COVID-19 Death Number Modeling and Forecasting of German-2020V

kobayashikorio@gmail.com

2020/4/29

2020/8/12

2021/3/14

2021/3/21

2021/3/24

初期値をデータに近づけた

解析の方針

事象のモデリング

事象のモデリングとは、数学的手法つまり、客観的な表現によって、ある一定の誤差範囲で、事象を式により表現することである。これを数学モデルと呼ぶ。このとき、誤差とは、観測値と式から導かれた数の間の偏差のことである。誤差は事象が環境のランダムネスを反映し、モデルのパラメータが環境のランダムネスを反映しないために、時間と共に拡大する。

両者の偏差が一定の範囲に止まっている間、モデルは、事象を予測できるであろう。

さらに、筆者は以下のように主張する。COVID-19事象については、数学モデルは一貫しており、そのパラメータのみが変化する。誤差が一定であり続ける間、数学モデルは事象をよく説明し、よく説明する故に、このモデルは事象の本質である。

モデルの対象は死亡者数とした。感染者数は、consistency(一貫性)に乏しく、その適切な補正方法を見つけられなかったために、これを対象としなかった。

死亡方程式の導出

死亡者数を積算した時系列が、指数関数的に増加し続けることはないだろう。人類の感染症の歴史からみて、死亡者数の変化傾向は、事象の時間経過と共に必ず一定値に漸近するであろう。この考えの下に、事象を表現する微分方程式を導くことができる。

考え方については、Google Slide (https://docs.google.com/presentation/d/1M3wCr575y5J8-sb5MJhg97VINIMc1JOps2tLqErQZXKk/edit)に述べた。

漸近する一定値は０とは限らない。死亡者数が年齢をパラメータとして持つとき、漸近する一定値が０でないことがあり得る。すなわち、ある感染症が死亡原因の一つとなる場合である。

報告値に対するGompertz曲線フィッティング

上記の考え方の下、目的変数yを死亡者数とした微分方程式 dy ＝ Ａ y ^Exp(-Ｂt) dt の解は、Gompertz曲線である。Gompertz曲線の含意するところは、死亡者数の増加傾向は、死亡者数をパラメータとするゆえに、指数関数的に増大し、減少は、時間tだけをパラメータとする（時間のみをパラメータとして持つのであれば、減少はウィルス変異の集積を通じて起きると考えられよう（他に原因となる仮説が見つからないゆえに）。すなわち集団免疫成立の否認である）。

Gompertz曲線は、日本以外の殆どの国についてフィッティングできるのであるが、事象の初期において、日本のケースだけは適切なフィッティングが出来なかった。

日本のケースにフィッティング出来ない理由を仮説推定してみた。設定した仮説は、日本の死亡者数の時系列は、複数の独立事象の和である、とした。この仮説の採用で、現在までの時系列が表現できた（事象が複数の独立事象の和で表現できることは、目的変数の現象が時間のみの関数であることを含意する）。

計算のポリシー

事象がただ一つの曲線では、フィッティング出来ないとの気づきを得て、以下のポリシーにより事象をモデル化する。

1. モデルはGompertz曲線により与えられる
2. 一定の誤差に偏差を収めるために複数のGompertz曲線を用いる

計算の手順

事象が収束するまで、すなわち、観測値とモデルの偏差が一定の誤差以下になるまで、以下を続ける。

1. 観測値を与える
2. Fittingされた（Fitting関数がオーバーフロー、アンダーフローが起こさない）モデルで将来予測する
3. ヒューリスティックに初期値を与えた後、Fitting関数によって曲線のパラメータを求める
4. 得られたGompert曲線を元の観測値から差し引く
5. 誤差が一定範囲内から外れるなら、新規の事象が発生したと考え、新規事象の始まりを定義し、最初に戻る
6. 最初に戻る

計算途中の気づきなど

(2020/7/12)死亡原因の主たるウィルス株は交代すると考えたなら、報告死亡者数をモデルがよく表現できた。これは日本の例に明らかであるが、他の国の場合にも当てはまる（米国の事象を表現できる）。

(2020/8/12)日本の報告値が、時間経過と共に、2つのGompertz曲線の足し合わせでは、表現できなくなった。事象は、３つのGompertz曲線の足し合わせで表現できた。

(2020/10/24)日本の報告値が、時間経過と共に、複数のGompertz曲線の足し合わせでしか、表現できなくなった。この時点の事象は、異なるパラメータを持つ、４つのGompertz曲線の足し合わせで表現されている。事象の全体は、個々の事象の再帰的な表現であるとの考え方が、確からしさを増した。個々の事象の収束がウィルスの変異に依るものであるように、個々の事象の足し合わせである全体の事象についても、ウィルスのアイデンティティが保持されつつ、起きる変異は、有限である筈なので、全体の事象もまた、個々の事象が収束するように、一定の時間経過の後、収束するだろうからである。

(2020/12/2)日本のモデルに５番目の事象を加えた。

データの準備

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
(オプション)JSONファイルの国名は以下で読める。

In[ ]:=
```
Association[Import[urlCovid19]] // Keys
```

解析の開始

In[ ]:=
```
entity = "Germany";
population = Normal[Entity["Country", entity]["Population"]]
```

