 640 &];
ListPlot[targetPeriod]
```

Out[ ]=

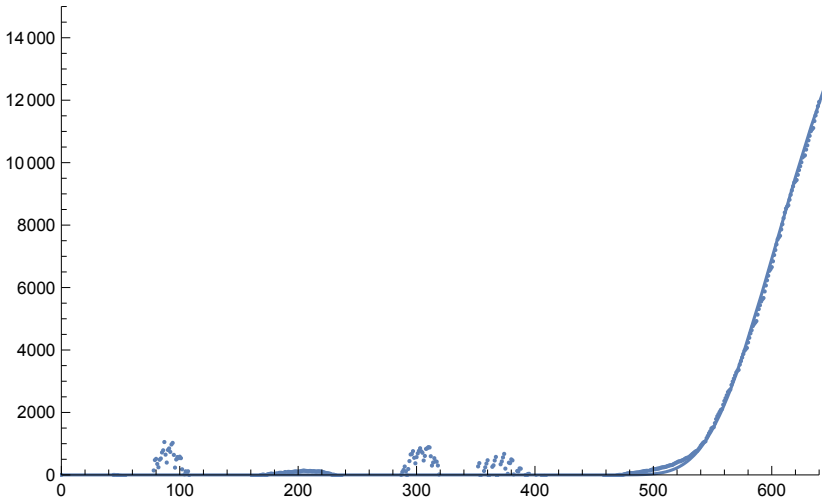


2. 分割した時系列のうち、時系列をGompertz曲線にマッチングさせるために、ヒューリスティックに初期値を与える。初期値を与えるために、報告値と比較しながら、初期値を調整する。

In[ ]:=
```
start = 480;
inter = 640;
model[4] = n b^Exp[-c (t - start)];
initialguess = {n → 20 000, c → 0.018, b → 0.0001};
Show[
 Plot[model[4] /. initialguess,
  {t, 0, 690}, PlotRange → {{0, inter}, {0, 15 000}}],
 ListPlot[targetPeriod]
]
```

··· General: $0.0001^{5651.9}$ 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Logistic曲線にFittingさせる。

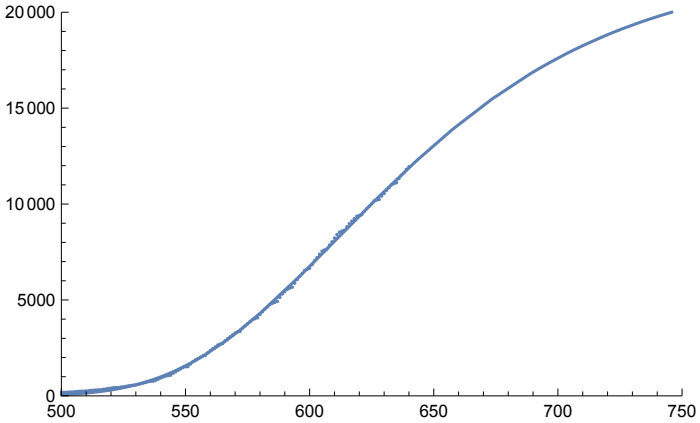
In[]:=
```
ans[4] = FindFit[targetPeriod, model[4],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[]=
```
{n → 22 529.1, c → 0.015874, b → 0.000303868}
```

In[]:=
```
upto = 750;
Show[
 Plot[model[4] /. ans[4], {t, 0, upto}, PlotRange → {{500, upto}, {0, 20 000}}],
 ListPlot[targetPeriod]
]
```

General: 0.000303868 $^{2037.07}$ 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．

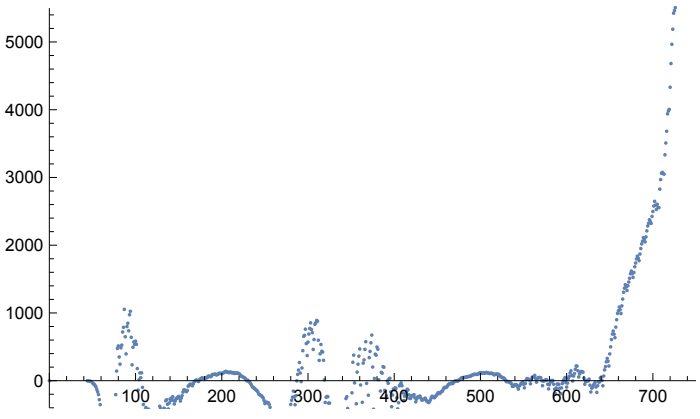
Out[]=



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び次のステップで用いる。

In[]:=
```
{pdays, death} = Transpose[rawModified[4]];
ListPlot[
 rawModified[5] = Transpose@{pdays, death - (model[4] /. ans[4] /. t → pdays)},
 PlotRange → {{000, upto}, {-400, 5500}}]
```

