COVID-19（新型コロナウイルス感染症）の新規感染者予測及び外出自粛と感染率の関係

一橋大学経済学研究科
EM195086
LAPISATEPUN SUTTIKAN

2021 年 1 月 12 日提出

概要

あああああああ

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

目次

第 1 章

はじめに

1.1　研究の動機

2020 年は年明けから COVID-19（新型コロナウイルス感染症）[i]の感染が全世界に拡大し、まさに全人類にとって波乱の 1 年であったといえよう。東京オリンピックといった世界規模の大きな催しの延期や外国への渡航の大幅な制限などの国際的な影響もさることながら、緊急事態宣言発令や飲食店の営業自粛、大学における対面授業の禁止などによる身の回りの影響も大きく、2020 年を境に人々の生活は大きく様変わりしたといえる。

個人的な話になってしまうが、著者は当初 COVID-19 に関する話題は経済学にあまり関係がないと思い修士論文のテーマにするつもりはなかった。しかし、COVID-19 に関する研究は今の人々の生活に密接に関わっていることや、大学院ゼミナールで勉強した Python による機械学習を活かせるテーマが望ましかったこと、今年しかできないタイムリーなテーマであるということも相まって、最終的には COVID-19 をテーマにした修士論文を執筆するに至った。

COVID-19 に関する研究は緊急性が求められる話題なだけあって、2020 年に入ってから名だたる疫学者や経済学者が精力的に取り組んできた話題であり、そこに門外漢な大学院生である著者が肩を並べるのはおこがましいことこの上ない。しかし、この修士論文が他の研究者のアイデアのもとになれば幸甚の極みである。

[i] 日本国内では国際的な正式名称である “COVID-19” よりも「新型コロナウイルス感染症」という名称が好んで使われる傾向にあるが、本論文では今後一貫して “COVID-19” という名称を使用する。

1.2　論文の構成

本論文は、タイトルにもあるようにCOVID-19について「新規感染者の予測」及び「外出自粛と感染率の関係」を明らかにすることを目的とする。

まずは、本章の次である第2章で、COVID-19に関する様々な情報をまとめ、同じような分野を扱っている先行研究などを紹介する。

次の第3章では、COVID-19の新規感染者の予測に使われる様々なモデルのうち2つを紹介したうえで、予測方法の違いや精度について比較するとともに考察を行う。その際、予測するのは日本国内感染者であるが、同じような議論はもちろん他の国や地域にも当てはまる。

第4章では、もう一つの目的である「外出自粛と感染率の関係」を明らかにするため、まずは説明変数や被説明変数の構成を議論する。そのうえで、分析の手法及び結果の構成方法を論じたうえで、感染を抑えるために必要とされる外出自粛の程度について考察する。最後には、日本全国ではなく、特定の地域にフォーカスしたうえでより詳しく日本政府の施策と外出自粛の程度の関係性を論じる。

最後の第5章では、第3章や第4章の議論の不十分な点などを検討し、今後の研究課題としてまとめる。また巻末の付録にて、分析を行うとき実際に使用したPythonのコードも掲載しているので、結果の再現等に役立てていただきたい。

第 2 章

COVID-19 について

2.1 COVID-19 の特徴

COVID-19 は、日本語で「新型コロナウイルス感染症」と訳され、SARS コロナウイルス 2（Severe acute respiratory syndrome coronavirus 2; SARS-CoV-2）がヒトに感染して発症する気道感染症である。流行の初期では、最初の感染拡大が見受けられた地域や感染者の症状から「武漢肺炎」などと呼ばれ、世界保健機関（WHO）もウイルスを “2019-nCoV” と命名していたが、2020 年 2 月 11 日に現在のウイルス名及び病名に改めた [1]。ちなみに COVID-19 は **Co**rona**vi**rus **d**isease 20**19** の略称である。

論文執筆当時の 2020 年 12 月現在、COVID-19 の感染経路はウイルスが付着した手で目や鼻、口などを触ることによる接触感染と、咳やくしゃみによる飛沫感染が確認されており、空気感染は確認されていない。症状は無症状のものから死亡までと幅広く、典型的な症状としては発熱、空咳、倦怠感、下痢などがある [2]。同じ論文の報告では、軽症及び中等症例が全体の約 8 割を占め、重症例が 13.8%、重篤例が 6.1% となっている。