Out[ ]= 82 114 224

データを取得する。

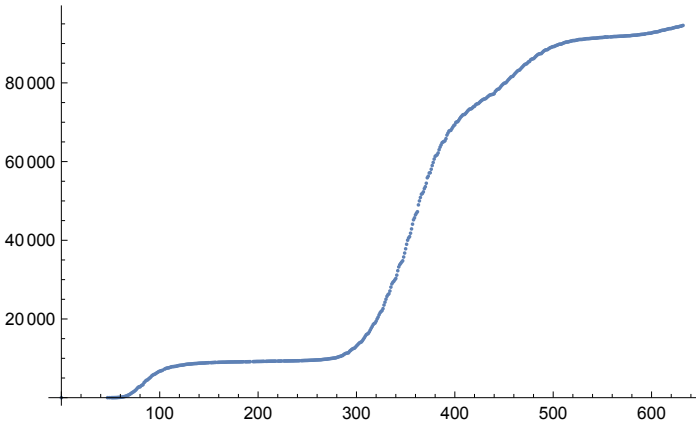
In[1]:=
```
country = "Germany";
population = 83 517 046;
urlCovid19 = "https://pomber.github.io/covid19/timeseries.json";
data = Map[Association, Association[Import[urlCovid19]][country], {1}];
ddata = {DateObject[#["date"]], #["deaths"]} & /@ data;
firstday = ddata[[1, 1]];
Last[ddata]
```

Out[6]= { Fri 15 Oct 2021 , 94 605 }

初回報告日からの経過日数と報告のなかった日のデータからの削除

In[7]:=
```
i4 = Map[{QuantityMagnitude[#[[1]] - firstday], #[[2]]} &, ddata];
raw = Map[First, Gather[i4, #1[[2]] == #2[[2]] &]];
rawModified[1] = raw;
ListPlot[raw]
```

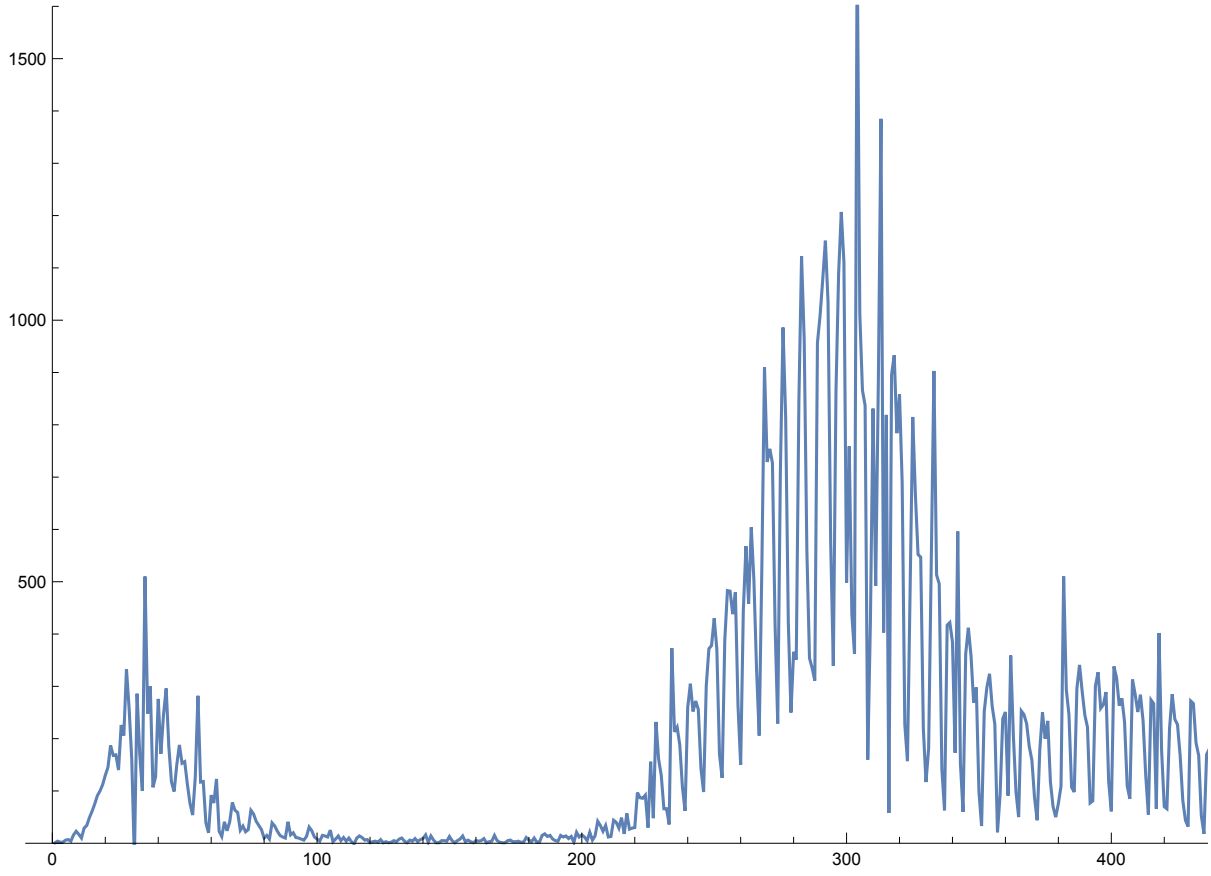
Out[10]=



In[ ]:=
```
delta = Apply[Subtract, Map[Reverse,
    BlockMap[Transpose, Select[raw, 30 < #[[1]] ≤ 550 &], 2, 1], {2}], {2}];
```

