COVID-20 Death Number Modeling and Forecasting of Japan Case

kobayashikorio@gmail.com

2020/4/29
2020/8/12
2021/7/3
2021/10/13
2022/6/27
2022/7/28
2022/11/14
03/17/2023

目的
COVID-19事象を数理モデルにより表現し、ウイルスの動態を理解するための手立てを提供するとともに、将来の現象を予測する。

対象事象とモデリング

事象の概要

COVID-19事象とは集団免疫の期待できない混乱状態にあった中で、mRNAワクチンがdeus ex māchinā として出現し、世界の混乱を一気に収束させるという物語である。

事象のモデリングの考え方

本事象のモデリングには、仮説推論を用いる。仮説推論（アブダクション） とは、「関連する証拠を （真である場合に） 最もよく説明する仮説を選択する推論法である。アブダクションは観察された事実の集合から出発し、それらの事実を、尤もらしく、ないしは最良の説明へと推論する」 方法である （Wikipedia）。

仮説推論１

仮説：人類は生き延びてきたので、事象は必ず収束する
推論：発生し収束する事象であるなら、事象は数学モデル、いわゆる成長曲線で表現されるだろう

仮説推論２

仮説：COVID-19事象（感染者数や死亡者数のパターン）は、ランダムに見える個々の事象（感染者や死亡者の惹起）の、積算により表現される

推論：個々の事象はあるパターンあるいは微分方程式で示される原理に、従うだろう

仮説推論３

仮説：COVID-19事象はある微分方程式で表現されるだろう
推論：推定された微分方程式は、事象の特徴（集団免疫の有無など）を表現しているであろう

事象の数理モデル

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。

事象の変化が緩やかであり、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、二種類の数学モデルで近似できる複数の事象の組み合わせである。ある事象を説明する数学モデルは一貫しており、モデルを決定するパラメータのみが変化する。誤差が一定であり続ける間、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。

数学モデルの概要

目的変数

モデルの対象データは死亡者数とした。感染者数は、データとしてのconsistency(一貫性)に乏しく、元データから真の感染者数を推定する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。

収束する死亡方程式

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と期待できる。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、以下にも紹介してある。
Google Slide に
1. COVID-19 Model Development & Deployment
(https://docs.google.com/presentation/d/1M3wCr575y5J8sb5MJhg97VINIMc1JOps2tLqErQZXKk/edit#slide=id.g846481d540_0_0)

Wokfram Community に
3. COVID - 19 superposition principle death model for the US event
https://community.wolfram.com/groups/-/m/t/2046331?p_p_auth=f88O1nBk
2. COVID-19 complex death model for Japan

https://community.wolfram.com/groups/-/m/t/1959251?p_p_auth=f88O1nBk
1. COVID-19 death model by the Gompertz curve for Italy, Spain, and US
https://community.wolfram.com/groups/-/m/t/1939307?p_p_auth=f88O1nBk

死亡方程式として、事象の初期にはLogistic曲線を考えたが、直ぐに、報告される積算死亡者数を適切に表現できなくなった。そこで、Gompertz曲線が導入された。Gompertz曲線は、集団免疫の存在を否定を含意するものである。ワクチンが登場した後、事象モデルは変化する可能性がある。

ただし、どちらの曲線についても、報告される死亡者数の積算値をモデル化するためには、モデルが非線形方程式であるために、フィッティング関数の初期値を、ヒューリスティックに与える必要がる。適切な推定計算回数でフィッティングするためには、ある程度報告死者数の積算値に、モデル計算値を近づけておく必要があるから。

ロジスティック曲線～収束状態の獲得モデル

死亡者数の時間的な変化を与える方程式を考える（一般的なロジスティック曲線を与える微分方程式を採用してみる）。目的変数の累積をy 、目的変数に変化する可能性を持つクラスタをN とする。感染について、ある人の死亡は、まわりで死亡した人の数に比例する（回りが感染していると自分も感染しやすくなる）とすると、死亡者数をy、比例係数を a として、potential-death-number人をNとすると、これから感染して死亡する可能性のある人数は N-y であるから、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 は、dy= a y(N-y) dt　と表わされるであろう。これから、dy/dt= a y(N-y)　が導かれる。

すなわち、この微分方程式は含意として、無限時間後にクラスタは全て感染し、そのうち一定割合が死亡することを前提としている。厚生省の定義では、集団免疫は感染に関わる定義として用いられているので、感染者と死亡者の関係が一意的で無いことを考慮すると、無限時間後の式の値は、事象の収束であると捉えた方が良いだろう。

なお、ワクチンの作用とは、非感染者に、ウィルスを感染させたのちに治癒している状態を、人為的に作り出すことである。そしてCODIV-19ワクチンのない現状において(2020/03現在)「The UK backs away from “herd immunity” coronavirus proposal amid blowback」
https://www.vox.com/world/2020/3/15/21180414/coronavirus-uk-herd-immunity-vallance-johnson として英国が取っている方法の結果に、感染者数モデルがよくマッチングしていることが確認できるであろう。

死亡者数微分方程式の解法～ロジスティック曲線

dy/dt= a y(N-y) の解としてロジスティック曲線が与えられる。

```
In[•]:= DSolve[{y'[t] == a * y[t] (N - y[t])}, y[t], t]
```

$$\text{Out[•]}= \left\{\left\{y[t] \to \frac{e^{a N t+N C[1]} N}{-1+e^{a N t+N C[1]}}\right\}\right\}$$

簡単化してみる

```
In[•]:= {y[t] → e^(a N t+N C[1]) N / (-1 + e^(a N t+N C[1]))} /. e^(a N t+N C[1]) → K
Out[•]= {y[t] → K N / (-1 + K)}
```

右辺を簡単化する

```
y = N / (1 - 1 / K)
```

```
y = N / (1 - Exp[-N * C] Exp[-a * t * N])
```

定数Exp[-N*C] を改めて、-cとおけば、y (t) = N/(1 + C Exp (-a N t)) となり （Nもnとしておく）、以下のモデルが得られる。

```
y (t) = n / (1 + c Exp[-a n (t)])
```

ここでcは定数、yは時刻tまでの死亡者数、nは死亡者者数でスケーリングした死亡し得る人数、aは感染後の死亡係数である。このような関数形から得られる曲線は、ロジスティック曲線となる。この曲線は変曲点で点対称となるので、曲線とデータとの間のマッチングを検知することができる。

報告値に対するLogistic曲線フィッティング

dy ＝ Ａ y (N-y) dt の解は、Logistic曲線である。Logistic曲線の含意するところは、y ->N となる時、yの増加率は０となる。死亡者は感染者の一部であり、全てが感染すれば、死亡者の増加は停止するという方程式である（すなわち事象の収束である）。

```
y (t) = n / (1 + c Exp[-a n (t)])
```

ここで、
y：時刻tまでの死亡者数
n: 死亡し得る人数
a: 感染後の死亡係数
c: 定数

無限時間後にy(t)はnに収束する。すなわち事象により死亡者数が関連する全ての人に及んだ時、死亡は止まる。事象の収束である。

死亡者数微分方程式の解法〜Gompertz曲線

別のいわゆる成長曲線も想定できる。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。

y = n b^Exp[-c t]

ここで、
y : 時刻 t までの死亡者数
n : 死亡し得る人数
b : 死亡係数
c : 減少定数

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**集団免疫の存在は否定される**）。

報告値が提示されるごとに、ヒューリスティックに与えてある初期値を用いて、Mathematicaにより報告値にFittingするモデルのパラメータを計算し（初期値は変える必要がない）、モデルと最新データが示す将来値を推定する。つまり、初期値から出発して、パラメータの修正だけで報告値を説明できる（両者の偏差がある誤差範囲にある）間は、モデルは現状を説明できると考える。

解析途中の気づき～まとめ～

仮説推論

仮説：ある国のCOVID-19事象は、複数の事象で構成されている
推論：全体事象は独立な事象のモデルの足し合わせで表されるであろう
仮説：独立な事象だとすると、その事象を説明する微分方程式は、集団免疫の獲得を含意しない
推論：集団免疫の存在を含意しない、Gompertz曲線を導く微分方程式が、事象を説明できるだろう
仮説：集団免疫の存在なしに時間のみに依存する収束は、ウイルスの変異株の出現を含意する
推論：変異株が存在し、集団免疫が存在しないなら、事象はウィルスのアイデンティティーが失われない限り、無限に継続するだろう

仮説推論

仮説：COVID-19の全体事象は、複数の事象モデルの結合により表現される
推論：各事象は、ある優勢株の出現に始まり、この株による事象が新たな変異株によって交代されるまで続くだろう

モデルは全ての変異株に対応するのではなく、その直前のモデルでは表現できないような、優勢な変異株の出現があった時に、新たな事象モデルを追加することで、対応することとする。このため、優勢株と見なされない中立的な変異株による事象が、独立したモデルではなく、その前後の別の優勢株のモデルに含まれることがある。

こうして全体の事象は、ある優勢な変異株を原因とするものの影響に、次に出現した優勢な変異株による影響が、次々に加わったものとなる。すなわち全体事象は、個々の事象の足し合わせで表されると考える。すなわち、各事象は互いに独立である。

仮説推論に導かれたモデリングの方針

事象を、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. 最初のモデルはGompertz曲線により与えられる
2. 事象は複数の独立な事象の重ね合わせである（単一の事象と考えるとモデルが発散してしまう）
3. 複数の事象には、それぞれの事象の出発点がある
4. ある曲線がモデリングする事象の主要原因は、単一変異株のウィルス主体であると考える
5. 誤差の偏差が一定の値から逸脱するのは新たな変異株の出現と考え、新たな曲線を与えるべきである

6. 非ワクチンクラスタ（ワクチンが射てない、あるいは射たないグループ）を主体としたクラスタの感染事象は、Gompertz曲線で表現されるだろう。

周辺情報：ワクチンによる免疫の構築

対象としているCOVID-19事象は、事象の観察中にウイルスの変異株の出現が、事象に重要な影響を与えていることが知られる様になった。であれば、事象の出現後に現れたワクチンが、事象にどのような影響を与え得るかは、考察に含まれなければならない。そこで、現在、世界各地で接種されているワクチンの種類と機序について調査した。

ワクチンの形式	メーカー（名称）	機序	活性化され免疫システ
弱毒化 不活化	・シノファーム ・シノバック	不活化ウィルスを 免疫システムに提示	液性免疫
組み替えタンパク	・ノババックス（武田薬品） ・シオノギ	ウィルスタンパクを免疫システムに提示 。 注射部位で抗原提示細胞を刺激するアジュバントが必須	（情報収集中
ウィルスベクター	・アストラゼネカ ・ヤンセン ・（スプートニクV）	ウィルスベクターが細胞質に導入した プラスミドDNAから リボソームがウィルスタンパクを生成	液性免疫と 細胞性免疫
mRNA	・ファイザー ・モデルナ	脂質粒子（LNP）で包まれたmRNAを 細胞に送り込み（エンドサイトーシス）、mRNAから リボソームがウィルス・スパイクタンパクを生成	液性免疫と 細胞性免疫

Reference
1. 新型コロナワクチンの開発状況について｜厚生労働省 -> https://www.mhlw.go.jp/stf/seisakunitsuite/bunya/0000121431_00223.html
2. 日本癌治療学会：Q６：COVID-19ワクチンの種類、効果、副反応は -> http://www.jsco.or.jp/jpn/index/page/id/2383
3. Chem-Station -> https://www.chem-station.com/chemglossary/biochem/2021/01/mrnavaccine.html
4. GSK、ノババックス社の新型コロナウイルス感染症（COVID-19）ワクチンの製造を支援 -> https://jp.gsk.com/jp/media/press-releases/2021/20210406_gsk-to-support-manufacture-of-novavax-covid-19-vaccine/
5. 【高校生物】細胞16 エンドサイトーシスとエキソサイトーシス -> https://www.youtube.com/watch?v=Q2c_4lYpNJ4
6. mRNA医薬に利用されるキャリア開発：ナノミセル -> https://www.jstage.jst.go.jp/article/dds/35/1/35_27/_article/-char/ja/
7. 新型コロナのワクチン、打った方が良い？〜mRNAワクチンの効果と安全性、よくある誤解 -> https://www.fizz-di.jp/archives/1078840555.html
8. 工業製品としてのmRNAワクチン -> https://iblc.co.jp/column/047/
9. 獲得免疫における細胞性免疫とは？ -> https://numon.pdbj.org/mom/246?l=ja

計算の手順

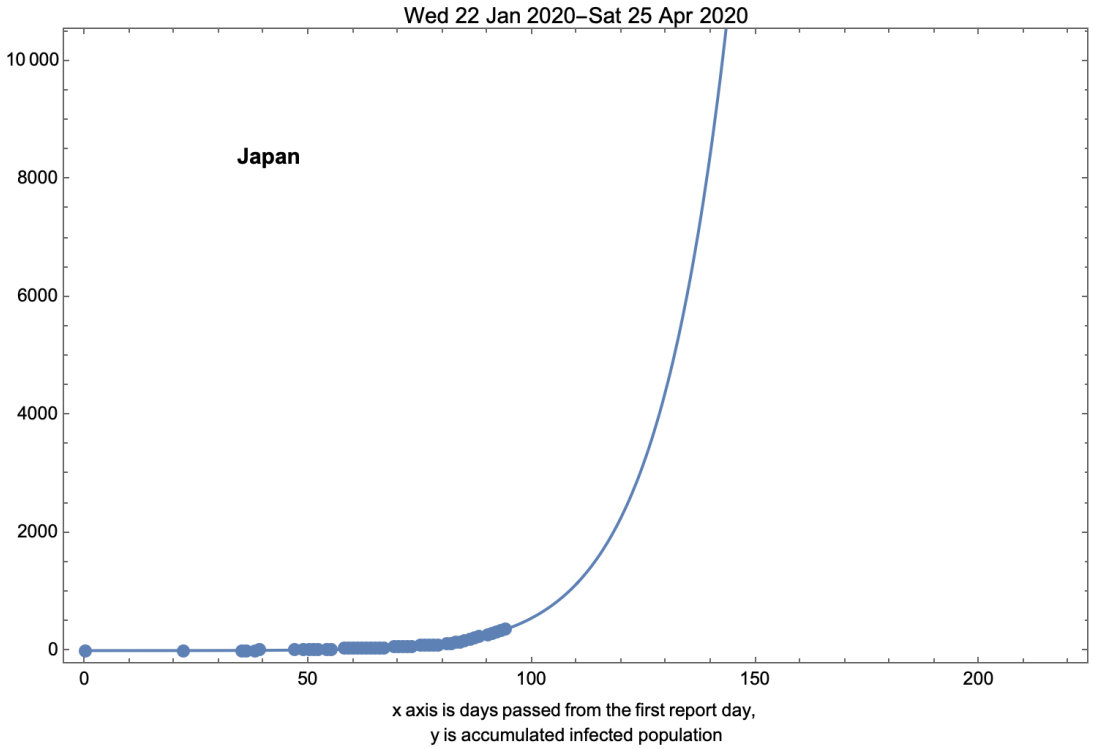
事象が収束するまで、すなわち、観測値とモデルの偏差が一定の誤差以下になるまで、以下