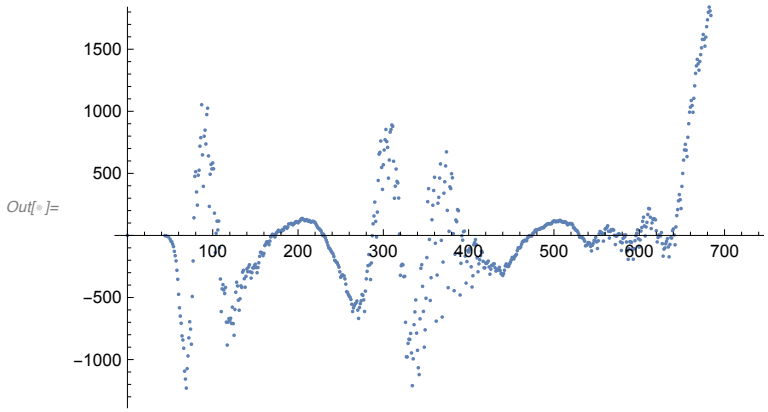
Out[]=



5回目 Gompertz 曲線

In[ ]:=
```
ListPlot[rawModified[5]]
```

Out[ ]=

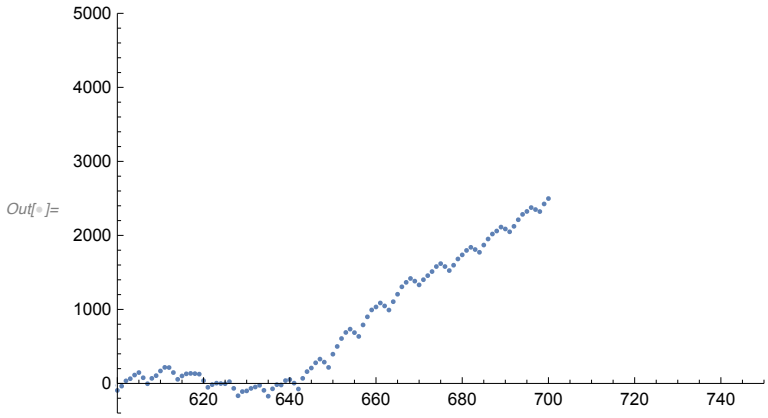


In[ ]:=
```
targetPeriod = rawModified[5];
```

1. ターゲットデータを切り分ける。

In[ ]:=
```
targetPeriod = Select[rawModified[5], #[[1]] ≤ 700 &];
ListPlot[targetPeriod, PlotRange → {{600, upto}, {-400, 5000}}]
```