In[ ]:=
```
ListLinePlot[Map[Last, delta], PlotRange → {0, 1600}]
```

Out[ ]=



Iteration計算

1回目

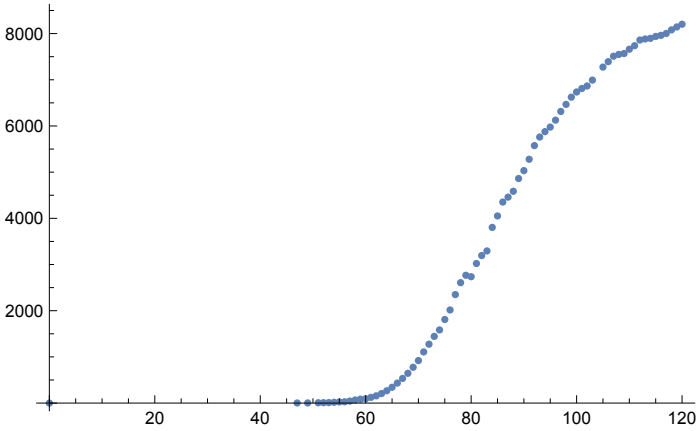
時系列の分割

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。

第一回は60日で切り分ける。

In[11]:=
```
rawModified[1] = raw;
targetPeriod = Select[rawModified[1], #[[1]] ≤ 120 &];
ListPlot[targetPeriod]
```

Out[13]=

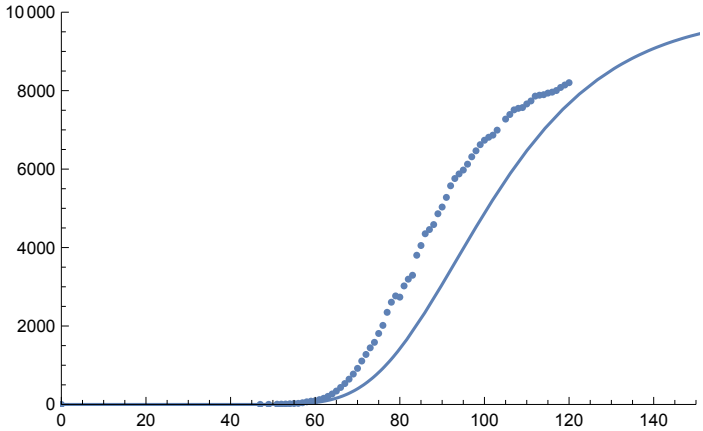


時系列へのマッチングの初期値設定

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。すなわち、試行錯誤を繰り返して、Gompertz曲線が、報告された時系列に近くなるようにする。

In[14]:=
```
start = 70;
model[1] = n b^Exp[-c (t - start)];
initialguess = {n → 10 000, c → 0.05, b → 0.04};
Show[
 Plot[model[1] /. initialguess, {t, 0, 200}, PlotRange → {{0, 150}, {0, 10 000}}],
 ListPlot[targetPeriod]
]
```

Out[17]=



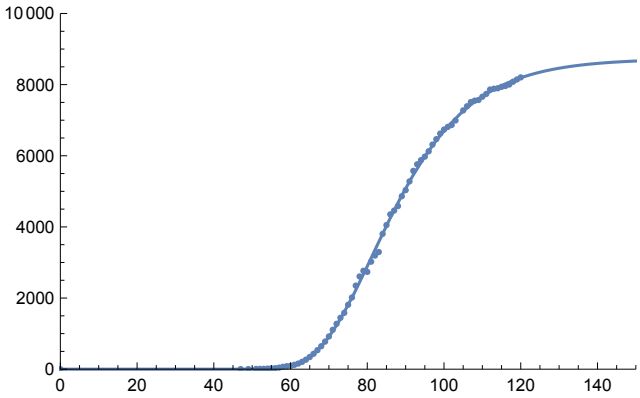
Fitting関数を用い、与えた初期値でパラメータを推定し、最初の時系列をGompertz曲線にFittingさせる。