を続ける。

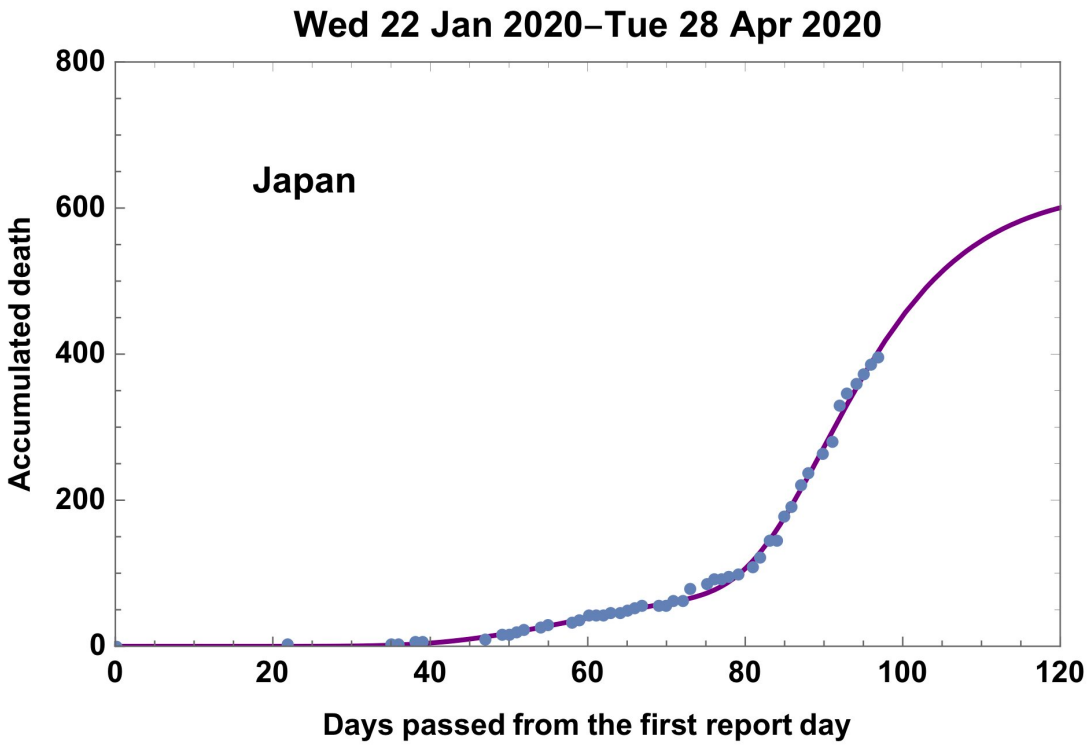
1. 観測値を与える
2. ヒューリスティックに初期値を与えた後、Fitting関数によって観測値に合致する曲線のパラメータを求める
3. Fittingされた（Fitting関数がオーバーフロー、アンダーフローが起こさない）モデルで将来予測する
4. 得られたGompert曲線あるいはLogistic曲線から得られた近似値を、元の観測値から差し引く
5. 誤差が一定範囲内から外れるなら、新規の事象が発生したと考え、新規事象の始まりを定義し、最初に戻る
6. 最初に戻る

計算途中の気づき～ロジスティック曲線では不可解な死亡者数が推定されること～

(2020/7/12) 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（最初のモデルから数十万人の死亡者が推定された）。



日本の事象の初期の状態。まだ高々500人程度の死者～単一の事象と考えたモデル～



日本の事象の初期の状態。まだ高々500人程度の死者～二つの事象の足し合わせと考えたモデル～

計算途中の気づき～Gompertz曲線の含意～

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**集団免疫の存在は否定される**）

計算途中の気づき～仮説推論の強化～

(2020/7/12) 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（最初のモデルから数十万人の死亡者が推定された）。

各国の比較を通じた仮説推論の強化

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）。複数の独立事象の和で事象が表現されるのは、死亡原因の主たるウィルス株は交代するためと考えることで、モデルにより報告死亡者数をよく表現できた。これは日本の例に明らかであるが、後に他の国の場合にも当てはまることが判明した（ドイツ

や英国、その他の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、個々の事象の反復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。

(2021/1/29)感染研による新型コロナウイルスSARS-CoV-2ゲノム情報による分子疫学調査（2021年1月14日現在）-> https://www.niid.go.jp/niid/ja/diseases/ka/corona-virus/2019-ncov-/2488-idsc/iasr-news/10152-493p01.html

(2021/1/30) 収束しなかったので、5回目計算の初期値をデータにより近くに再設定した。

(2021/8/17) δ株と推定される株による積算死者数への影響が無視できないスケールとなったため、第7番目のモデルを追加した。
(2021/9/30) 我が国の他に、対象国としてドイツと英国を選んで、継続的に事象を追跡することとした。
(2021/10/21) 全ての事象をGompertz曲線に替えても最終的な結果はあまり変わらないことを確認した。ただし、全事象の途中の誤差が生じる。
(2022/3/14) 我が国の死亡者が増加するに従い、初期には適切とも考えられたLogistic曲線からの乖離が、見られるようになった。そこで、Gompertz曲線に変更し、より良いフィッティング結果を得た。ドイツや英国の例を比較すると、コロナ事象はGompertz曲線が示す、本質を持つと考えるべきだと思われるようになった。
(2022/4/27) モデルと報告時系列に乖離が見られたので、事象を追加した。

データの収集

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
```

解析の始点

以下の行は最初の一回だけ実施する。国名を与えて、人口を求めるためである。

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

Out[]= 126 860 299

解析のルーティン

毎日あるいは適当な間隔でJohns Hopkins大学に集積されている死亡者数を取得する。日本時間では18時頃にアップデートされているようだ。Johns Hopkins大学に集積されたデータはJSON形式でgithubにアップされている。

3/09/2023にJohns Hopkis大学はデータ収集を停止し、これに引き続いてgithubのアップデートも停止された。（03/17/2023）

In[]:=
```
country = "Japan";
population = 126 860 299;
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[]= {Wed 22 Jan 2020, {Thu 9 Mar 2023, 72 997}}

In[]:=
```
ddata[[1 ;; 108]]
```

startPoint of daily analysis

In[]:=
```
country = "Japan";
population = 126 860 299;
precursol = {{Wed 22 Jan 2020, 0}, {Thu 23 Jan 2020, 0}, {Fri 24 Jan 2020, 0},
    {Sat 25 Jan 2020, 0}, {Sun 26 Jan 2020, 0}, {Mon 27 Jan 2020, 0},
    {Tue 28 Jan 2020, 0}, {Wed 29 Jan 2020, 0}, {Thu 30 Jan 2020, 0},
    {Fri 31 Jan 2020, 0}, {Sat 1 Feb 2020, 0}, {Sun 2 Feb 2020, 0}, {Mon 3 Feb 2020, 0},
    {Tue 4 Feb 2020, 0}, {Wed 5 Feb 2020, 0}, {Thu 6 Feb 2020, 0}, {Fri 7 Feb 2020, 0},
```

```
{{Sat 8 Feb 2020, 0}, {Sun 9 Feb 2020, 0}, {Mon 10 Feb 2020, 0}, {Tue 11 Feb 2020, 0},
 {Wed 12 Feb 2020, 0}, {Thu 13 Feb 2020, 1}, {Fri 14 Feb 2020, 1}, {Sat 15 Feb 2020, 1},
 {Sun 16 Feb 2020, 1}, {Mon 17 Feb 2020, 1}, {Tue 18 Feb 2020, 2},
 {Wed 19 Feb 2020, 2}, {Thu 20 Feb 2020, 2}, {Fri 21 Feb 2020, 2}, {Sat 22 Feb 2020, 2},
 {Sun 23 Feb 2020, 2}, {Mon 24 Feb 2020, 2}, {Tue 25 Feb 2020, 3},
 {Wed 26 Feb 2020, 3}, {Thu 27 Feb 2020, 4}, {Fri 28 Feb 2020, 5}, {Sat 29 Feb 2020, 6},
 {Sun 1 Mar 2020, 6}, {Mon 2 Mar 2020, 6}, {Tue 3 Mar 2020, 6}, {Wed 4 Mar 2020, 7},
 {Thu 5 Mar 2020, 7}, {Fri 6 Mar 2020, 7}, {Sat 7 Mar 2020, 7}, {Sun 8 Mar 2020, 8},
 {Mon 9 Mar 2020, 10}, {Tue 10 Mar 2020, 13}, {Wed 11 Mar 2020, 15},
 {Thu 12 Mar 2020, 19}, {Fri 13 Mar 2020, 21}, {Sat 14 Mar 2020, 22},
 {Sun 15 Mar 2020, 24}, {Mon 16 Mar 2020, 30}, {Tue 17 Mar 2020, 30},
 {Wed 18 Mar 2020, 32}, {Thu 19 Mar 2020, 33}, {Fri 20 Mar 2020, 34},
 {Sat 21 Mar 2020, 36}, {Sun 22 Mar 2020, 40}, {Mon 23 Mar 2020, 41},
 {Tue 24 Mar 2020, 42}, {Wed 25 Mar 2020, 45}, {Thu 26 Mar 2020, 47},
 {Fri 27 Mar 2020, 53}, {Sat 28 Mar 2020, 56}, {Sun 29 Mar 2020, 63},
 {Mon 30 Mar 2020, 65}, {Tue 31 Mar 2020, 67}, {Wed 1 Apr 2020, 72},
 {Thu 2 Apr 2020, 75}, {Fri 3 Apr 2020, 81}, {Sat 4 Apr 2020, 87}, {Sun 5 Apr 2020, 93},
 {Mon 6 Apr 2020, 96}, {Tue 7 Apr 2020, 98}, {Wed 8 Apr 2020, 105},
 {Thu 9 Apr 2020, 113}, {Fri 10 Apr 2020, 125}, {Sat 11 Apr 2020, 138},
 {Sun 12 Apr 2020, 147}, {Mon 13 Apr 2020, 156}, {Tue 14 Apr 2020, 170},
 {Wed 15 Apr 2020, 186}, {Thu 16 Apr 2020, 204}, {Fri 17 Apr 2020, 217},
 {Sat 18 Apr 2020, 236}, {Sun 19 Apr 2020, 255}, {Mon 20 Apr 2020, 275},
 {Tue 21 Apr 2020, 303}, {Wed 22 Apr 2020, 323}, {Thu 23 Apr 2020, 351},
 {Fri 24 Apr 2020, 360}, {Sat 25 Apr 2020, 379}, {Sun 26 Apr 2020, 395},
 {Mon 27 Apr 2020, 417}, {Tue 28 Apr 2020, 436}, {Wed 29 Apr 2020, 454},
 {Thu 30 Apr 2020, 481}, {Fri 1 May 2020, 510}, {Sat 2 May 2020, 536},
 {Sun 3 May 2020, 560}, {Mon 4 May 2020, 582}, {Tue 5 May 2020, 593},
 {Wed 6 May 2020, 605}, {Thu 7 May 2020, 617}, {Fri 8 May 2020, 634}};
```

日付を初回報告日からの経過日数へ変換すると共に、前日から報告値に変化のなかった日を削除する。

https://data.corona.go.jp/converted-json/covid19japan-ndeaths.json

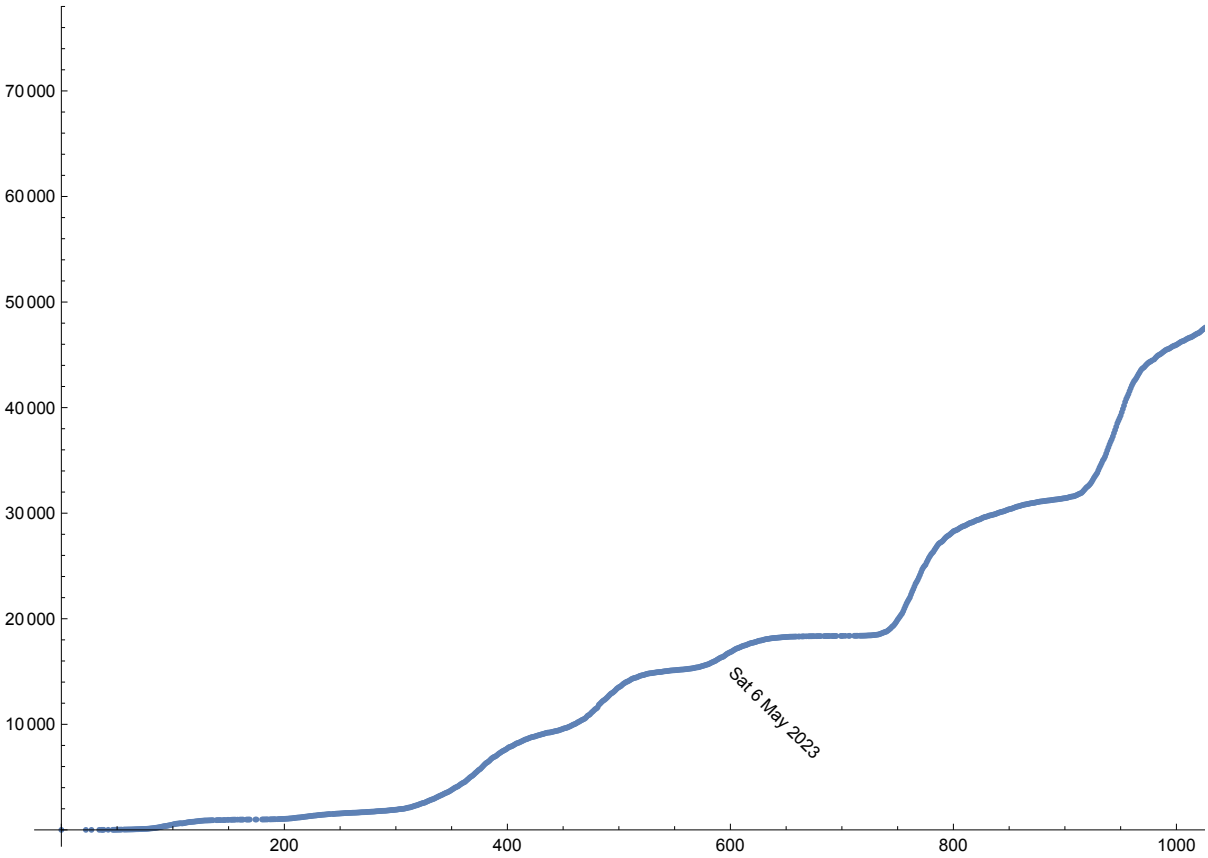
厚労省は2023年5月8日公表分以降、厚労省の個票積み上げを止め、各自治体が公表しているものを取りまとめる、という方針に変更した。これに従って、積算死亡者数の公表フォーマットも変更された。というわけで、
当方のモデリング作業も終了することにした。

In[ ]:=
```
urlCovid19x =
  "https://data.corona.go.jp/converted-json/covid19japan-ndeaths.json";
data = Import[urlCovid19x];
ddata = Map[{DateObject[#["date"]], #["ndeaths"]} &, Map[Association, data]];
ddata = Join[precursol, ddata];
{firstday = ddata[[1, 1]], Last[ddata]}
```

Out[ ]= { Wed 22 Jan 2020 , { Fri 7 Apr 2023 , 74 060 } }

In[ ]:=
```
separation = {0};
i4 = Map[{QuantityMagnitude[#[[1]] - firstday], #[[2]]} &, ddata];
raw = Map[First, Gather[i4, #1[[2]] == #2[[2]] &]];
rawModified[1] = raw;
Show[
 ListPlot[raw, PlotStyle → AbsolutePointSize[3], PlotRange → Automatic],
 Graphics[Text[DateString[DatePlus[firstday, 1200], "DateShort"],
   {600, 15 000}, {-1, 0}, {1, -1}]]]
```

Out[ ]=



markerの日付を読み取って文字列に変更し、markerとの位置を調整する関数である。

```
In[•]:= markerLegend[m_] :=
          Graphics[Text[DateString[DatePlus[firstday, First[m]], "DateShort"],
            {First[m] + 10, Last[m] - 500}, {-1, 0}, {1, -1}]];
```

日々のデルタ

Iteration計算

現在のところ、最終結果は14回の繰り返し計算により求めることとした。すなわち、死者数の結果は14個の事象の合成であると考える。

1回目-Gompertz曲線

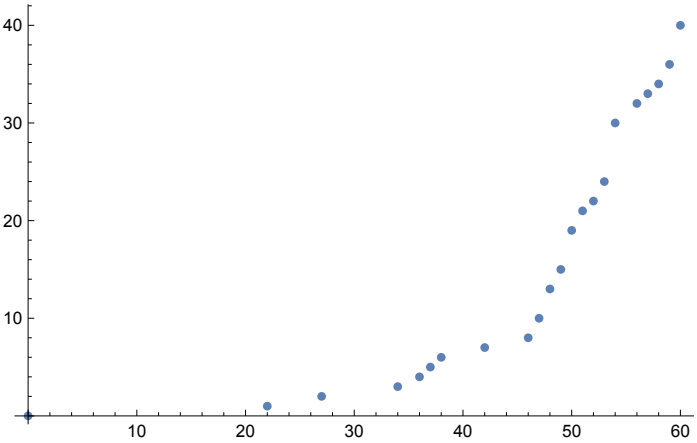
時系列の分割

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。

第一回は60日で切り分ける。切り分けたリストがtargetPeriodである。

```
In[•]:= separation = {0};
        rawModified[1] = raw;
        targetPeriod = Select[rawModified[1], #[[1]] ≤ 60 &];
        separation = Append[separation, 60];
        ListPlot[targetPeriod]
```