Out[ ]=



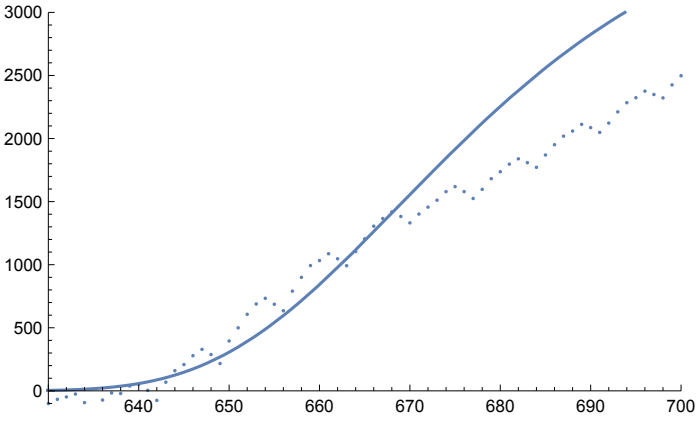
In[ ]:=

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 = 620;
inter = 700;
model[5] = n b^Exp[-c ( t - start)];
initialguess = {n → 4000, c → 0.05, b → 0.00001};
Show[
 Plot[model[5] /. initialguess,
  {t, 0, inter}, PlotRange → {{630, inter}, {-100, 3000}}],
 ListPlot[targetPeriod]
]
```

General: $0.00001^{2.90281\times10^{13}}$ 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、時系列をGompertz曲線にFittingさせる。

In[ ]:=
```
ans[5] = FindFit[targetPeriod, model[5],
   initialguess /. Rule → List, t, MaxIterations → 100]
```

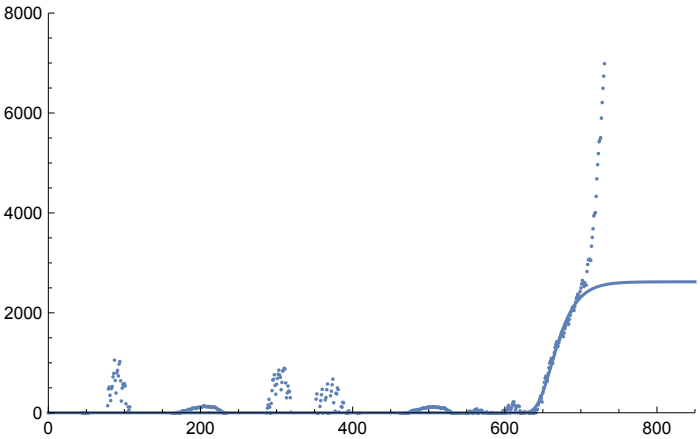
Out[ ]=
```
{n → 2621.1, c → 0.0539549, b → 0.0000899458}
```

In[ ]:=
```
upto = 850;
Show[
 Plot[model[5] /. ans[5], {t, 0, upto}, PlotRange → {{0, upto}, {0, 8000}}],
 ListPlot[rawModified[5]]
]
```

General:

$0.0000899458^{3.37007\times10^{14}}$ 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．

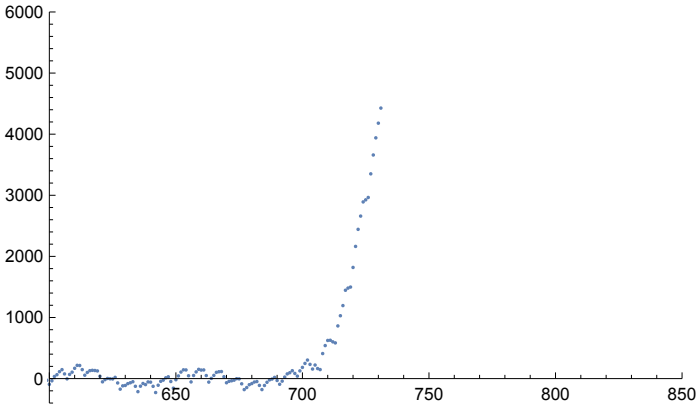
Out[ ]=



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[5]];
        ListPlot[
         rawModified[6] = Transpose@{pdays, death - (model[5] /. ans[5] /. t → pdays)},
         PlotRange → {{600, upto}, {-400, 6000}}]
```

Out[ ]=

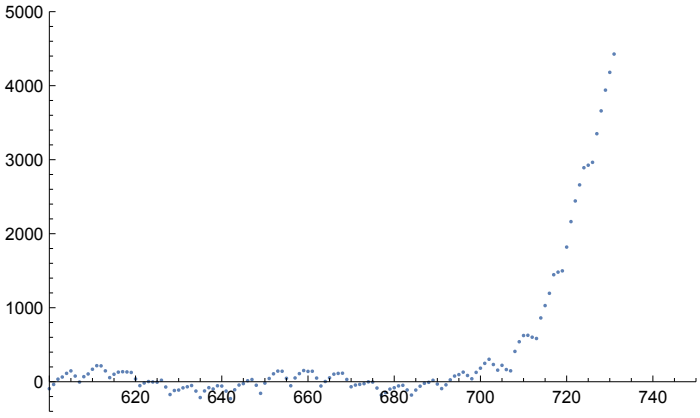


6回目 Gompertz 曲線

1. ターゲットデータ。

```
In[ ]:= targetPeriod = Select[rawModified[6], #[[1]] ≤ 800 &];
        ListPlot[targetPeriod, PlotRange → {{600, 750}, {-400, 5000}}]
```