In[18]:=
```
ans[1] = FindFit[targetPeriod, model[1],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[18]=
```
{n → 8728.04, c → 0.0714435, b → 0.104764}
```

In[19]:=
```
Show[
 Plot[model[1] /. ans[1], {t, 0, 150}, PlotRange → {{0, 150}, {0, 10 000}}],
 ListPlot[targetPeriod]
]
```

Out[19]=



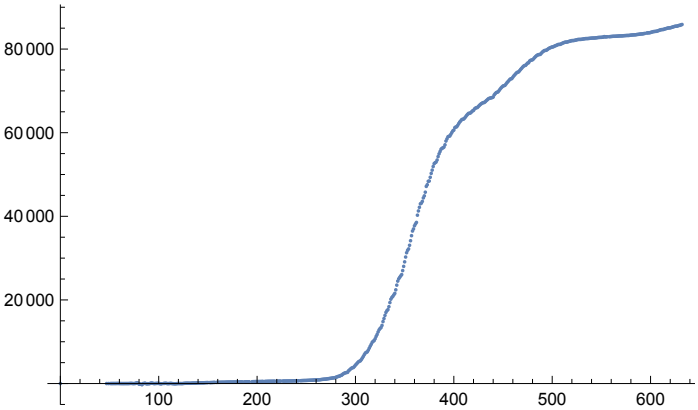
報告時系列のモディファイ

報告開始日から60日目までを最初のウィルス株による結果、それ以降を次のウィルス株による結果と考える。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てある。

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。つまり、ウィルス株の影響を差し引く。Fittingが成功しているなら、報告時系列の当該部分は０となるであろう。差し引いた時系列をモディファイド時系列と呼ぶ。

In[20]:=
```
{pdays, death} = Transpose[rawModified[1]];
ListPlot[
 rawModified[2] = Transpose@{pdays, death - (model[1] /. ans[1] /. t → pdays)}]
```

Out[21]=



日以前のデータの時系列が0にされた、rawModified[2]が生成された。

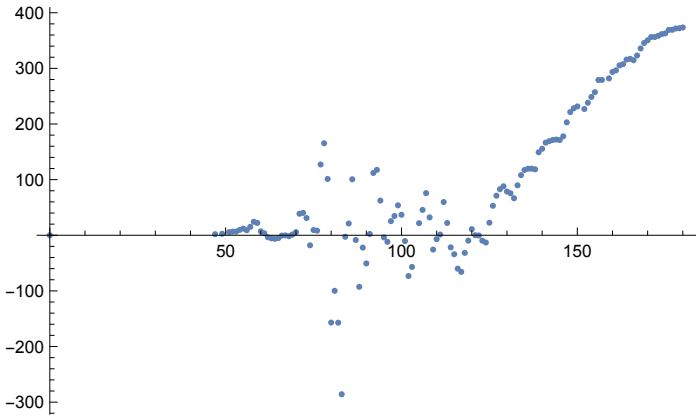
In[22]:=

2回目

第２回はx日で切り分けて、着目するのがtargetPeriod である。

In[23]:=
```
targetPeriod = Select[rawModified[2], #[[1]] ≤ 180 &];
ListPlot[targetPeriod]
```

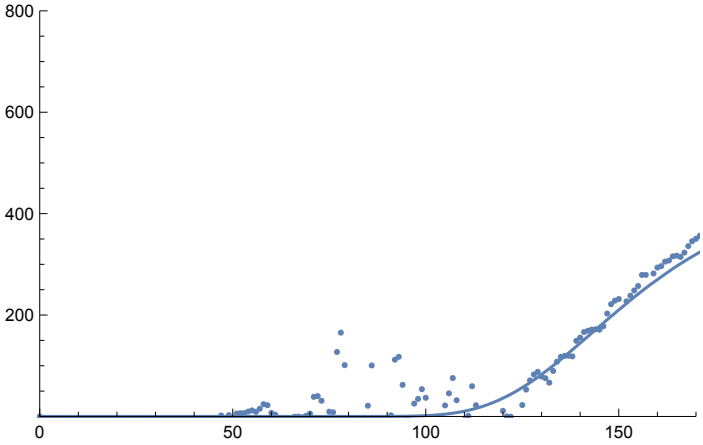
Out[24]=



二番目の時系列をGompertz曲線にFittingさせるために、ヒューリスティックに初期値を与える。

In[25]:=
```
start = 120;
model[2] = n b^Exp[-c (t - start)];
initialguess = {n → 450, c → 0.04, b → 0.08};
Show[
 Plot[model[2] /. initialguess, {t, 0, 1000}, PlotRange → {{0, 170}, {0, 800}}],
 ListPlot[targetPeriod]
]
```

Out[28]=



初期値を与えて、ターゲットとなる時系列をGompertz曲線にFittingさせる。

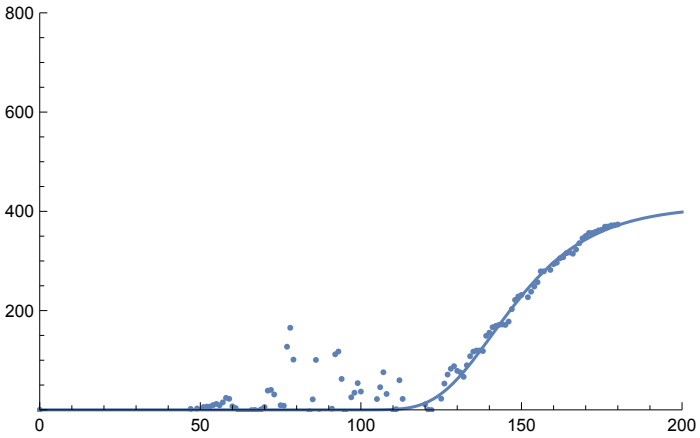
In[29]:=
```
ans[2] = FindFit[targetPeriod, model[2],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[29]=
```
{n → 410.996, c → 0.0588294, b → 0.0332348}
```

In[30]:=
```
upto = 200;
Show[
 Plot[model[2] /. ans[2], {t, 0, upto}, PlotRange → {{0, 200}, {0, 800}}],
 ListPlot[targetPeriod]
]
```

General: 0.0332348$^{1163.62}$ 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[31]=

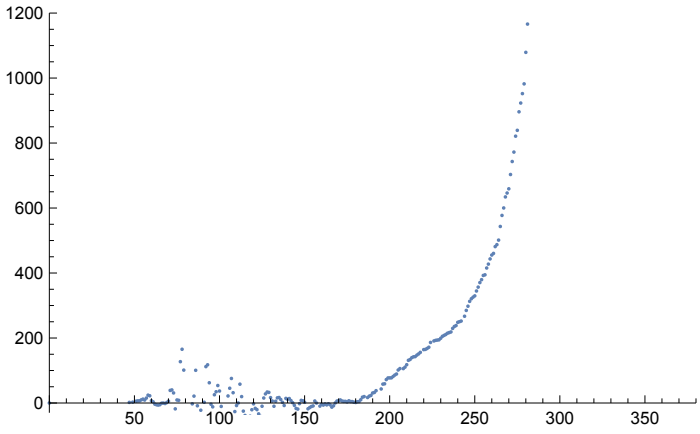


報告時系列のモディファイ

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。つまり、ウィルス株の影響を差し引く。Fittingが成功しているなら、報告時系列の当該部分は０となるであろう。rawModified[3] が生成される。