Out[•]=



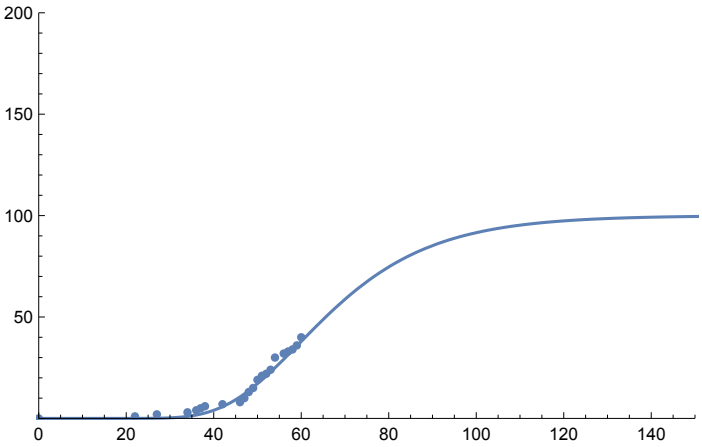
時系列へのマッチングの初期値設定

分割した時系列のうち、最初の時系列をモデル曲線にマッチングさせる。ここで、モデル曲線としてGompertz曲線を用いる。Gompertz曲線を報告時系列にマッチングさせるために、Gompertz曲線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。

ヒューリスティックに与える初期値と別に、Gompertz曲線の開始点、startを与える。必要な初期値は、n, c, bである。初期値を与え、プロットすることを繰り返すことで、凡その値が得られるであろう。

In[ ]:=
```
start = 40;
model[1] = n b^Exp[-c (t - start)];
initialguess = {n → 100, c → 0.06, b → 0.04};
Show[
 Plot[model[1] /. initialguess, {t, 0, 1000}, PlotRange → {{0, 150}, {0, 200}}],
 ListPlot[targetPeriod]
]
```

Out[ ]=



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。この後、初期値を与え直す必要はないであろう。

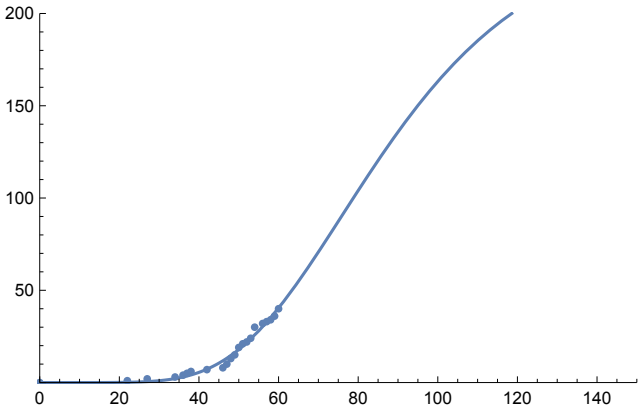
In[ ]:=
```
ans[1] = FindFit[targetPeriod, model[1],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[ ]=
```
{n → 245.461, c → 0.0371564, b → 0.0225056}
```

報告値と推測値を比較する。

In[ ]:=
```
Show[
 Plot[model[1] /. ans[1], {t, 0, 150}, PlotRange → {{0, 150}, {0, 200}}],
 ListPlot[targetPeriod]
]
```

Out[ ]=



報告時系列のモディファイ

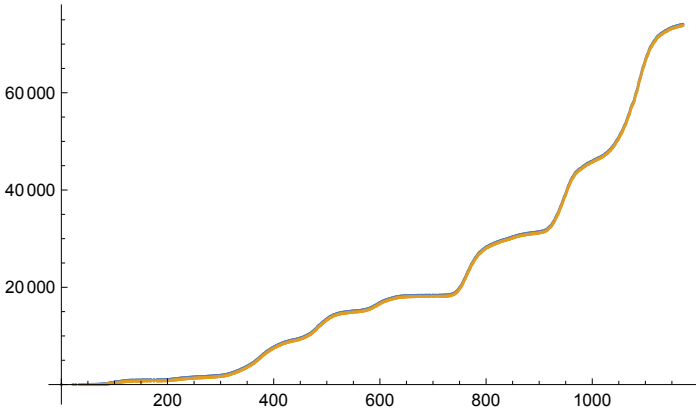
報告開始日から60日目までを最初のウィルス株による結果、それ以降を次のウィルス株による結果と考える。この切りわけ日の設定は、報告数の時系列が作る曲線の変曲点からヒューリスティックに決めてある。

時系列にfittingさせたGompertz曲線を用いて、報告時系列の全てから、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。ただし、最終的に推定される積算死者数モデルは、個々の対象とした事象範囲を超えて、この事象に引き続く全ての報告時系列に影響を及ぼしていることに注意する必要がある。

すなわち、事象の全ては加え合わせることのできる、独立な事象、つまり独立なウィルス株による結果とみなされる。

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

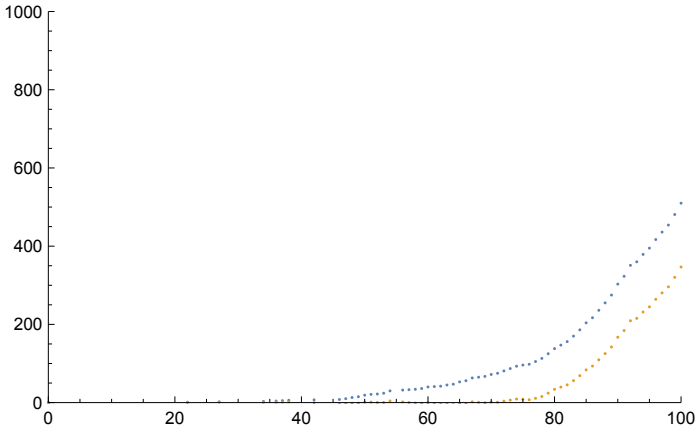
Out[ ]=



0点付近を拡大してみよう。

In[ ]:=
```
ListPlot[{rawModified[1], rawModified[2]}, PlotRange → {{0, 100}, {0, 1000}}]
```

Out[ ]=



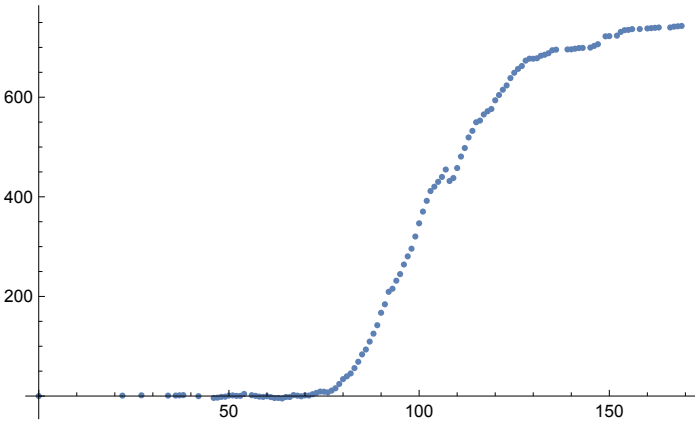
60日以前のデータの時系列が0にされた、rawModified[2]が生成された。新たに生成された時系列を、イタレーション計算のための時系列とみなすことで、同じパターンの解析を繰り返し、実行することができる。

2回目-Gompertz曲線

第２回は170日で切り分けて、着目するのがtargetPeriod である。

In[ ]:=
```
targetPeriod = Select[rawModified[2], #[[1]] ≤ 170 &];
separation = Append[separation, 170];
ListPlot[targetPeriod]
```

Out[ ]=

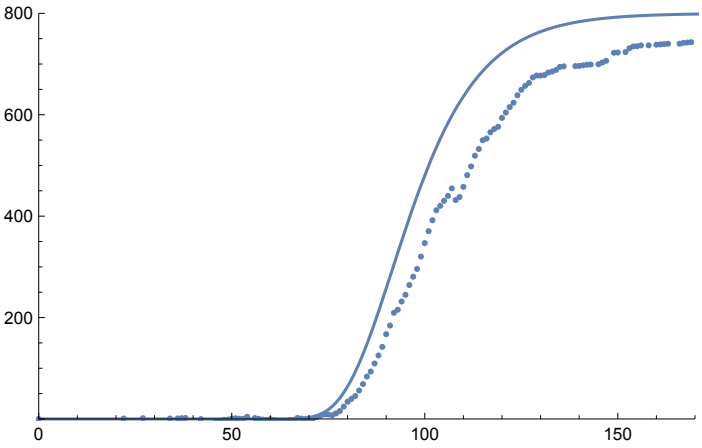


二番目の時系列をGompertz曲線にFittingさせるために、ヒューリスティックに初期値を与える。

In[ ]:=
```
start = 80;
model[2] = n b^Exp[-c ( t - start)];
initialguess = {n → 800, c → 0.08, b → 0.08};
Show[
 Plot[model[2] /. initialguess, {t, 0, 1000}, PlotRange → {{0, 170}, {0, 800}}],
 ListPlot[targetPeriod]
]
```

General : 0.08 $^{600.862}$ 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[ ]=



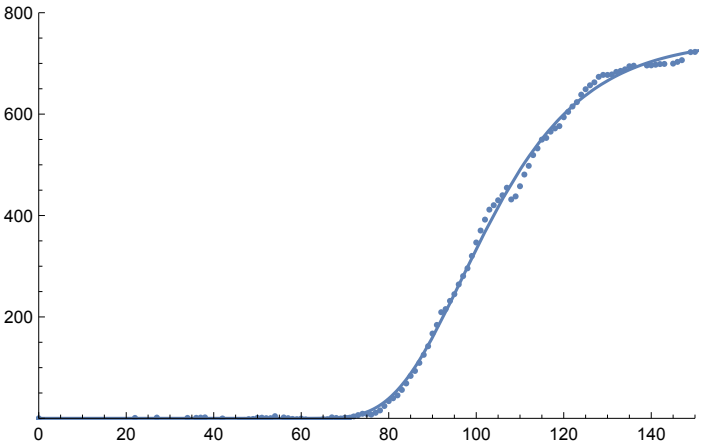
初期値を与えて、ターゲットとなる時系列をGompertz曲線にFittingさせる。

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

Out[ ]=
```
{n → 747.169, c → 0.0648842, b → 0.0521716}
```

In[ ]:=
```
upto = 200;
Show[
 Plot[model[2] /. ans[2], {t, 0, upto}, PlotRange → {{0, 150}, {0, 800}}],
 ListPlot[targetPeriod]
]
```

Out[ ]=



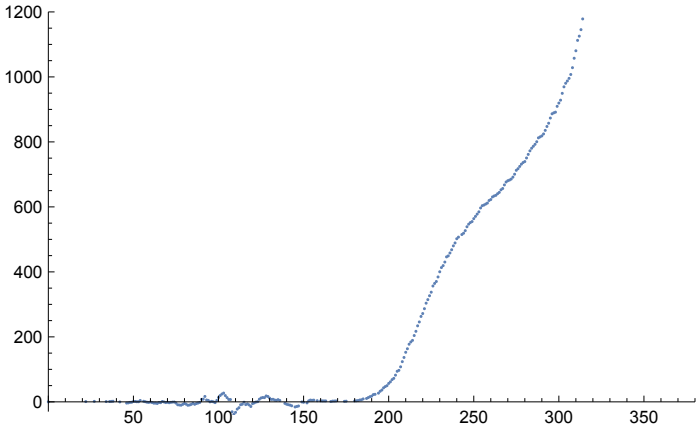
報告時系列のモディファイ

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

Out[ ]=



In[ ]:=

3回目-Gompertz曲線

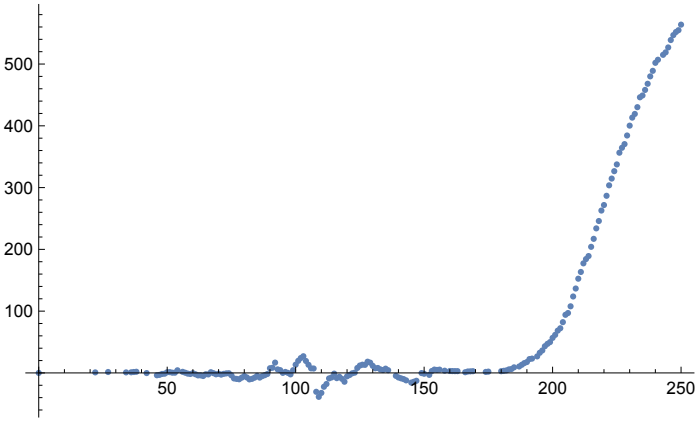
時系列の分割は終了

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250日で切り分けて、着目するのがtargetPeriod である。

In[ ]:=
```
targetPeriod = Select[rawModified[3], #[[1]] ≤ 250 &];
separation = Append[separation, 250];
ListPlot[targetPeriod]
```

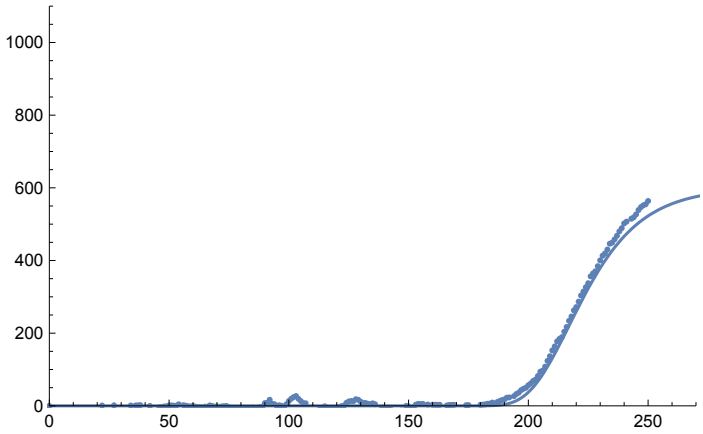
Out[ ]=



In[ ]:=
```
start = 180;
model[3] = n b^Exp[-c (t - start)];
initialguess = {n → 600, c → 0.06, b → 0.0001};
Show[
 Plot[model[3] /. initialguess, {t, 0, 1000}, PlotRange → {{0, 270}, {0, 1100}}],
 ListPlot[targetPeriod]
]
```

General : 0.0001 $^{48960.8}$ 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[ ]=



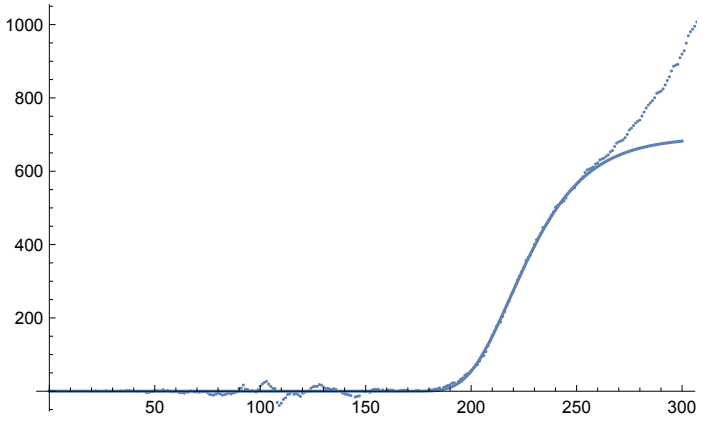
In[ ]:=
```
ans[3] = FindFit[targetPeriod, model[3],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[ ]=
```
{n → 693.324, c → 0.0504466, b → 0.000939972}
```

In[ ]:=
```
upto = 300;
Show[
 ListPlot[rawModified[3]],
 Plot[model[3] /. ans[3], {t, 0, upto}],
 PlotRange → {{0, upto}, {0, 1000}}]
```

General : 0.000939972 $^{8778.61}$ 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[ ]=