COVID-19 は 2019 年 12 月に中華人民共和国の湖北省武漢市で原因不明の肺炎として発生したのが最初とされているが、最近の研究では 2019 年 9 月にすでにイタリア共和国で拡散していたと指摘する研究 [3] もある。その他の世界的な出来事は以下の表を参照されたい。なお、表の内容は NHK の特設ページ [4] 及び世界保健機関（WHO）の公表 [5] を参考にした。

表 2.1　COVID-19 に関連する出来事（全世界）

日付	出来事
2019 年 12 月	中国湖北省武漢で原因不明の肺炎患者発生
2020 年 1 月 5 日	WHO が初めて武漢における肺炎のクラスター感染に言及
2020 年 1 月 13 日	初の中国国外の感染例がタイで確認される
2020 年 1 月 30 日	国際的に懸念される公衆衛生上の緊急事態（PHEIC）に指定
2020 年 2 月 11 日	WHO が感染症を “COVID-19” と命名
2020 年 3 月 6 日	全世界の患者数が 10 万人を超える
2020 年 3 月 11 日	WHO が COVID-19 を「パンデミック」と初めて表現
2020 年 3 月 17 日	中国が治療薬として「アビガン」使用開始
2020 年 3 月 20 日	全世界の死者数が 1 万人を超える
2020 年 4 月 3 日	全世界の患者数が 100 万人を超える
2020 年 4 月 11 日	全世界の死者数が 10 万人を超える
2020 年 5 月 21 日	「レムデシビル」の臨床試験の結果公表
2020 年 6 月 29 日	全世界の患者数が 1,000 万人を超える
2020 年 9 月 10 日	ロシアがワクチンの臨床試験を開始
2020 年 9 月 29 日	全世界の死者数が 100 万人を超える
2020 年 10 月 14 日	ヨーロッパ圏で感染が再拡大しフランスで再度緊急事態宣言
2020 年 11 月 9 日	全世界の患者数が 5,000 万人を超える
2020 年 11 月 19 日	ファイザー社のワクチンに対し「95% の有効性」の分析結果
2020 年 12 月 4 日	全世界の死者数が 150 万人を超える
2020 年 12 月 15 日	アメリカでファイザー社のワクチン接種が始まる

2020 年 12 月 17 日現在、全世界で累計約 7,285 万人の感染者、約 164 万人の死者がおり、1 日あたり約 640,000 人の感染者及び約 12,000 人の死者が新規に発生している。国別にみるとアメリカ合衆国が圧倒的に多く、累計感染者は全世界の約 23%、累計死者数の約 18% を占める。以下の図 2.1 及び図 2.2 はぞれぞれ、棒グラフに大陸別に色づけした累計の感染者や死者を、折れ線グラフに日毎の新規感染者及び死者をプロットしたものである。

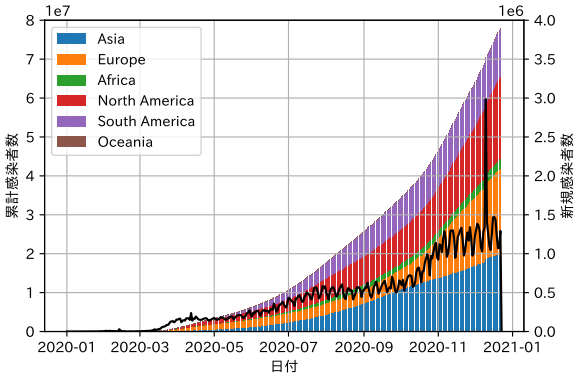


図 2.1　全世界の累計感染者及び日毎の新規感染者

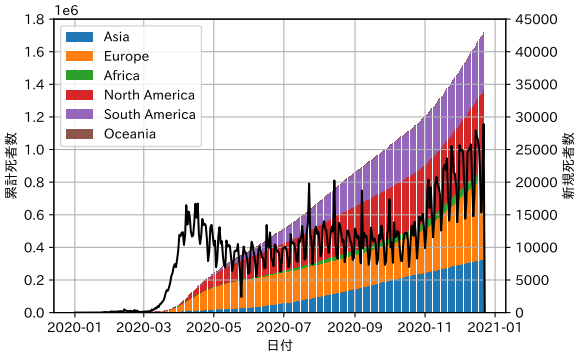


図 2.2　全世界の累計死者及び日毎の新規死者

2.2　日本における COVID-19

本節では、全世界の傾向ではなく日本国内に焦点を当てて、日本特有の傾向や政府の施策などをまとめる。日本国内における COVID-19 に関連する出来事は次ページの年表を参照されたい。