In[32]:=
```
{pdays, death} = Transpose[rawModified[2]];
ListPlot[
 rawModified[3] = Transpose@{pdays, death - (model[2] /. ans[2] /. t → pdays)},
 PlotRange → {{0, 380}, {-30, 1200}}]
```

Out[33]=



In[34]:=

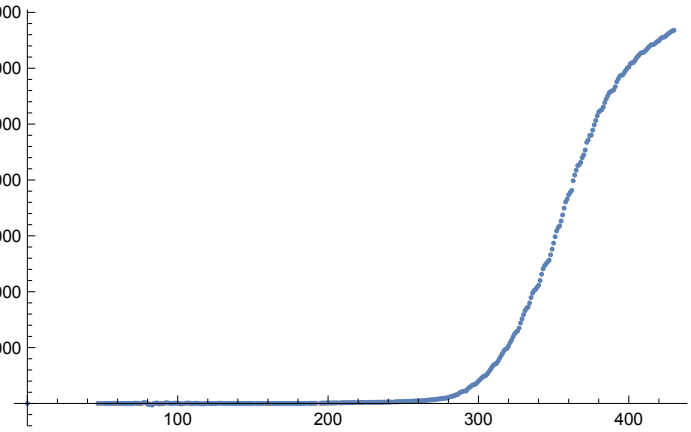
3回目

残った時系列は、これまでに推定した時系列を差し引いた、ターゲット時系列である。ヒューリスティックに初期値を与えて、Fittingする。

第3回はx日で切り分けて、着目するのがtargetPeriod である。

In[35]:=
```
targetPeriod = Select[rawModified[3], #[[1]] ≤ 430 &];
ListPlot[targetPeriod]
```

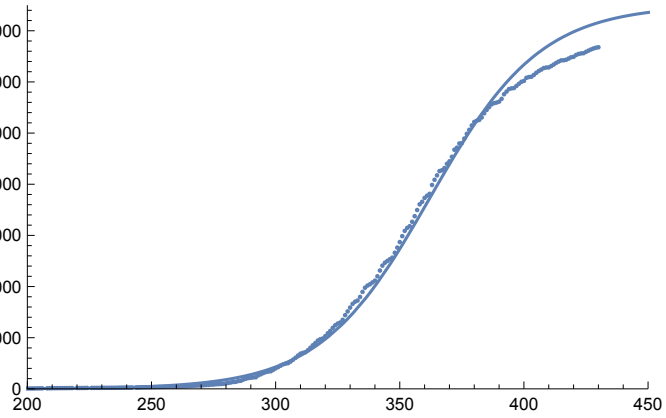
Out[36]=



In[37]:=
```
start = 260;
model[3] = n / (1 + c Exp[-a n (t - start)]);
initialguess = {n → 75000, c → 100, a → 0.0000006}
Show[
 Plot[model[3] /. initialguess,
  {t, 0, 1000}, PlotRange → {{200, 450}, {0, 75000}}],
 ListPlot[targetPeriod]]
```