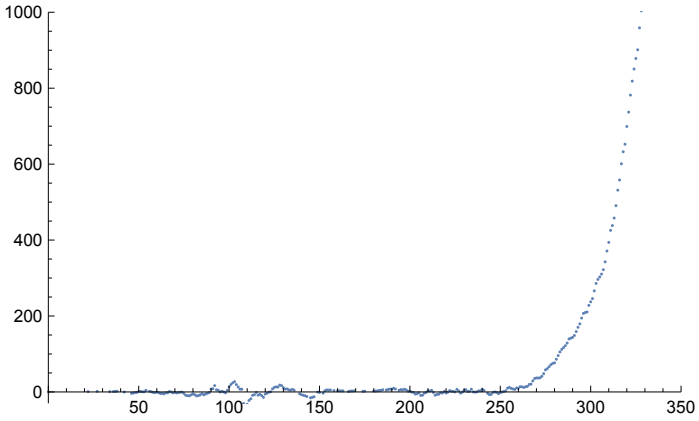


報告時系列のモディファイ

rawModified[4] が生成される。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[3]];
ListPlot[
 rawModified[4] = Transpose@{pdays, death - (model[3] /. ans[3] /. t → pdays)},
 PlotRange → {{0, 350}, {-30, 1000}}]
```

Out[ ]=



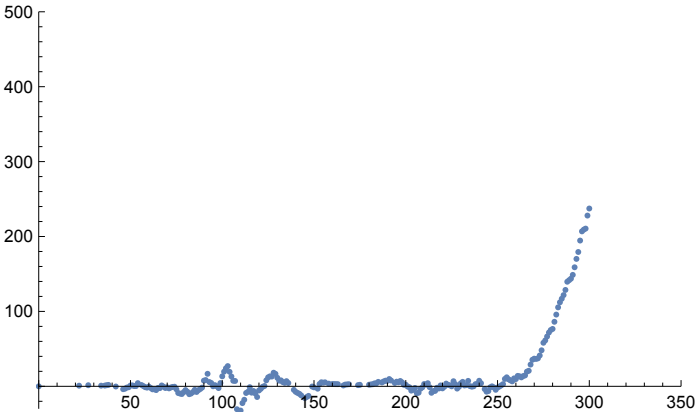
In[ ]:=

4回目-Gompertz曲線

第4回は290日で切り分けて、着目するのがtargetPeriod である。

In[ ]:=
```
targetPeriod = Select[rawModified[4], #[[1]] ≤ 300 &];
separation = Append[separation, 300];
ListPlot[targetPeriod, PlotRange → {{0, 350}, {-30, 500}}]
```

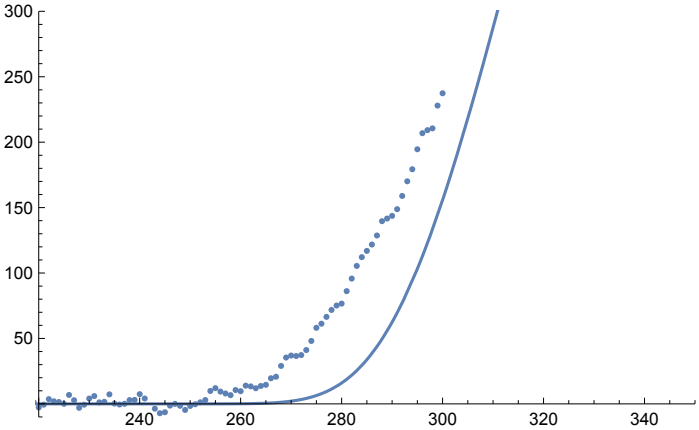
Out[ ]=



In[ ]:=
```
start = 260;
model[4] = n b^Exp[-c (t - start)];
initialguess = {n → 1000, c → 0.04, b → 0.0001};
Show[
 ListPlot[targetPeriod, PlotRange → {{220, 350}, {-10, 300}}],
 Plot[model[4] /. initialguess, {t, 0, 1000}]
]
```

··· General : 0.0001 $^{32832.8}$ 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[ ]=



与えた初期値で、ターゲット時系列をGompertz曲線にFittingさせる。

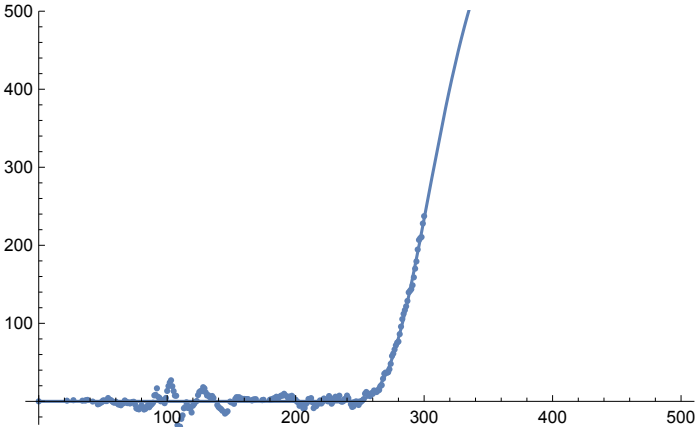
In[ ]:=
```
ans[4] = FindFit[targetPeriod, model[4],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[ ]= {n → 704.716, c → 0.0336725, b → 0.0143594}

In[ ]:=
```
upto = 500;
Show[
 Plot[model[4] /. ans[4], {t, 0, upto}, PlotRange → {0, 550}],
 ListPlot[targetPeriod],
 PlotRange → {{0, 500}, {-30, 500}}
]
```

··· General : 0.0143594 $^{6339.12}$ 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[ ]=



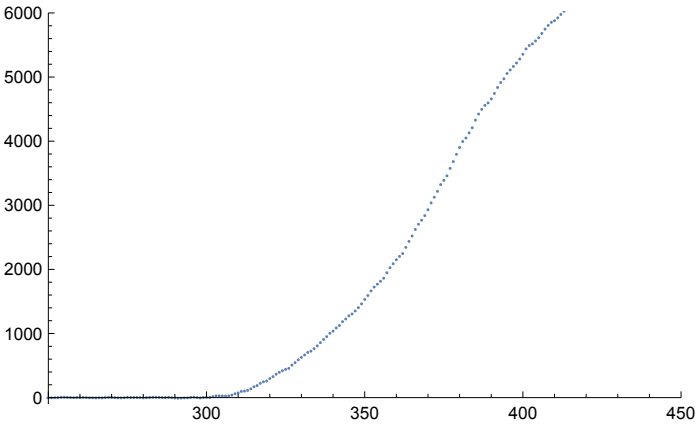
In[ ]:=

報告時系列のモディファイ

rawModified[5] が生成される。

In[]:=
```
{pdays, death} = Transpose[rawModified[4]];
ListPlot[
 rawModified[5] = Transpose@{pdays, death - (model[4] /. ans[4] /. t → pdays)},
 PlotRange → {{250, 450}, {-100, 6000}}]
```

Out[]=



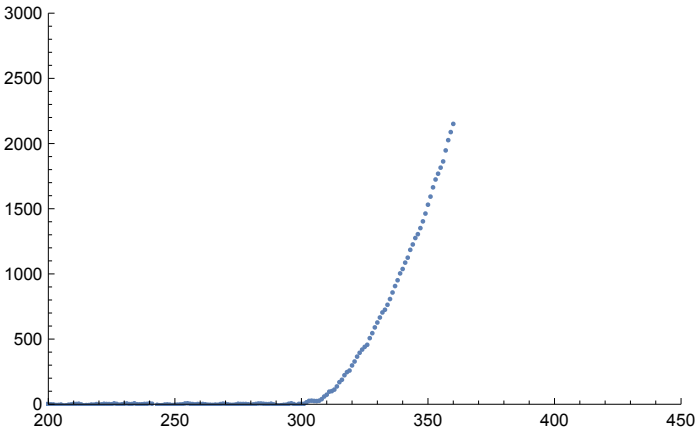
5回目-Gompertz曲線

第5回は360日で切り分けて、着目するのがtargetPeriod である。

In[]:=
```
start = 300;
targetPeriod = Select[rawModified[5], #[[1]] <= (360) &];
separation = Append[separation, 360];
```

In[]:=
```
ListPlot[targetPeriod, PlotRange → {{200, 450}, {0, 3000}}]
```

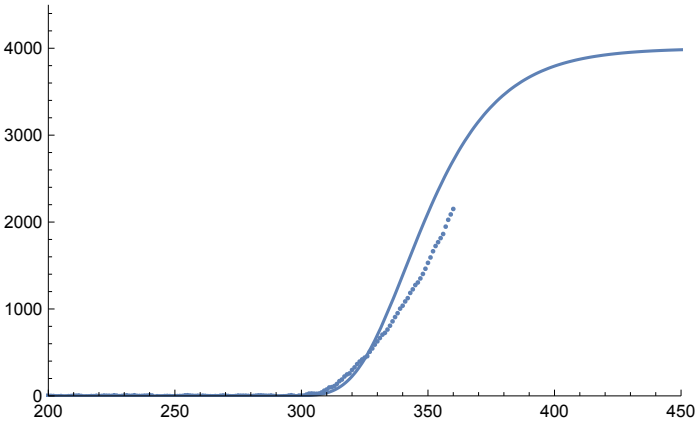
Out[]=



In[ ]:=
```
model[5] = n b^Exp[-c (t - start)];
initialguess = {n → 4000, c → 0.05, b → 0.0004};
Show[
 Plot[model[5] /. initialguess,
  {t, 0, 1000}, PlotRange → {{200, 450}, {0, 4500}}],
 ListPlot[targetPeriod]
]
```

General : 0.0004 $^{3.26568\times10^{6}}$ 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[ ]=



targetperiodにフィッティングする

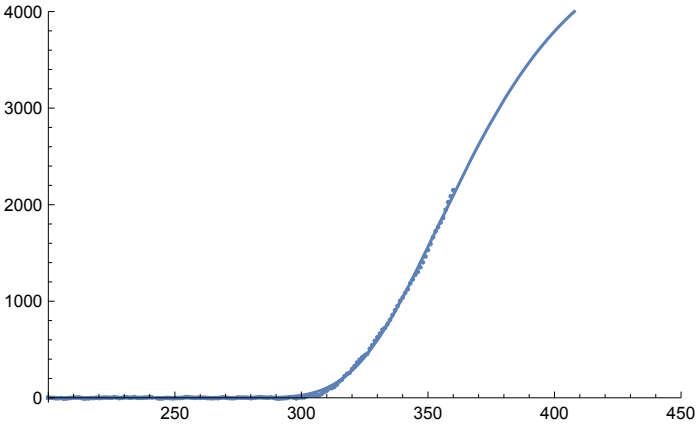
In[ ]:=
```
ans[5] = FindFit[targetPeriod, model[5],
   initialguess /. Rule → List, t, MaxIterations → 200]
```

Out[ ]= {n → 4859.96, c → 0.0305339, b → 0.00530514}

In[ ]:=
```
upto = 600;
Show[
 Plot[model[5] /. ans[5], {t, 0, upto}, PlotRange → {{200, 450}, {-50, 4000}}],
 ListPlot[targetPeriod]]
```

General : 0.00530514 $^{9507.}$ 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[ ]=

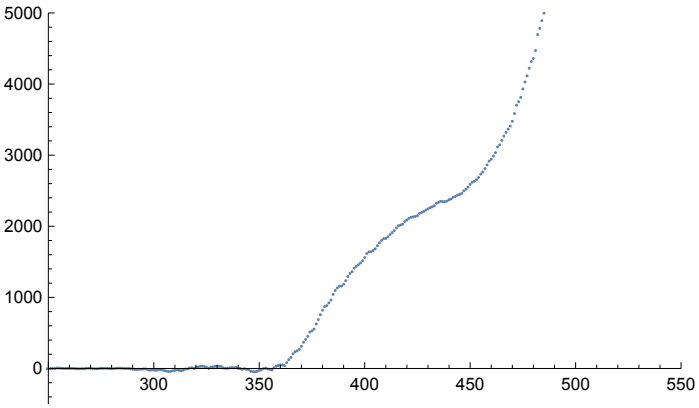


報告時系列のモディファイ

rawModified[6] を生成する。

In[•]:=
```
{pdays, death} = Transpose[rawModified[5]];
ListPlot[
 rawModified[6] = Transpose@{pdays, death - (model[5] /. ans[5] /. t → pdays)},
 PlotRange → {{250, 550}, {-500, 5000}}]
```

Out[•]=



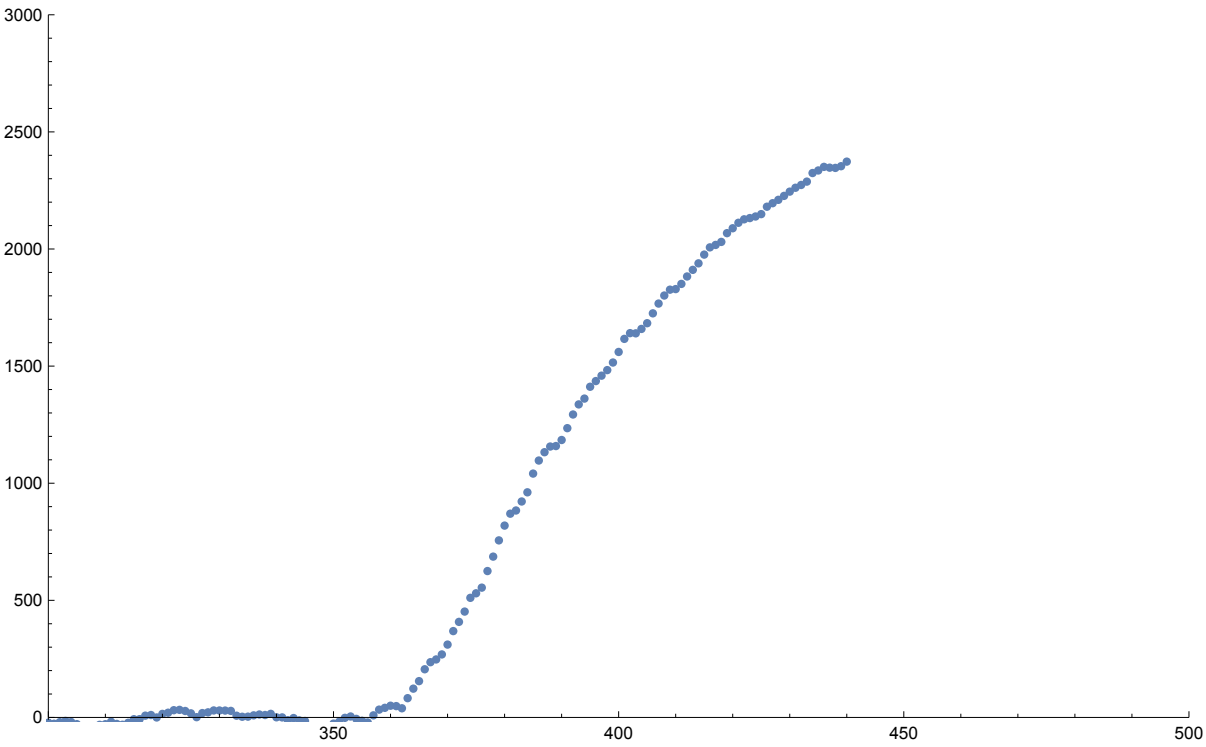
6回目-Gompertz曲線

第6回は560日で切り分けて、着目するのがtargetPeriod である。

In[•]:=
```
start = 400;
targetPeriod = Select[rawModified[6], #[[1]] <= (440) &];
separation = Append[separation, 440];
```

In[•]:=
```
ListPlot[targetPeriod, PlotRange → {{300, 500}, {-10, 3000}}]
```

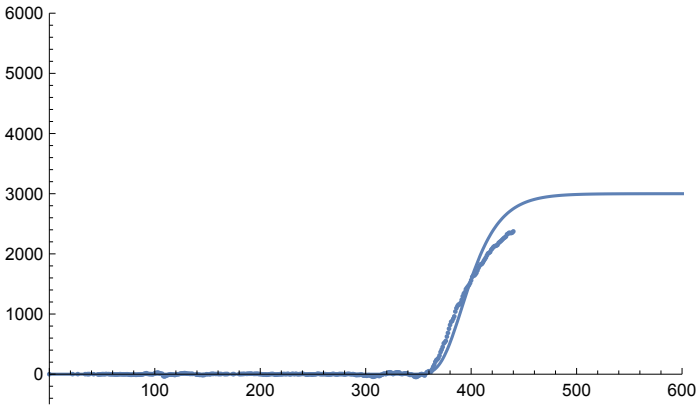
Out[•]=



In[ ]:=
```
start = 350;
model[6] = n b^Exp[-c ( t - start)];
initialguess = {n → 3000, c → 0.05, b → 0.0004};
Show[
 Plot[model[6] /. initialguess,
  {t, 0, 1000}, PlotRange → {{0, 600}, {-500, 6000}}],
 ListPlot[targetPeriod]
]
```