Out[ ]=



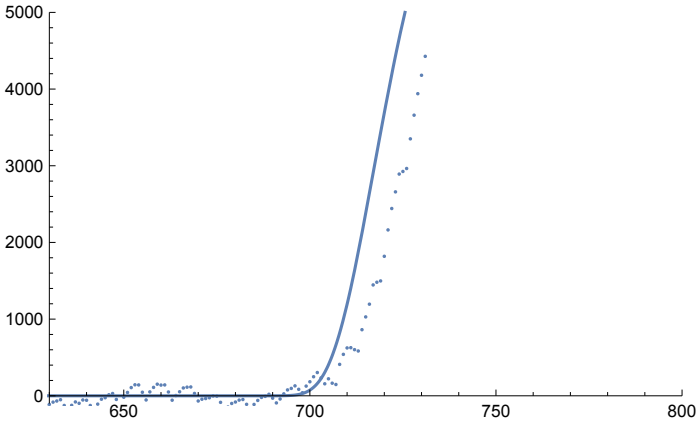

```
In[ ]:=
```

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 = 690;
inter = 800;
model[6] = n b^Exp[-c (t - start)];
initialguess = {n → 8000, c → 0.09, b → 0.00001};
Show[
 Plot[model[6] /. initialguess,
  {t, 0, inter}, PlotRange → {{630, inter}, {-100, 5000}}],
 ListPlot[targetPeriod]
]
```

General: $0.00001^{9.31212\times10^{26}}$ 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、時系列をGompertz曲線にFittingさせる。

In[ ]:=
```
ans[6] = FindFit[targetPeriod, model[6],
   initialguess /. Rule → List, t, MaxIterations → 100]
```

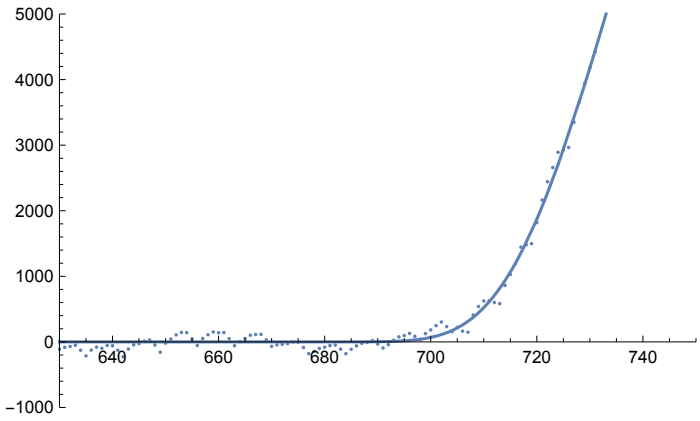
Out[ ]=
```
{n → 16 273.9, c → 0.0464073, b → 0.000166924}
```

In[ ]:=
```
upto = 850;
Show[
 Plot[model[6] /. ans[6], {t, 0, upto}, PlotRange → {{630, 750}, {-1000, 5000}}],
 ListPlot[targetPeriod]
]
```

General:

$0.000166924^{8.05752\times10^{13}}$ 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[ ]=

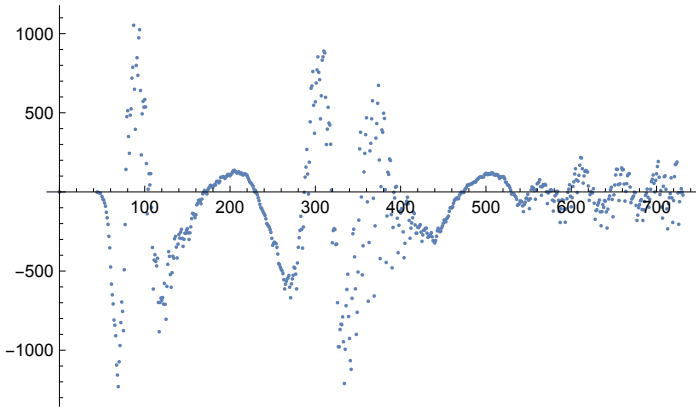


4. 時系列にfittingさせたGompertz曲線を用いて、報告時系列から、分割された時系列から推定した、推定死亡数時系列を差し引く。差し引いた時系列をモディファイド時系列と呼ぶ。

最後のモディファイド時系列は、これまでのFittinng操作の残差とも言える。残差を減らすために、残差の大きい部分を別の事象と捉えて、Fitting残差を減らすことも可能であるが、その操作が変異株の存在を主張するものとは、言えないだろう。

```
In[•]:= {pdays, death} = Transpose[rawModified[6]];
        ListPlot[
         rawModified[7] = Transpose@{pdays, death - (model[6] /. ans[6] /. t → pdays)}]
```

Out[•]=



全ての時系列へのフィッティング

得られたGompertz曲線とそのパラメータで、報告値を表現することが出来る。全ての部分モデルが繋ぎ合わされるのではなく、足し合わされる。即ち、全ての部分モデルの時間パラメータ t は、t→Infinityとして計算される。

```
In[•]:= percent =
         100 Divide[(model[1] /. ans[1]) + (model[2] /. ans[2]) + (model[3] /. ans[3]) +
              (model[4] /. ans[4]) + (model[5] /. ans[5]) +
              (model[6] /. ans[6]) /. t → Infinity, population]
Out[•]= 0.255897

In[•]:= percent * population / 100
Out[•]= 169 357.
```

In[ ]:=
```
fdate = First[ddata][[1]];
ldate = Last[ddata][[1]];
upto = 800;

Show[
 Plot[(model[1] /. ans[1]) + (model[2] /. ans[2]) + (model[3] /. ans[3]) +
    (model[4] /. ans[4]) + (model[5] /. ans[5]) + (model[6] /. ans[6]),
  {t, 0, upto}, PlotRange → {{0, upto}, {0, 180 000}}, 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$^{6.4954\times10^{9}}$ 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．

General: 0.000303868$^{2037.03}$ 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．

General: 0.0000899458$^{3.37026\times10^{14}}$ 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．

General: この計算中に，General::munflのこれ以上の出力は表示されません．

Out[ ]=