Out[39]= $\{n \to 75000,\ c \to 100,\ a \to 6. \times 10^{-7}\}$

Out[40]=

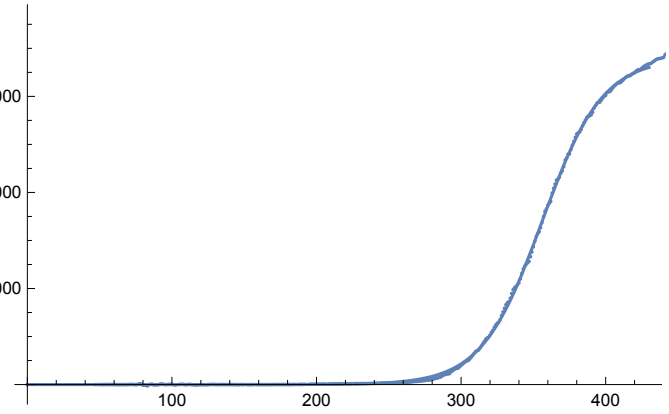


In[41]:=
```
ans[3] = FindFit[targetPeriod, model[3],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[41]= $\{n \to 67921.,\ c \to 100.836,\ a \to 7.07315 \times 10^{-7}\}$

In[42]:=
```
upto = 430;
Show[
 ListPlot[rawModified[3]],
 Plot[model[3] /. ans[3], {t, 0, upto}],
 PlotRange → {{0, upto}, {0, 75 000}}]
```

Out[43]=

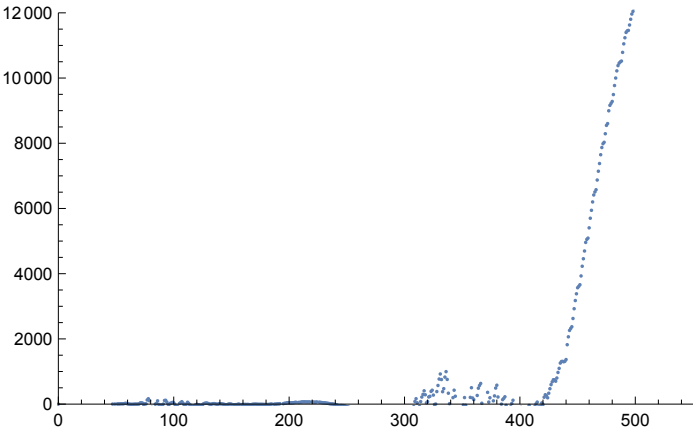


報告時系列のモディファイ

rawModified[4] が生成される。

In[44]:=
```
{pdays, death} = Transpose[rawModified[3]];
ListPlot[
 rawModified[4] = Transpose@{pdays, death - (model[3] /. ans[3] /. t → pdays)},
 PlotRange → {{0, 550}, {-30, 12 000}}]
```

Out[45]=



In[ ]:=

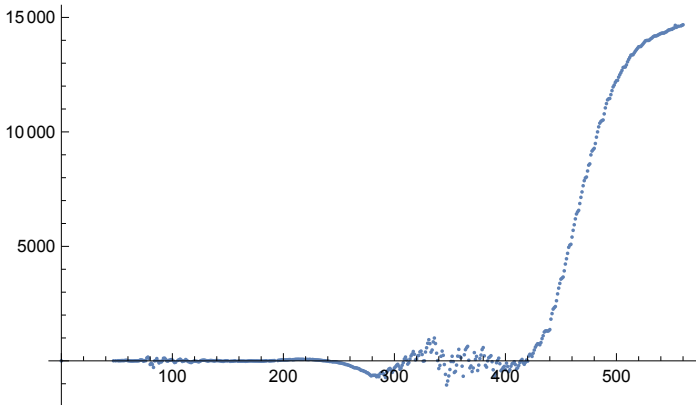
4回目

start地点から前のデータをノイズとみなす。

Logistic curve

In[46]:=
```
targetPeriod = Select[rawModified[4], #[[1]] <= (560) &];
ListPlot[targetPeriod]
```

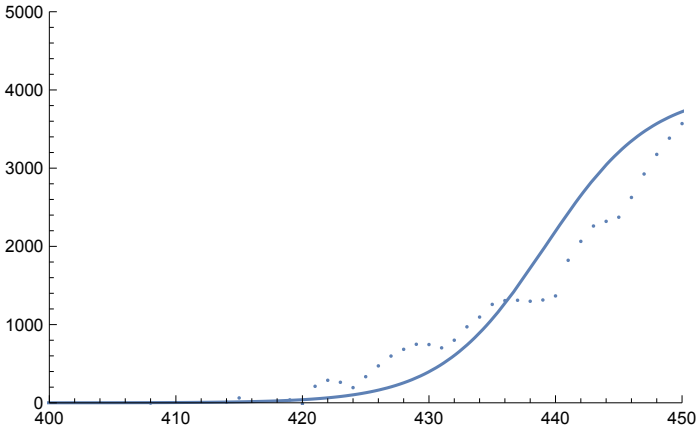
Out[47]=



In[48]:=
```
start = 420;
model[4] = n / (1 + c Exp[-a n (t - start)]);
initialguess = {n → 4000, c → 100, a → 0.00006}
Show[
 Plot[model[4] /. initialguess,
  {t, 0, 1000}, PlotRange → {{400, 450}, {0, 5000}}],
 ListPlot[targetPeriod]]
```

Out[50]= $\{n \to 4000, c \to 100, a \to 0.00006\}$

Out[51]=



In[52]:=
```
ans[4] = FindFit[targetPeriod, model[4],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

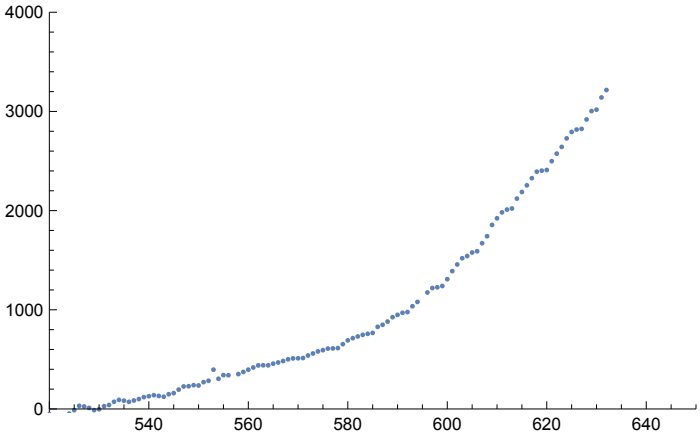
Out[52]= $\left\{n \to 14\,329.6,\ c \to 22.9594,\ a \to 4.42136 \times 10^{-6}\right\}$

報告時系列のモディファイ

rawModified[5] が生成される。

In[53]:=
```
{pdays, death} = Transpose[rawModified[4]];
ListPlot[
 rawModified[5] = Transpose@{pdays, death - (model[4] /. ans[4] /. t → pdays)},
 PlotRange → {{520, 650}, {-30, 4000}}]
```

Out[54]=



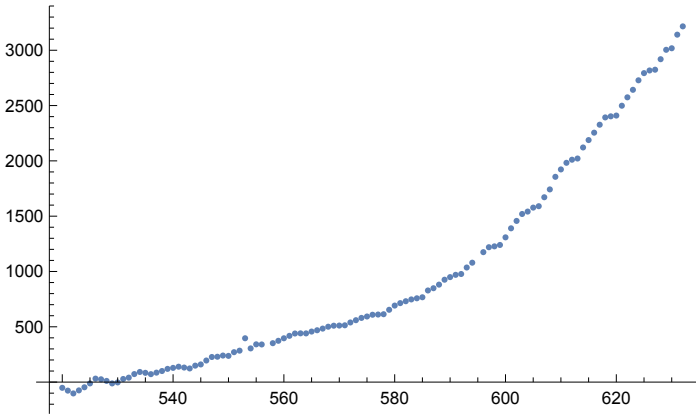
5回目

start地点から前のデータをノイズとみなす。

Logistic curve

In[55]:=
```
targetPeriod = Select[rawModified[5], #[[1]] ≥ (520) &];
ListPlot[targetPeriod]
```