General : 0.0004 $^{3.97841 \times 10^{7}}$ 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[ ]=



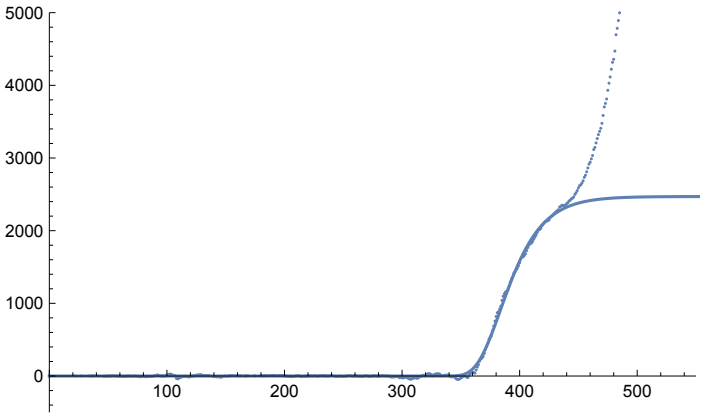
In[ ]:=
```
ans[6] = FindFit[targetPeriod, model[6],
   initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]=
```
{n → 2469.96, c → 0.0496281, b → 0.00487266}
```

In[ ]:=
```
Show[
 Plot[model[6] /. ans[6], {t, 0, 1000}, PlotRange → {{0, 550}, {-500, 5000}}],
 ListPlot[rawModified[6]]
]
```

General : 0.00487266 $^{3.49285 \times 10^{7}}$ 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[ ]=

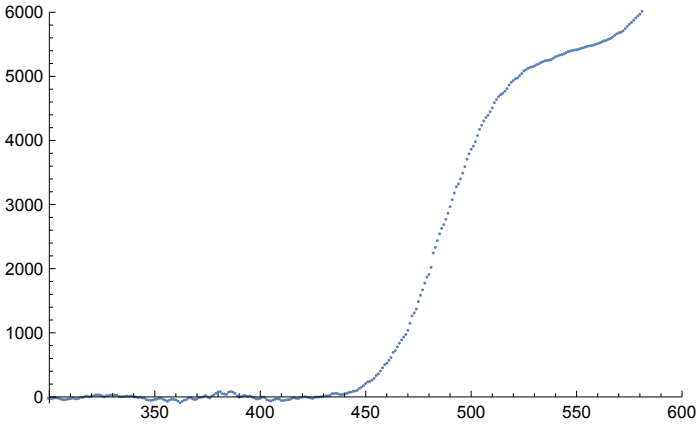


報告時系列のモディファイ

rawModified[7] を生成する。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[6]];
ListPlot[
 rawModified[7] = Transpose@{pdays, death - (model[6] /. ans[6] /. t → pdays)},
 PlotRange → {{300, 600}, {-100, 6000}}]
```

Out[ ]=



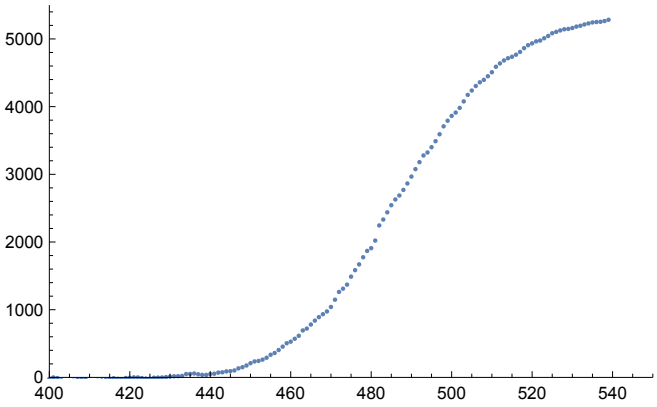
7回目-Gompertz曲線

第7回は520日で切り分けて、着目するのがtargetPeriod である。

In[ ]:=
```
targetPeriod = Select[rawModified[7], #[[1]] < (540) &];
separation = Append[separation, 540];
```

In[ ]:=
```
ListPlot[targetPeriod, PlotRange → {{400, 550}, {-10, 5500}}]
```

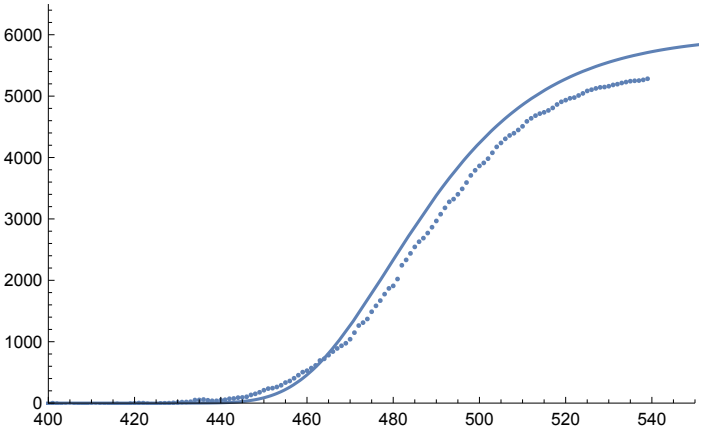
Out[ ]=



In[ ]:=
```
start = 430;
model[7] = n b^Exp[-c (t - start)];
initialguess = {n → 6000, c → 0.05, b → 0.00001};
Show[
 Plot[model[7] /. initialguess,
  {t, 0, 1200}, PlotRange → {{400, 550}, {-10, 6500}}],
 ListPlot[targetPeriod]
]
```

General : 0.00001 $^{2.1717\times10^{9}}$ 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[ ]=



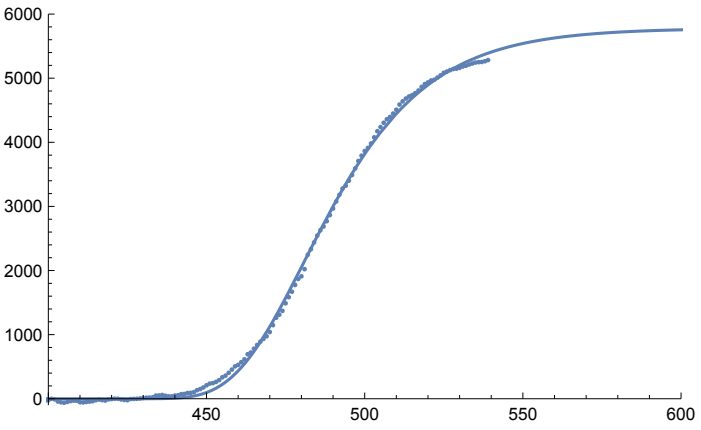
In[ ]:=
```
ans[7] = FindFit[targetPeriod, model[7],
   initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]=
```
{n → 5779.19, c → 0.0457629, b → 0.0000368938}
```

In[ ]:=
```
Show[
 Plot[model[7] /. ans[7], {t, 0, 1000}, PlotRange → {{400, 600}, {-100, 6000}}],
 ListPlot[targetPeriod]
]
```

General : 0.0000368938 $^{3.51284\times10^{8}}$ 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[ ]=

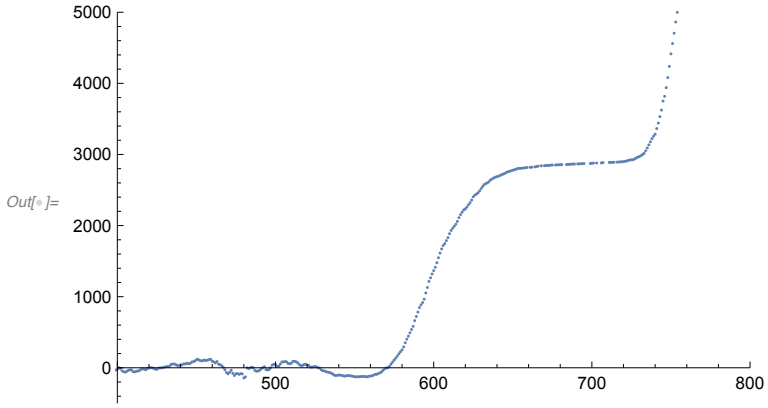


報告時系列のモディファイ

rawModified[8] を生成する。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[7]];
ListPlot[
 rawModified[8] = Transpose@{pdays, death - (model[7] /. ans[7] /. t → pdays)},
 PlotRange → {{400, 800}, {-500, 5000}}]
```

Out[ ]=



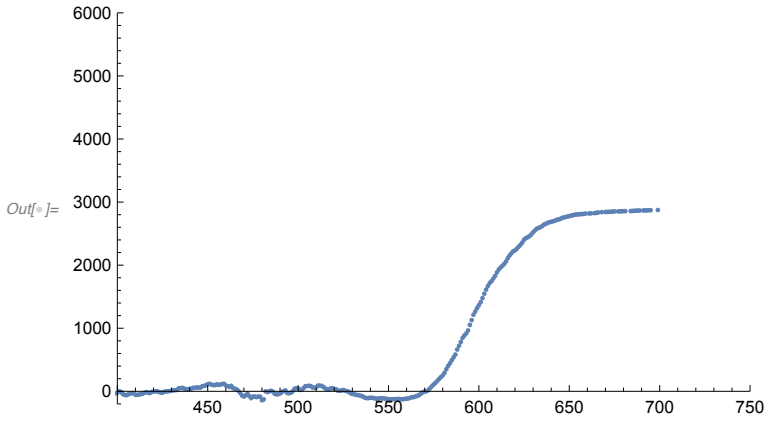
8回目-Gompertz曲線

第8回は700日で切り分けて、着目するのがtargetPeriod である。

In[ ]:=
```
targetPeriod = Select[rawModified[8], #[[1]] < (700) &];
separation = Append[separation, 700];
```

In[ ]:=
```
ListPlot[targetPeriod, PlotRange → {{400, 750}, {-200, 6000}}]
```

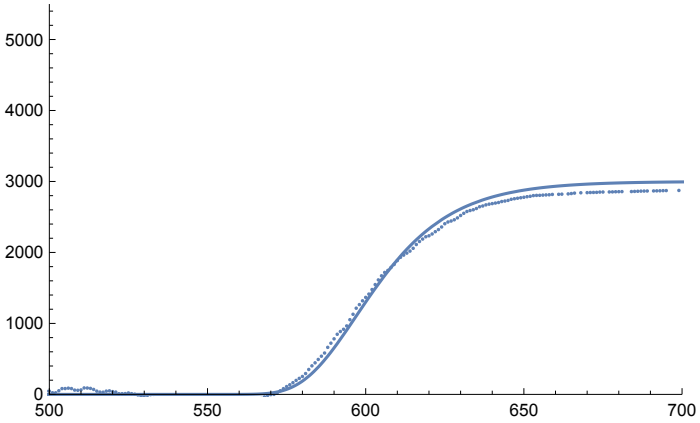
Out[ ]=



In[ ]:=
```
start = 560;
model[8] = n b^Exp[-c (t - start)];
initialguess = {n → 3000, c → 0.06, b → 0.0001};
Show[
 Plot[model[8] /. initialguess,
  {t, 0, 1200}, PlotRange → {{500, 700}, {-10, 5500}}],
 ListPlot[targetPeriod]
]
```

General : 0.0001 $^{3.90531\times10^{14}}$ 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[ ]=



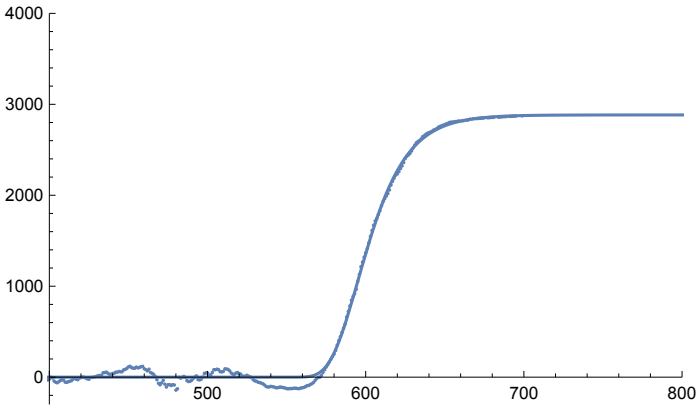
In[ ]:=
```
ans[8] = FindFit[targetPeriod, model[8],
   initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]=
```
{n → 2883.17, c → 0.0575884, b → 0.000529802}
```

In[ ]:=
```
Show[
 Plot[model[8] /. ans[8], {t, 0, 1000}, PlotRange → {{400, 800}, {-300, 4000}}],
 ListPlot[targetPeriod]
]
```

General : 0.000529802 $^{1.01221\times10^{14}}$ 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[ ]=

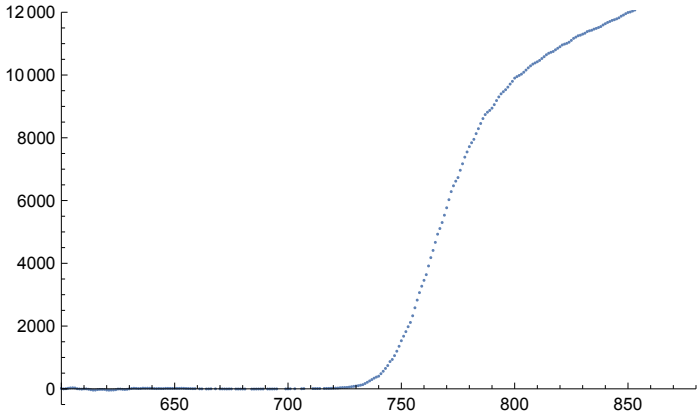


報告時系列のモディファイ

rawModified[9] を生成する。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[8]];
ListPlot[
 rawModified[9] = Transpose@{pdays, death - (model[8] /. ans[8] /. t → pdays)},
 PlotRange → {{600, 880}, {-500, 12 000}}]
```

Out[ ]=

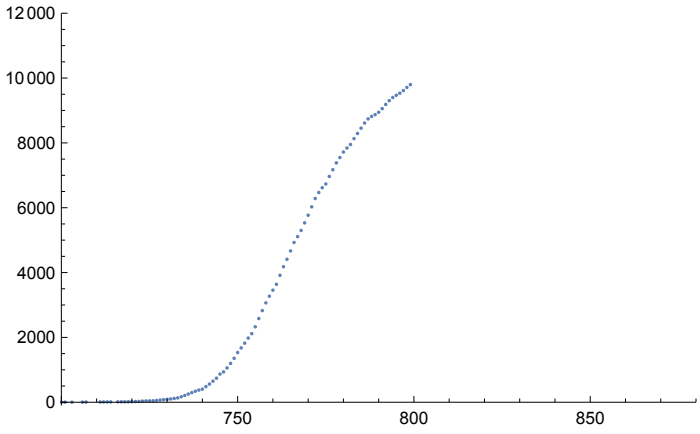


9回目-Gompertz曲線

第8回は700日で切り分けて、着目するのがtargetPeriod である。
9回目のtargetPeriodをGompertz曲線では、fittingできないが、Logistic曲線はfittingできる。t-argetPeriotdを構成する変異株に対して、集団免疫が発現したと言えるだろうか。対象とする国民にワクチンが行き渡り、死亡に至らなくなっていると見なすなら、これを免疫の発現としてもよいであろうか、とも考えたが、事象が進展するに従って、Gompertz曲線によるフィッティングが可能となった。

In[ ]:=
```
targetPeriod = Select[rawModified[9], #[[1]] < (800) &];
separation = Append[separation, 800];
ListPlot[targetPeriod, PlotRange → {{700, 880}, {-100, 12 000}}]
```

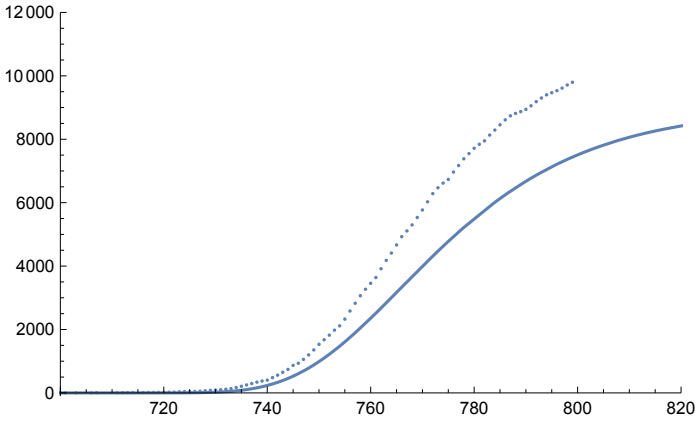
Out[ ]=



In[ ]:=
```
start = 720;
model[9] = n b^Exp[-c (t - start)];
initialguess = {n → 9000, c → 0.050, b → 0.00005};

Show[
 Plot[model[9] /. initialguess,
  {t, 0, 1200}, PlotRange → {{700, 820}, {-100, 12 000}}],
 ListPlot[targetPeriod]
]
```