表 2.2　COVID-19 に関連する出来事（日本）

日付	出来事
2020 年 1 月 16 日	日本国内で初の感染者が確認される
2020 年 2 月 3 日	感染者が確認された「ダイヤモンド・プリンセス号」が入港
2020 年 2 月 13 日	日本国内で初めて感染者が死亡
2020 年 2 月 26 日	全国的なイベントの中止・延期・規模縮小を要請
2020 年 2 月 27 日	全国の小中高に対し臨時休校の要請
2020 年 3 月 24 日	2020 年東京五輪の延期発表
2020 年 4 月 1 日	安倍元首相が全世帯にマスク配布の方針表明
2020 年 4 月 7 日	7 都府県に緊急事態宣言
2020 年 4 月 16 日	緊急事態宣言を全国に拡大
2020 年 4 月 18 日	日本国内の感染者数が 1 万人を超える
2020 年 5 月 4 日	緊急事態宣言を 5 月 31 日まで延長
2020 年 5 月 14 日	8 都道府県を残し緊急事態宣言を解除
2020 年 5 月 25 日	全国で緊急事態宣言を解除
2020 年 6 月 19 日	移動自粛の緩和及び休業要請の撤兵
2020 年 7 月 22 日	「Go To トラベル」キャンペーン開始
2020 年 7 月 28 日	クルーズ船を除く国内の感染症死者が 1,000 人を超える
2020 年 8 月 11 日	日本国内の感染者数が 5 万人を超える
2020 年 10 月 30 日	日本国内の感染者数が 10 万人を超える
2020 年 11 月 7 日	感染拡大を受け北海道で警戒ステージの引き上げ
2020 年 11 月 27 日	大阪府で飲食店の営業時間の短縮要請
2020 年 12 月 3 日	大阪府で医療非常事態宣言が発令され、外出自粛が要請される
2020 年 12 月 12 日	5 都道府県で「医療の提供体制が機能不全のおそれ」
2020 年 12 月 15 日	Go To トラベルの全国一斉停止

2.2.1　日本国内の感染の傾向

2020 年 12 月 18 日現在、日本国内のクルーズ船を含めない累計感染者数は 193,765 人で、日毎の新規感染者数は 3,000 人前後となっている。累計感染者数及び新規感染者数の推移は以下の図 2.3 を参照されたい。

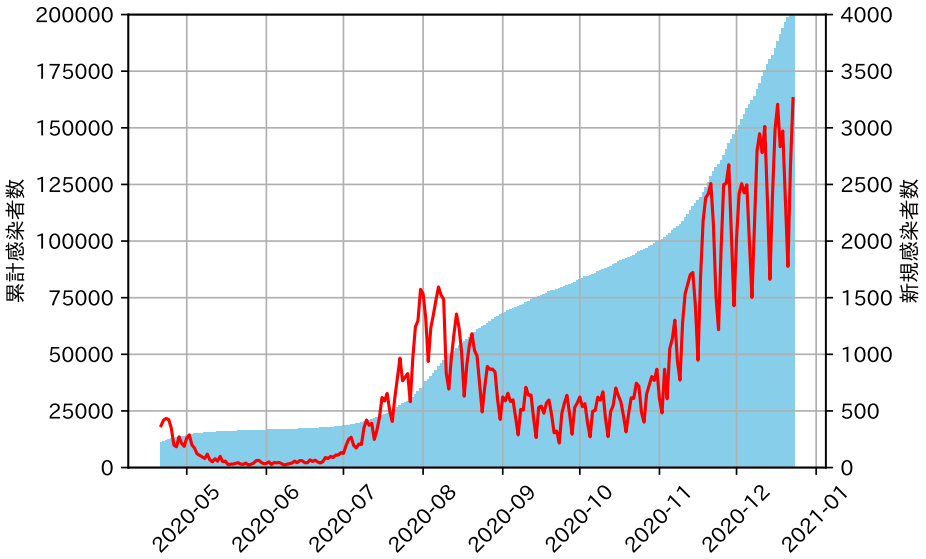


図 2.3　日本国内の累計感染者及び日毎の新規感染者の推移（クルーズ船を除く）

一方で、2020 年 12 月 18 日現在の日本国内における死亡者数は 2,841 人となっており、人口 100 万人当たりの死者数は 20.44 人となっている。これはアメリカの 910.88 人、イギリスの 965.15 人などと比較すると非常に小さい。