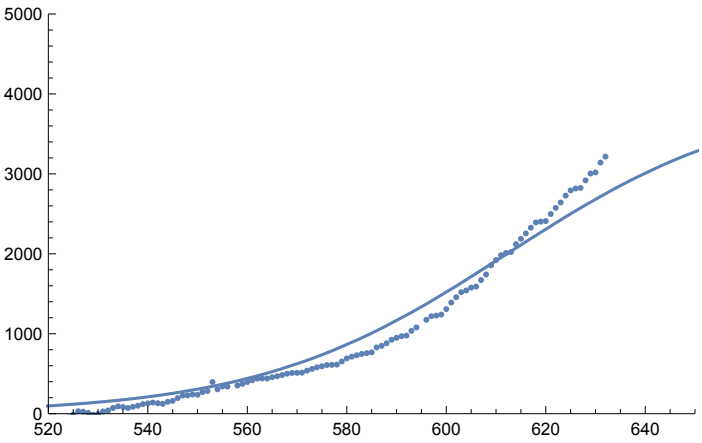
Out[56]=



In[57]:=
```
start = 520;
model[5] = n / (1 + c Exp[-a n (t - start)]);
initialguess = {n → 4000, c → 40, a → 0.00001}
Show[
 Plot[model[5] /. initialguess,
  {t, 0, 1000}, PlotRange → {{520, 650}, {0, 5000}}],
 ListPlot[targetPeriod]]
```

Out[59]=
```
{n → 4000, c → 40, a → 0.00001}
```

Out[60]=



In[61]:=
```
ans[5] = FindFit[targetPeriod, model[5],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[61]=
$$\{n \to 6313.28,\ c \to 96.7156,\ a \to 6.51272 \times 10^{-6}\}$$

計算結果と評価

人口に対する死亡率の推定

In[62]:=
```
percent =
 100 Divide[(model[1] /. ans[1]) + (model[2] /. ans[2]) + (model[3] /. ans[3]) +
      (model[4] /. ans[4]) + (model[5] /. ans[5]) /. t → Infinity, population]
```

Out[62]=
```
0.116986
```

In[63]:=
```
population * percent / 100
```

Out[63]=
```
97 703.
```

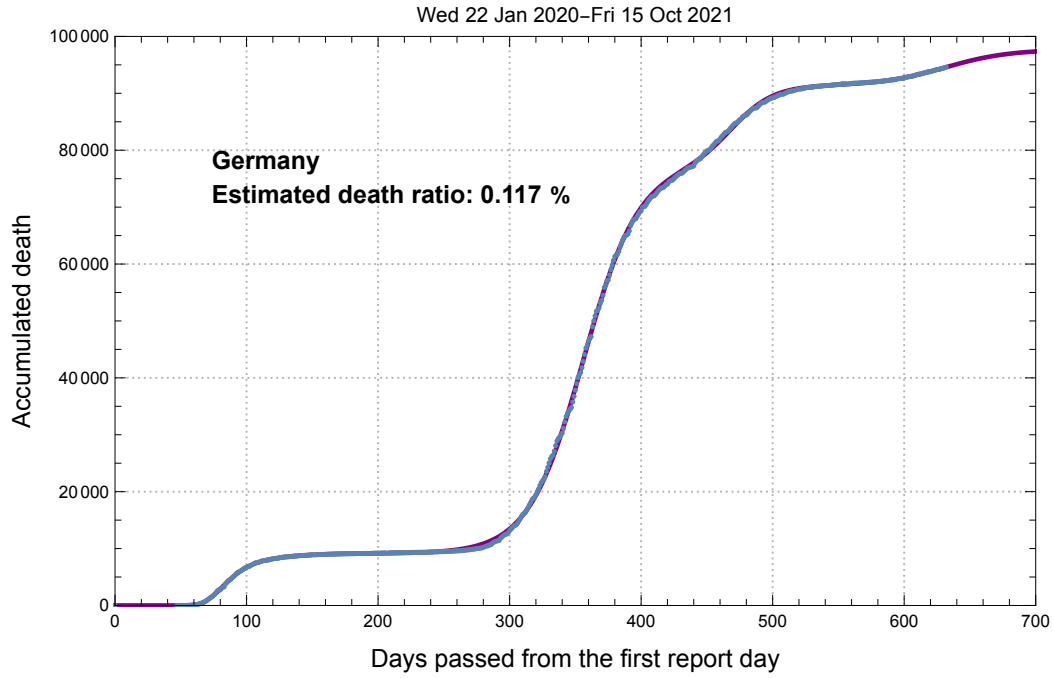
全ての時系列へのフィッティング

得られた4つのGompertz曲線の和と報告時系列を表示する。

In[64]:=
```
fdate = First[ddata][[1]];
ldate = Last[ddata][[1]];

upto = 700;

Show[
 Plot[(model[1] /. ans[1]) + (model[2] /. ans[2]) + (model[3] /. ans[3]) +
    (model[4] /. ans[4]) + (model[5] /. ans[5]), {t, 0, upto}, PlotTheme → "Grid",
  PlotRange → {{0, upto}, {0, 100 000}}, PlotStyle → {Purple, 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: 0.0332348$^{1162.92}$ 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．