General : 0.00005 $^{4.30595 \times 10^{15}}$ 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[ ]=



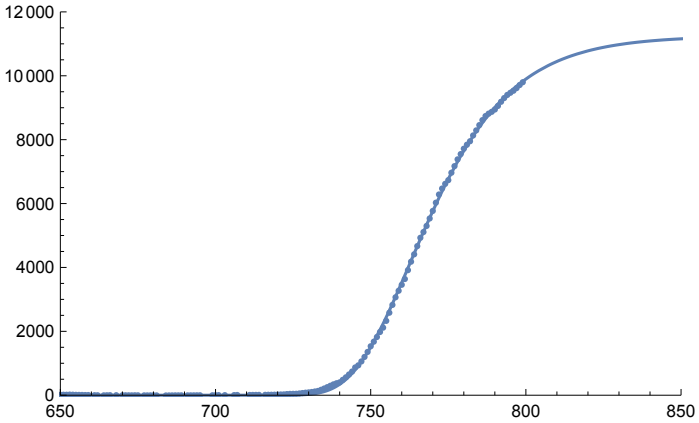
In[ ]:=
```
ans[9] = FindFit[targetPeriod, model[9],
   initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]=
```
{n → 11 247.8, c → 0.0551093, b → 0.0000271844}
```

In[ ]:=
```
Show[
 Plot[model[9] /. ans[9], {t, 0, 1200}, PlotRange → {{650, 850}, {-10, 12 000}}],
 ListPlot[targetPeriod, PlotStyle → {PointSize[0.01]}]
]
```

General : 0.0000271844 $^{1.70471 \times 10^{17}}$ 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[ ]=

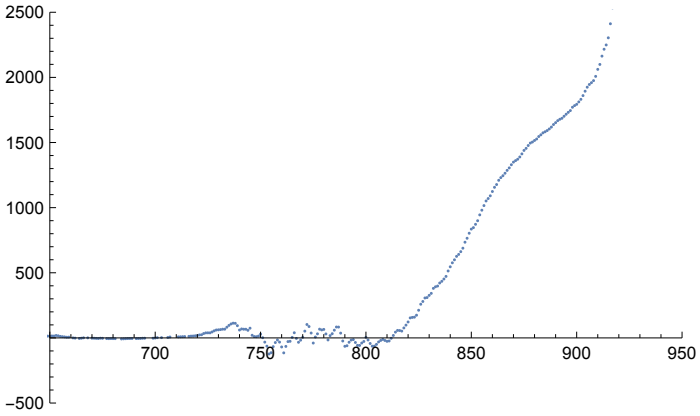


報告時系列のモディファイ

rawModified[10] を生成する。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[9]];
ListPlot[
 rawModified[10] = Transpose@{pdays, death - (model[9] /. ans[9] /. t → pdays)},
 PlotRange → {{650, 950}, {-500, 2500}}]
```

Out[ ]=

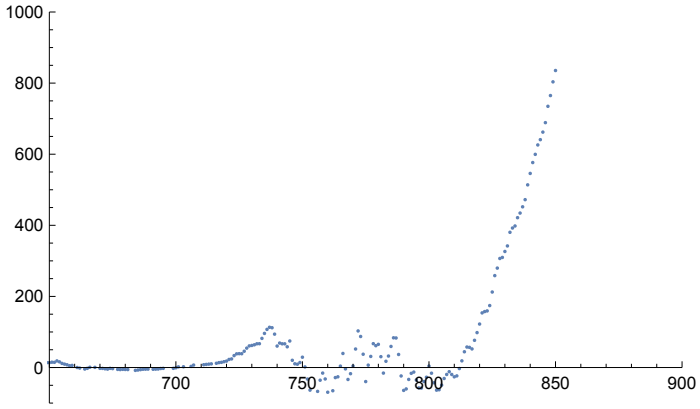


10回目-Gompertz曲線

第10回は850日で切り分けて、着目するのがtargetPeriod である。
9回目のtargetPeriodをGompertz曲線では、fittingできないが、Logistic曲線はfittingできた。t-argetPeriotdを構成する変異株に対して、集団免疫が発現したと言えるだろうか。対象とする国民にワクチンが行き渡り、死亡に至らなくなっていると見なすなら、これを免疫の発現としてもよいであろうか、とも考えたが、事象が進展するに従って、Gompertz曲線によるフィッティングが可能となった。

In[ ]:=
```
targetPeriod = Select[rawModified[10], #[[1]] <= (850) &];
separation = Append[separation, 810];
ListPlot[targetPeriod, PlotRange → {{650, 900}, {-100, 1000}}]
```

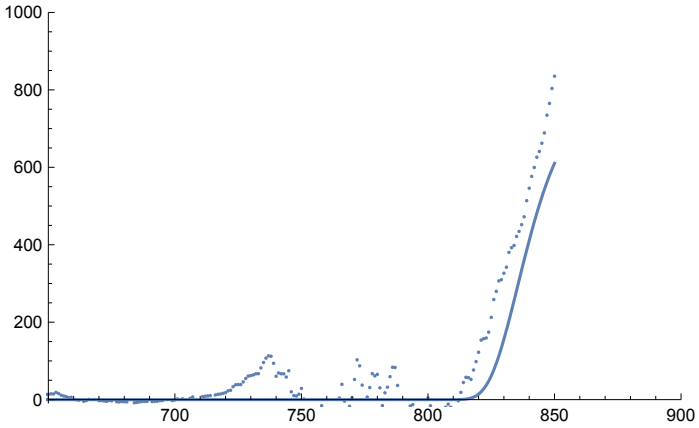
Out[ ]=



In[ ]:=
```
start = 810;
model[10] = n b^Exp[-c ( t - start)];
initialguess = {n → 800, c → 0.090, b → 0.00005};

Show[
 Plot[model[10] /. initialguess,
  {t, 0, 850}, PlotRange → {{650, 900}, {-10, 1000}}],
 ListPlot[targetPeriod]
]
```

General : 0.00005 $^{4.56446 \times 10^{31}}$ 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[ ]=



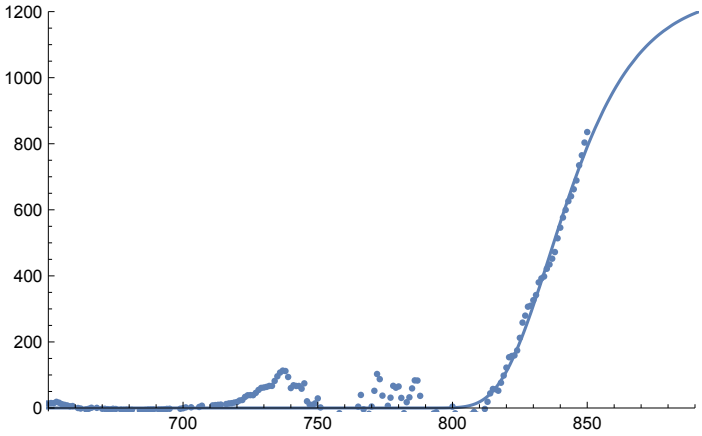
In[ ]:=
```
ans[10] = FindFit[targetPeriod, model[10],
   initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]=
```
{n → 1257.75, c → 0.0551704, b → 0.0147019}
```

In[ ]:=
```
Show[
 Plot[model[10] /. ans[10], {t, 0, 1200}, PlotRange → {{650, 890}, {-10, 1200}}],
 ListPlot[targetPeriod, PlotStyle → {PointSize[0.01]}]
]
```

General : 0.0147019 $^{2.55364 \times 10^{19}}$ 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[ ]=

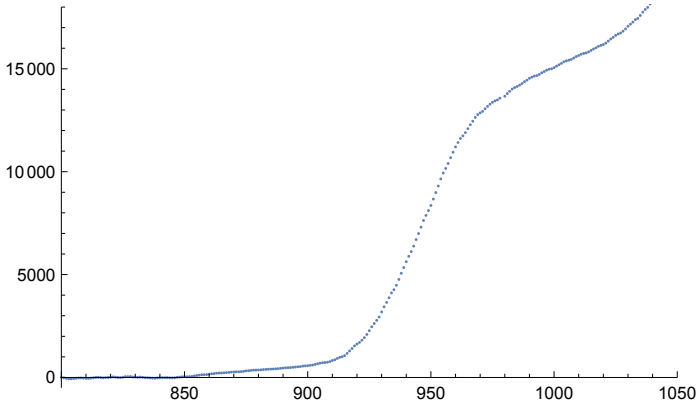


報告時系列のモディファイ

rawModified[11] を生成する。

In[]:=
```
{pdays, death} = Transpose[rawModified[10]];
ListPlot[
 rawModified[11] = Transpose@{pdays, death - (model[10] /. ans[10] /. t → pdays)},
 PlotRange → {{800, 1050}, {-500, 18 000}}]
```

Out[]=

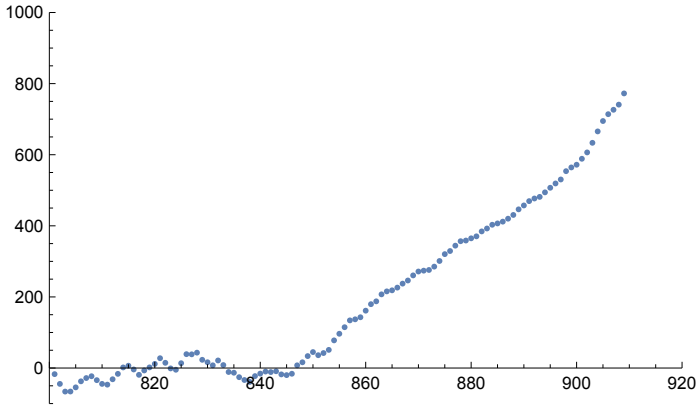


11回目-Gompertz曲線

評価されておらず、従前のモデルでは表現できていない、rawModified[11]時系列について評価を加える。

In[]:=
```
targetPeriod = Select[rawModified[11], (800) < #[[1]] < (910) &];
ListPlot[targetPeriod, PlotRange → {{800, 920}, {-100, 1000}}]
```

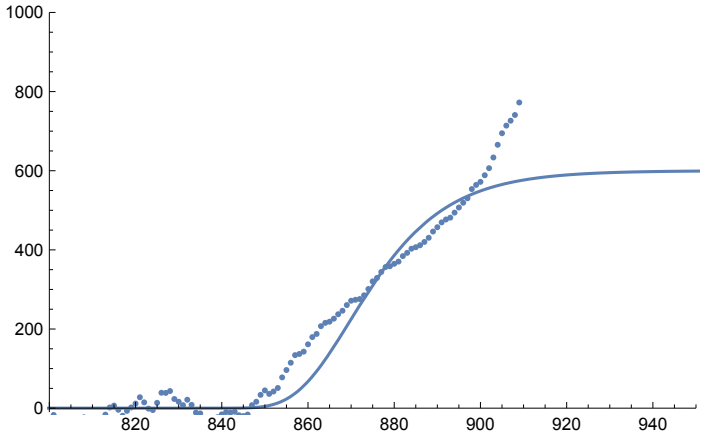
Out[]=



In[ ]:=
```
start = 840;
model[11] = n b^Exp[-c (t - start)];
initialguess = {n → 600, c → 0.080, b → 0.00002};
separation = Append[separation, 840];
Show[
 Plot[model[11] /. initialguess,
  {t, 0, 1000}, PlotRange → {{800, 950}, {-10, 1000}}],
 ListPlot[targetPeriod]
]
```

··· General : 0.00002 $^{1.52714 \times 10^{29}}$ 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[ ]=



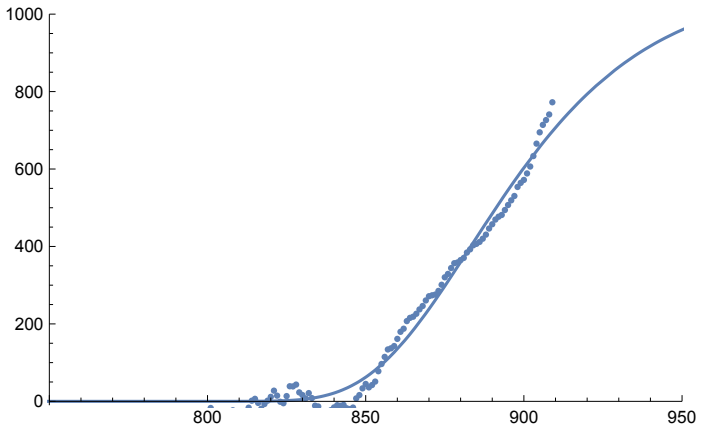
In[ ]:=
```
ans[11] = FindFit[targetPeriod, model[11],
   initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]=
```
{n → 1080.94, c → 0.0317501, b → 0.0198488}
```

In[ ]:=
```
Show[
 Plot[model[11] /. ans[11], {t, 0, 1200}, PlotRange → {{750, 950}, {-10, 1000}}],
 ListPlot[targetPeriod, PlotStyle → {PointSize[0.01]}]
]
```

··· General : 0.0198488 $^{3.82237 \times 10^{11}}$ 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[ ]=

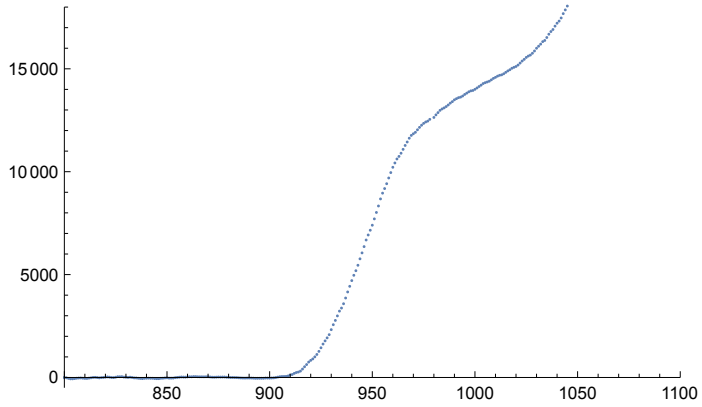


報告時系列のモディファイ

rawModified[12] を生成する。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[11]];
ListPlot[
 rawModified[12] = Transpose@{pdays, death - (model[11] /. ans[11] /. t → pdays)},
 PlotRange → {{800, 1100}, {-500, 18 000}}]
```

Out[ ]=

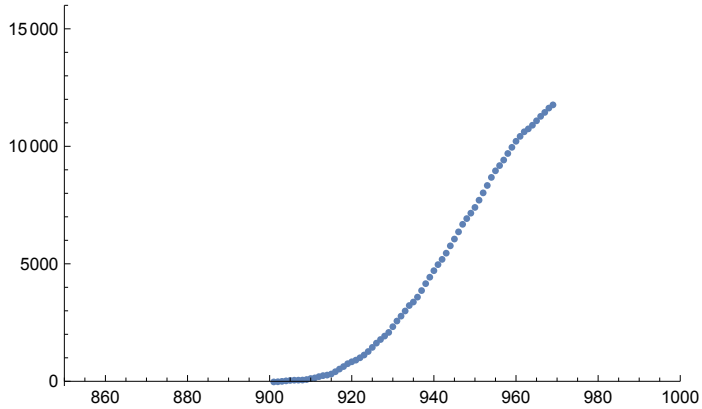


12回目-Logistic曲線

評価されておらず、従前のモデルでは表現できていない、rawModified[12]時系列について評価を加える。

In[ ]:=
```
targetPeriod = Select[rawModified[12], (900) < #[[1]] < (970) &];
ListPlot[targetPeriod, PlotRange → {{850, 1000}, {-200, 16 000}}]
```

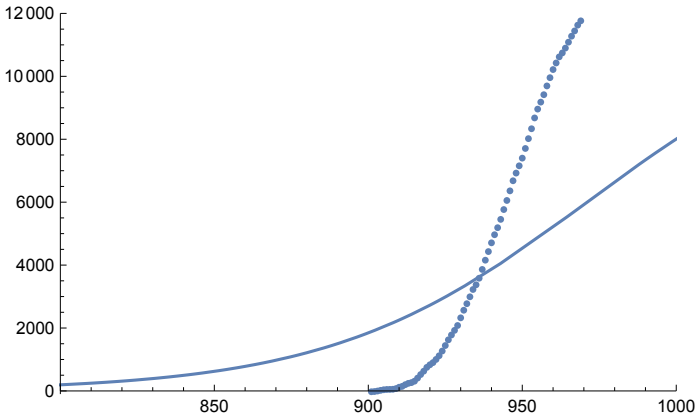
Out[ ]=



ここで、モデル[12]としてGompertz曲線を採用すると、モデルは950日辺りの報告値の減少を表現することができず、モデル[12]が過剰に報告値データ数を削減することとなる。その結果、モデル[13]に十分なプラスの報告数を提示することができなくなってしまう（従来にない現象である）。

In[•]:=
```
start = 900;
model[12] = n / (1 + c Exp[-a n (t - start)]);
initialguess = {n → 12 000, c → 5.5, a → 0.000002};
separation = Append[separation, 900];
Show[
 Plot[model[12] /. initialguess,
  {t, 0, 1100}, PlotRange → {{800, 1000}, {-100, 12 000}}],
 ListPlot[targetPeriod]
]
```

Out[•]=

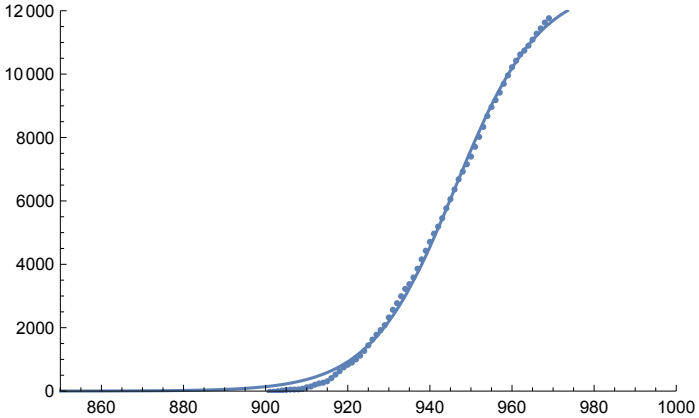


In[•]:=
```
ans[12] = FindFit[targetPeriod, model[12],
   initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[•]= $\{n \to 12\,832.6,\ c \to 90.3437,\ a \to 7.60382 \times 10^{-6}\}$

In[•]:=
```
Show[
 Plot[model[12] /. ans[12], {t, 0, 1100},
  PlotRange → {{850, 1000}, {-10, 12 000}}],
 ListPlot[targetPeriod, PlotStyle → {PointSize[0.01]}]
]
```

Out[•]=

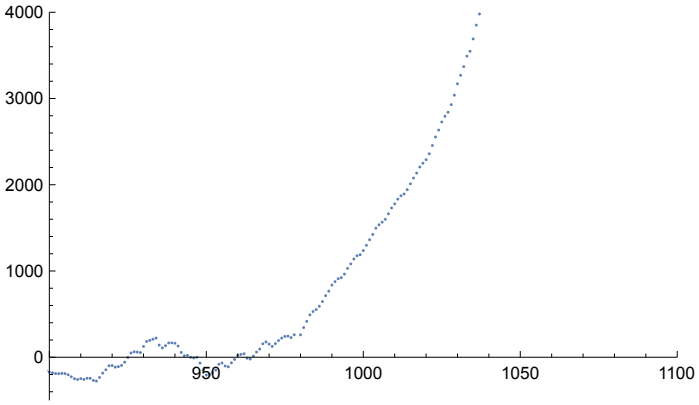


報告時系列のモディファイ

rawModified[13] を生成する。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[12]];
ListPlot[
 rawModified[13] = Transpose@{pdays, death - (model[12] /. ans[12] /. t → pdays)},
 PlotRange → {{900, 1100}, {-500, 4000}}]
```

Out[ ]=

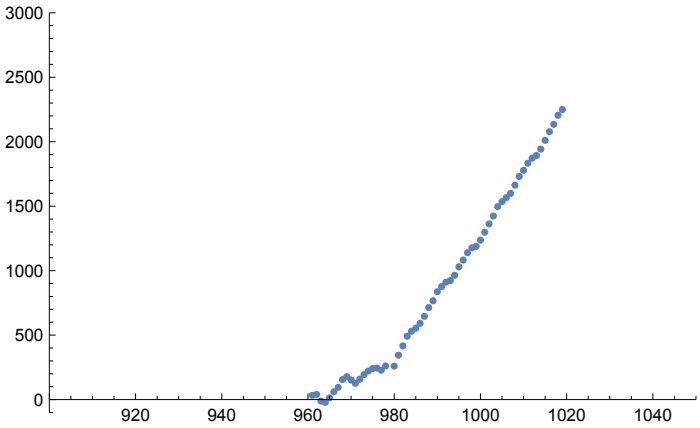


13回目-Gompertz曲線

評価されておらず、従前のモデルでは表現できていない、rawModified[13]時系列について評価を加える。

In[ ]:=
```
targetPeriod = Select[rawModified[13], (960 < #[[1]] < 1020) &];
ListPlot[targetPeriod, PlotRange → {{900, 1050}, {-100, 3000}}]
```

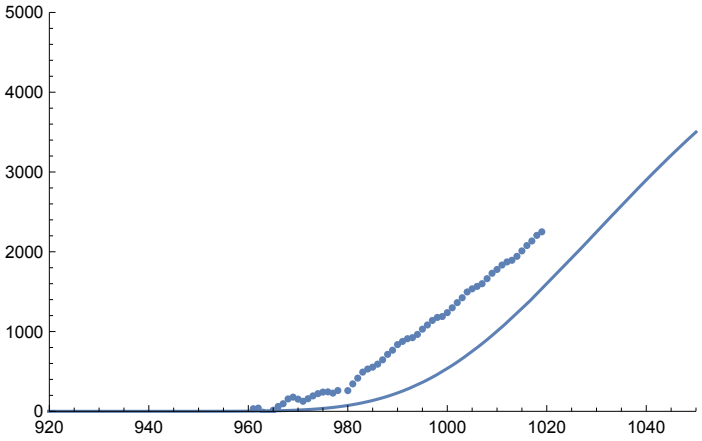
Out[ ]=



In[ ]:=
```
start = 950;
model[13] = n b^Exp[-c ( t - start)];
initialguess = {n → 6000, c → 0.03, b → 0.00002};
separation = Append[separation, 950];
Show[
 Plot[model[13] /. initialguess,
  {t, 0, 1050}, PlotRange → {{920, 1050}, {-10, 5000}}],
 ListPlot[targetPeriod]
]
```

General : 0.00002 $^{2.38294\times10^{12}}$ 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[ ]=



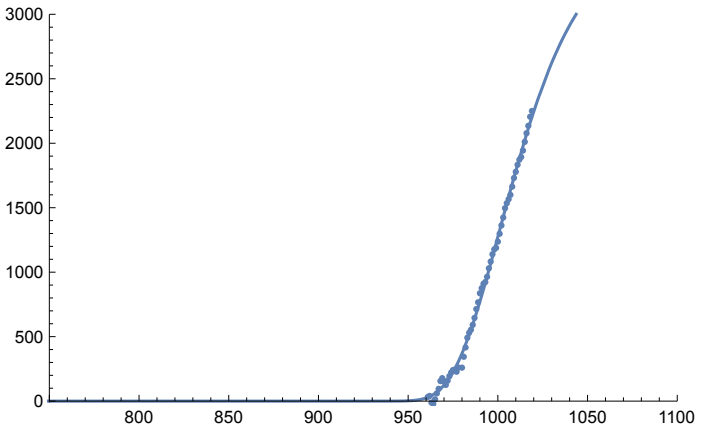
In[ ]:=
```
ans[13] = FindFit[targetPeriod, model[13],
   initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]=
```
{n → 3633.5, c → 0.0389732, b → 0.000632954}
```

In[ ]:=
```
Show[
 Plot[model[13] /. ans[13], {t, 0, 1200}, PlotRange → {{750, 1100}, {-10, 3000}}],
 ListPlot[targetPeriod, PlotStyle → {PointSize[0.01]}]
]
```

General : 0.000632954 $^{1.19992\times10^{16}}$ 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[ ]=

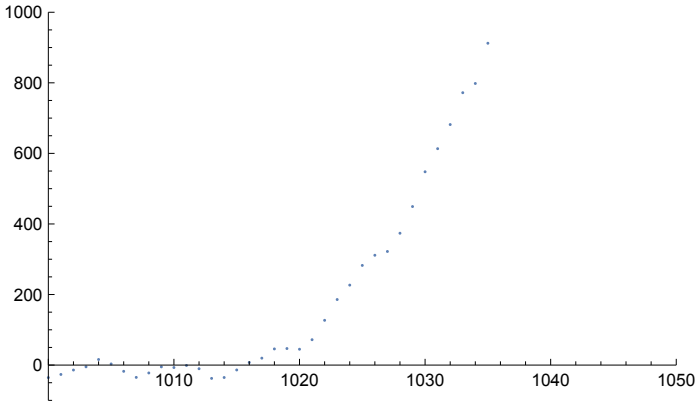


報告時系列のモディファイ

rawModified[14] を生成する。

In[]:=
```
{pdays, death} = Transpose[rawModified[13]];
ListPlot[
 rawModified[14] = Transpose@{pdays, death - (model[13] /. ans[13] /. t → pdays)},
 PlotRange → {{1000, 1050}, {-100, 1000}}]
```

Out[]=

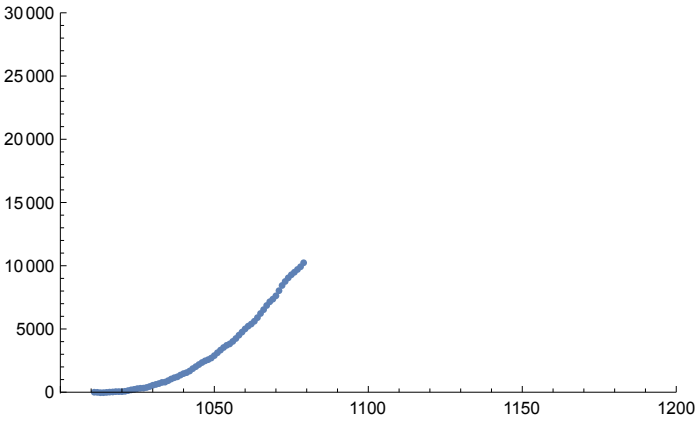


14回目-Logistic曲線

評価されておらず、従前のモデルでは表現できていない、rawModified[14]時系列について評価を加える。

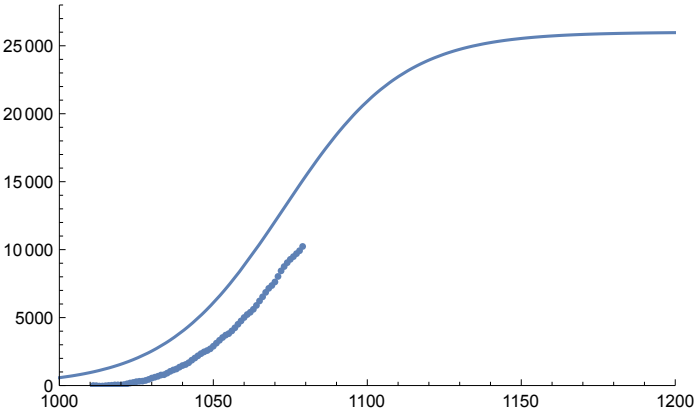
In[]:=
```
targetPeriod = Select[rawModified[14], (1010 < #[[1]] < 1080) &];
ListPlot[targetPeriod, PlotRange → {{1000, 1200}, {-100, 30 000}}]
```

Out[]=



In[ ]:=
```
start = 1040;
model[14] = n / (1 + c Exp[-a n (t - start)]);
initialguess = {n → 26 000, c → 5.5, a → 0.000002};
Show[
 Plot[model[14] /. initialguess,
  {t, 0, 1200}, PlotRange → {{1000, 1200}, {-10, 28 000}}],
 ListPlot[targetPeriod]
]
```

Out[ ]=

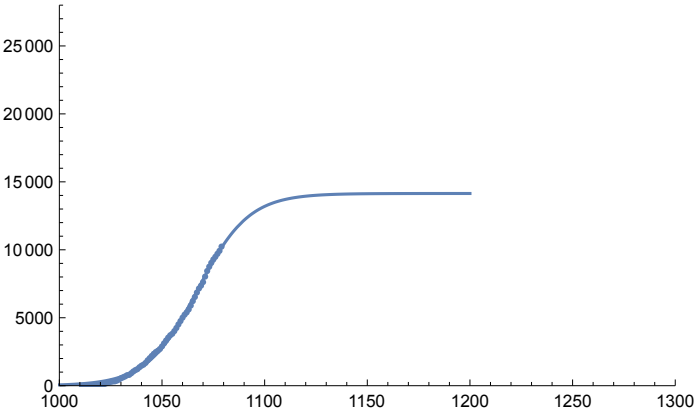


In[ ]:=
```
ans[14] = FindFit[targetPeriod, model[14],
  initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]= $\{n \to 14\,143.2,\ c \to 9.19489,\ a \to 5.71749 \times 10^{-6}\}$

In[ ]:=
```
Show[
 Plot[model[14] /. ans[14], {t, 0, 1200},
  PlotRange → {{1000, 1300}, {-10, 28 000}}],
 ListPlot[targetPeriod, PlotStyle → {PointSize[0.01]}]
]
```

Out[ ]=

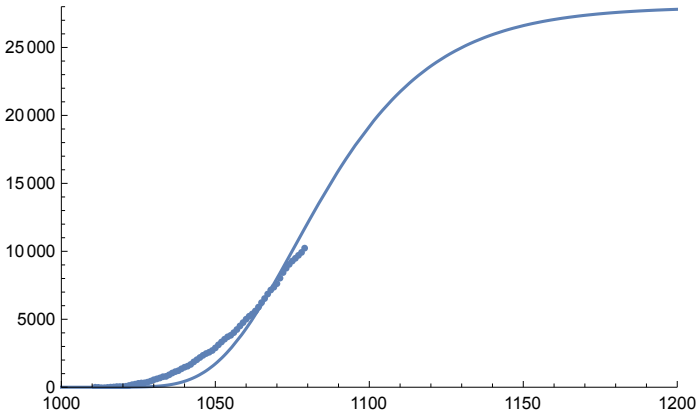


In[ ]:=

In[ ]:=
```
start = 1010;
model[14] = n b^Exp[-c (t - start)];
initialguess = {n → 28 000, c → 0.04, b → 0.000001};
separation = Append[separation, 1010];
Show[
 Plot[model[14] /. initialguess,
  {t, 0, 1200}, PlotRange → {{1000, 1200}, {-10, 28 000}}],
 ListPlot[targetPeriod]
]
```

General : 1. ×10^-6^3.50809 ×10^17^ 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[ ]=



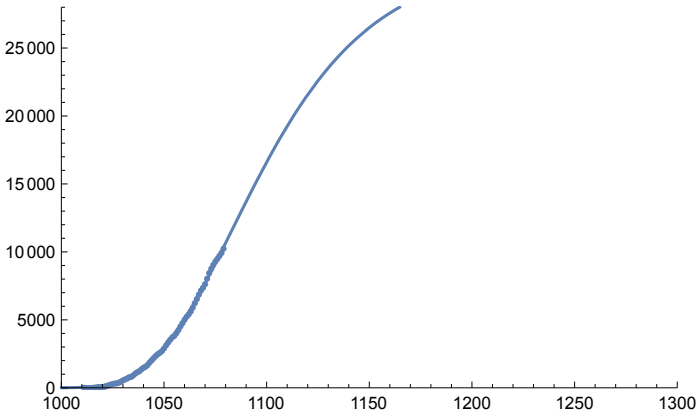
In[ ]:=
```
ans[14] = FindFit[targetPeriod, model[14],
   initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]=
```
{n → 31439.4, c → 0.0263604, b → 0.00106182}
```