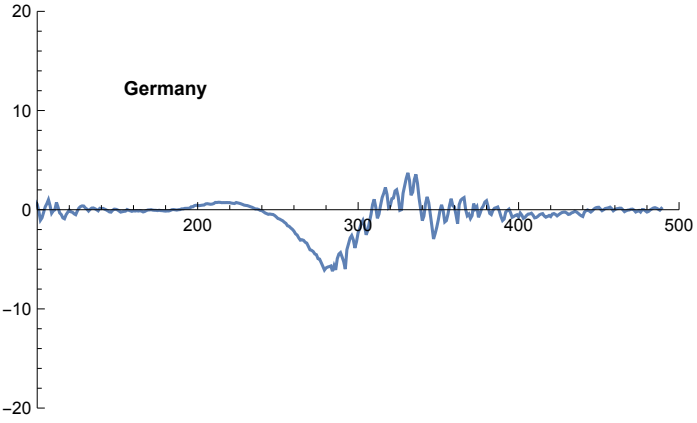
Out[67]=



モデルと報告値の偏差

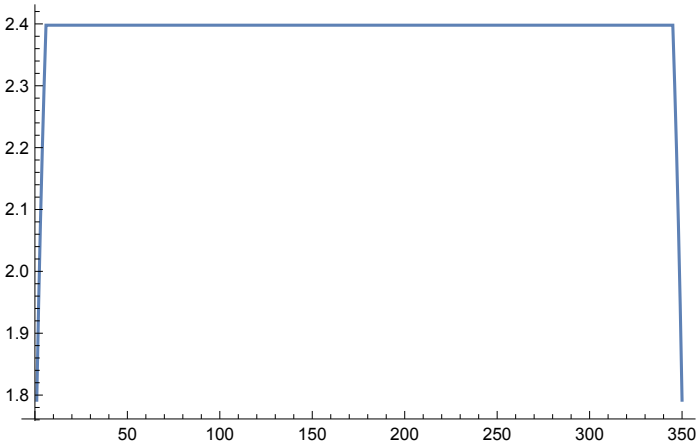
```
In[ ]:= Clear[et]

In[ ]:= {pdays, death} = Transpose[raw];
        diff =
          Transpose@{pdays, 100 * (death - (et = (model[1] /. ans[1]) + (model[2] /. ans[2]) +
                      (model[3] /. ans[3]) + (model[4] /. ans[4]) /. t → pdays)) / et};

        ListLinePlot[diff,
         PlotRange → {{100, 500}, {-20, 20}},
         PlotLegends → Placed[Text[Style[country, Bold]], {0.2, 0.8}]]
```

Out[ ]=




```
In[ ]:= filter = EntropyFilter[Transpose[raw][[2]], 5] // N

In[ ]:= ListLinePlot[filter]
```

Out[ ]=



まとめ

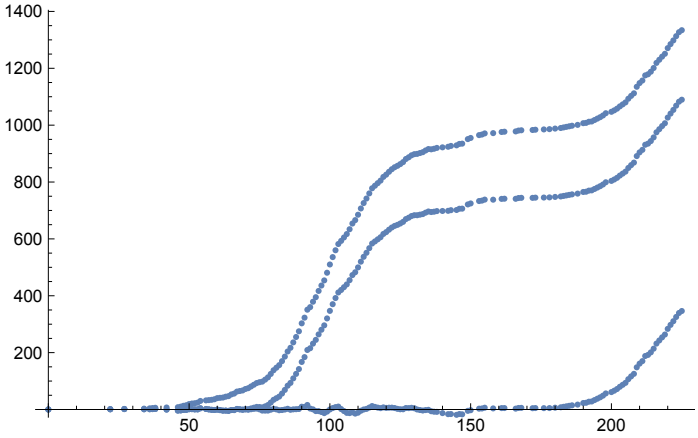
報告時系列をモディファイ（時系列からモデル推定値を差し引いた残差を新たな時系列とする）する操作の図解（3回までの時）

例えば、報告時系列を、事象の始まりの部分から順に、Gompertz曲線にフィッティングさせ

て、フィッティング結果を報告時系列から差し引く操作を２回実行する。フィッティング操作は３回実行する。

```
In[ ]:= Show[ListPlot[raw], ListPlot[rawModified[2]],
         ListPlot[rawModified[3], PlotRange → All]]
```

Out[ ]=



報告時系列にGompertz曲線をフィッティングさせる操作のまとめ

1. 事象を、微分方程式の解を用いてモデル化することと、多項式を用いてフィッティングすることは、全く異なるモデリングの方法である。微分方程式は、事象が物理過程に従うという確信の表現であるが、多項式が事象の表現であるとは言えない。

2. 報告数の時系列をモデル化するためには、時系列を、適切と考えられる曲線（この事象に対しては、微分方程式の解であるGompertz曲線）にフィッティングさせる必要がある。Gompertz曲線は非線形であるので、FindFit関数を適用するためには、初期値を必要とする。

3. イタレーションを繰り返しても、フィッティングできない場合、対象とする事象が、想定した方程式が表現する物理過程と異なる、と考えるべきである（上記の例で言えば、報告時系列を分割しない限り、単一のGompertz曲線で事象に適切なフィッティングを得ることは不可能である）。

4. モデルと報告値の偏差の、報告値に対する割合によって、モデルの精度を推定することができる。

5. 事象の報告値にはノイズが含まれているが、例えば、事象開始後145日目に起きたディップを、有意な信号か否かは判然としない。ただし、モデルと報告値の偏差の時系列が、何らかのインディケーションを示す可能性は残る。

EOF

■配列A，BにおけるEER計算方法
配列A　Aの出現確率 = 3 / 10 Bの出現確率 = 3 / 10 Cの出現確率 = 4 / 10
配列B　Aの出現確率 = 6 / 10 Bの出現確率 = 4 / 10

配列A　AAABBBCCCC
配列B　AAAAAABBBB

$SE(P) = -\sum P(X) \log P(X)$