In[ ]:=
```
Show[
 Plot[model[14] /. ans[14], {t, 0, 1200},
  PlotRange → {{1000, 1300}, {-10, 28 000}}],
 ListPlot[targetPeriod, PlotStyle → {PointSize[0.01]}]
]
```

General : 0.00106182 3.6508 ×10^11^ 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[ ]=

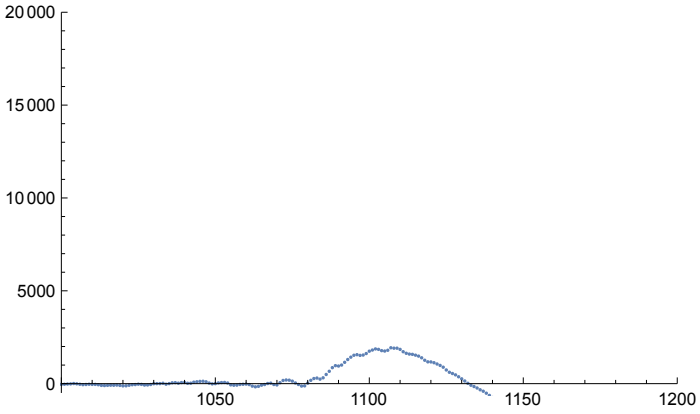


報告時系列のモディファイ

rawModified[15] を生成する。

In[ ]:=
```
{pdays, death} = Transpose[rawModified[14]];
ListPlot[
 rawModified[15] = Transpose@{pdays, death - (model[14] /. ans[14] /. t → pdays)},
 PlotRange → {{1000, 1200}, {-500, 20 000}}]
```

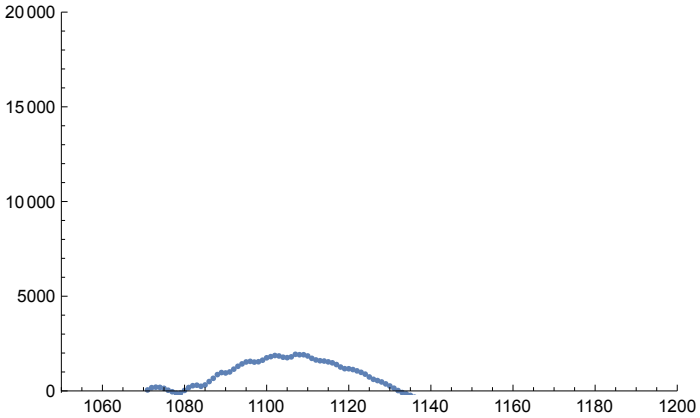
Out[ ]=



15回目-Gompertz曲線

In[ ]:=
```
targetPeriod = Select[rawModified[15], (1070 < #[[1]]) &];
ListPlot[targetPeriod, PlotRange → {{1050, 1200}, {-100, 20 000}}]
```

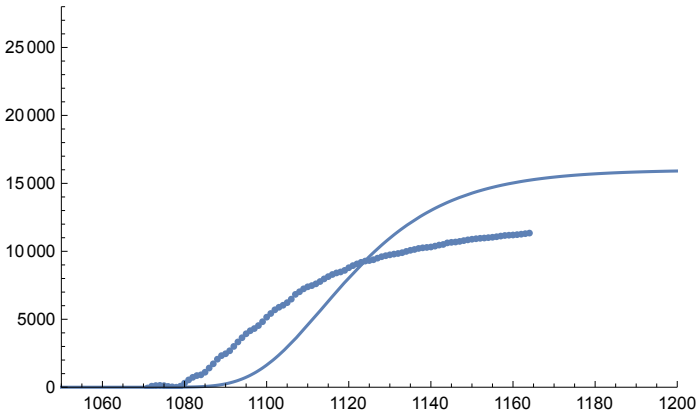
Out[ ]=



In[ ]:=
```
start = 1070;
model[15] = n b^Exp[-c (t - start)];
initialguess = {n → 16 000, c → 0.06, b → 0.000001};
separation = Append[separation, 1070];
Show[
 Plot[model[15] /. initialguess,
  {t, 0, 1200}, PlotRange → {{1050, 1200}, {-10, 28 000}}],
 ListPlot[targetPeriod]
]
```

General : 1. ×10^-67.60444 ×10^27 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[ ]=



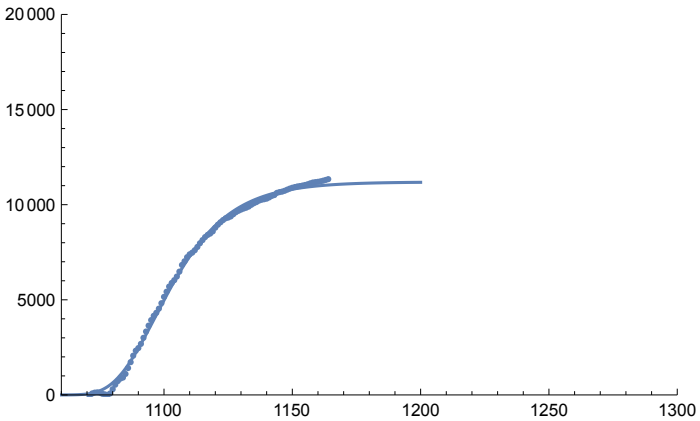
In[ ]:=
```
ans[15] = FindFit[targetPeriod, model[15],
  initialguess /. Rule → List, t, MaxIterations → 300]
```

Out[ ]=
```
{n → 11 191.3, c → 0.063191, b → 0.00455229}
```

In[ ]:=
```
Show[
 Plot[model[15] /. ans[15], {t, 0, 1200},
  PlotRange → {{1060, 1300}, {-10, 20 000}}],
 ListPlot[targetPeriod, PlotStyle → {PointSize[0.01]}]
]
```

General : 0.00455229 2.31132 ×10^29 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[ ]=



計算結果と評価

人口に対する死亡率と死亡者数の推定

各事象モデル model[x]に事象のデータにFittingさせた曲線のパラメータの推定値 ans[x]を適用した各事象モデルを加え合わせた後、時刻tを未来に延長し(t→Infinity)、全体モデルを得る。

```
In[•]:= nf = (model[1] /. ans[1]) + (model[2] /. ans[2]) +
    (model[3] /. ans[3]) + (model[4] /. ans[4]) +
    (model[5] /. ans[5]) + (model[6] /. ans[6]) + (model[7] /. ans[7]) +
    (model[8] /. ans[8]) + (model[9] /. ans[9]) + (model[10] /. ans[10]) +
    (model[11] /. ans[11]) + (model[12] /. ans[12]) + (model[13] /. ans[13]) +
    (model[14] /. ans[14]) + (model[15] /. ans[15]);
  {percent = 100 Divide[nf /. t → Infinity, population], population * percent / 100}
Out[•]= {0.0581505, 73 770.}
```

```
In[•]:= marker = Map[{#, nf /. t → #} &, separation];
```

General : 0.0000112182 $^{1.33758 \times 10^{68}}$ 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 : 0.0000344438 $^{1.35338 \times 10^{17}}$ 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 : 0.0000465129 $^{3.11473 \times 10^{8}}$ 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．

全ての時系列へのフィッティング

得られた14個のGompertz-Logistic曲線の和と報告時系列を表示する。

In[ ]:=
```
fdate = First[ddata][[1]];
ldate = Last[ddata][[1]];
upto = 1300;

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]) +
    (model[7] /. ans[7]) + (model[8] /. ans[8]) + (model[9] /. ans[9]) +
    (model[10] /. ans[10]) + (model[11] /. ans[11]) + (model[12] /. ans[12]) +
    (model[13] /. ans[13]) + (model[14] /. ans[14]) + (model[15] /. ans[15]),
  {t, 0, upto}, PlotRange → {{0, upto}, {0, 85 000}}, PlotTheme → "Grid",
  PlotStyle → {Orange, Thickness[0.005]}],
 ListPlot[raw, PlotStyle → AbsolutePointSize[3],
  PlotLegends → Placed[Text[Style[country <> "\n" <> "Estimated death ratio: " <>
       ToString[DecimalForm[percent, 3]] <> " %", 13, Bold]], {0.3, 0.75}]],

 (*ListPlot[marker],
 Map[markerLegend[marker[[#]]]&,{6,7,8,9,10,11,12,13,14}],*)

 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}]
```

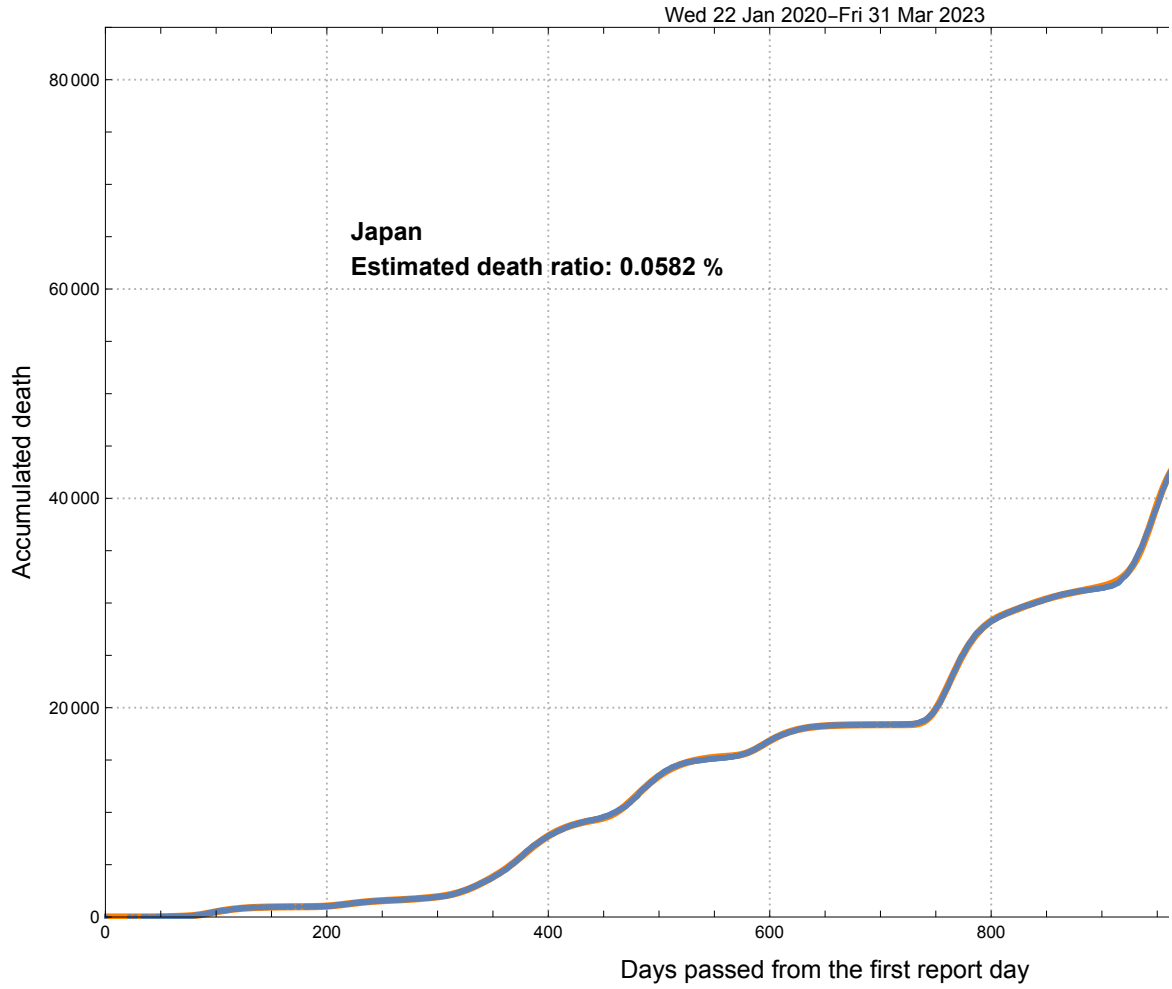
··· General : 0.000939972 $^{8769.57}$ は正規化された機械数として表すには小さすぎます．精度が失われる可能性があります．

··· General : 0.0143594 $^{6335.63}$ 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 : 0.00530514 $^{9502.85}$ 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[ ]=



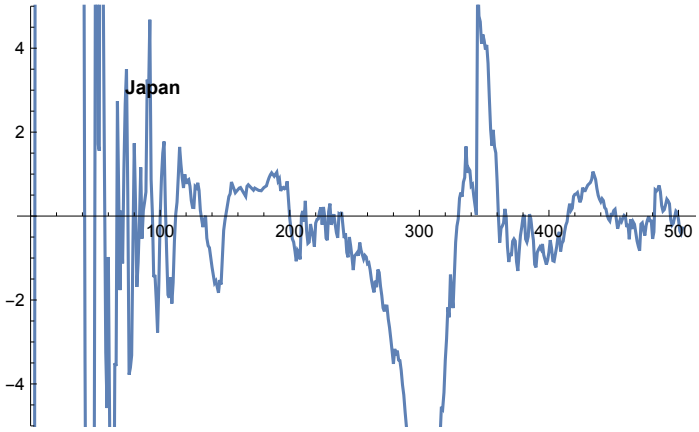
*コメントを外し、ListPlot[marker] を実行すると事象モデルのスタート点が表示される。

モデルと報告値の偏差

モデルと報告値の偏差を%表示してみる。疑問点が提出できる。
1. 各事象毎に、偏差の大きさが違うのは何故か。
2. 200日から300日までの期間の偏差が小さいのは何故か。

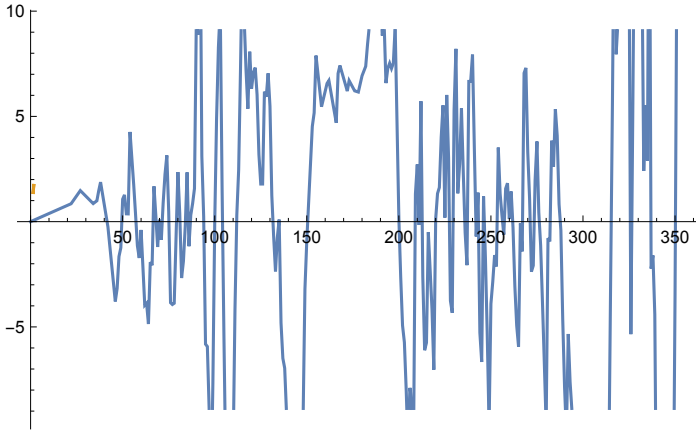
In[ ]:=
```
{pdays, death} = Transpose[raw];
diff =
  Transpose@{pdays, 100 * (death - (et = (model[1] /. ans[1]) + (model[2] /. ans[2]) +
              (model[3] /. ans[3]) + (model[4] /. ans[4]) +
              (model[5] /. ans[5]) + (model[6] /. ans[6]) /. t → pdays)) / et};

ListLinePlot[diff,
 PlotRange → {Automatic, {-5, 5}},
 PlotLegends → Placed[Text[Style[country, Bold]], {0.2, 0.8}]]
```

Out[ ]=




```
ListLinePlot[{diff, EntropyFilter[diff, 1] // N}, PlotRange]
```

Out[ ]=




```
s
```

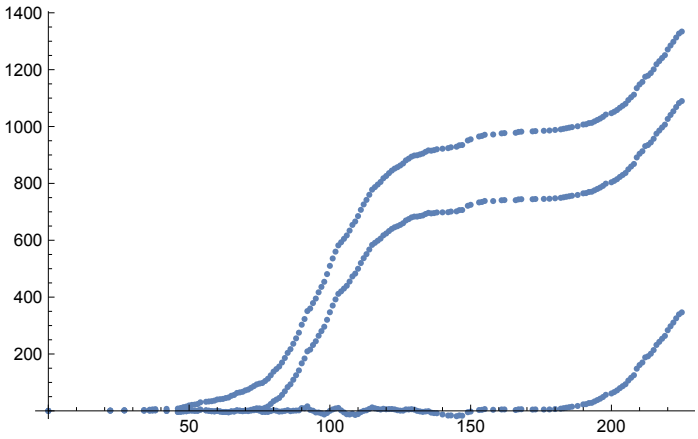
まとめ

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、事象が何らかの物理/社会応答過程に従うという確信の表現であるが、多項式による方法が確信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曲線あるいはLogistic曲線）にフィッティングさせる必要がある。両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