2.2.2　「日本モデル」という考え方

日本国内における COVID-19 の感染の傾向を全世界と比較すると、高齢化率が高いにもかかわらず人口当たりの死亡率が低いことが挙げられる。これは COVID-19 の死亡率が高齢者で特に高いことを考えると、かなり特殊な例であるといえる。一般財団法人アジア・パシフィック・イニシアティブが出版した書籍の中では、これの背景に「法的な強制力を伴う行動制限措置を採らず、クラスター対策による個別症例追跡と罰則を伴わない自粛要請と休業要請を中心とした行動変容策の組み合わせにより、感染拡大の抑止と経済ダメージ限定の両立を目指した日本政府のアプローチ」があるとし、これを「日本モデル」

と定義している [6]。また、この書籍によれば新型コロナウイルス感染症対策専門家会議では、具体的に以下のような要素が日本モデルとしての成功につながったと挙げられている。

- 中国及び欧州等由来の感染拡大を早期に検出したこと
- ダイヤモンド・プリンセス号への対応の経験が活かされたこと
- 国民皆保険による医療へのアクセスが良いこと、公私を問わず医療機関が充実し、地方においても医療レベルが高いこと等により、流行初期の頃から感染者を早く検知できたこと
- 全国に整備された保健所を中心とした地域の公衆衛生水準が高いこと
- 市民の衛生意識の高さや（欧米等と比較した際の）もともとの生活習慣の違い
- 政府や専門家会議からの行動変容の要請に対する国民の協力の度合いの高さ

「日本モデル」が国内外で一定の評価を得たとする声もある一方で、根拠のない「37.5 度、4 日間」という基準が保健所で杓子定規のように用いられていたことやマスクや消毒用アルコールなど感染症対策用品の転売への対応の遅れ、中央の政府と地方の自治体の対応の齟齬など批判にさらされた対応も数多い。また「日本モデル」は感染初期である 2020 年 2 月から 2020 年 5 月にかけての対応を指したものだが、2020 年 7 月から 2020 年 9 月にかけての 2 回目の感染拡大、そして 2020 年 11 月以降の感染者の大幅な拡大については、まだ時間が経っていないこともあり有用な説明がなされていない。「日本モデル」のより詳細な内容の解説と結果の分析は当該書籍 [6] を参考にされたい。

2.3 COVID-19 に関する先行研究

COVID-19 については、全世界に影響を及ぼす大型感染症だということもあり 2020 年に入ってから様々な分野で精力的な研究が行われている。中国湖北省の武漢で 2019 年 12 月に感染が拡大した際の初期症状やウイルスの分析は 2020 年 1 月 24 日という早い段階から China Novel Coronavirus Investigating and Research Team により論文 [7] として公表されている。COVID-19 を発生させるウイルスである SARS-CoV-2 のゲノム配列に関する分析は Nature 誌の掲載された Wu, F., Zhao, S., Yu, B. *et al.* による論文 [8] に詳しい。

SARS-CoV-2 のタンパク質の分析に使われる分子動学シミュレーションには多くの

コンピュータによる計算量を必要とする。そこで、分散コンピューティング[i]プロジェクトである “Folding@home” が早期から新型コロナウイルス関連のプロジェクトを創設し、様々な研究所や企業、個人が所有しているサーバーやコンピュータの計算力を提供した。2000 年に始まったこのプロジェクトは注目を集めていなかったが、今回の SARS-CoV-2 関連のプロジェクトを開始してから参加者が大幅に増え、2020 年初頭にはわずか 30,000 台以下だった接続デバイス数がおよそ 3 か月後には 120 万台以上となった。Folding@home の発表によれば、プロジェクト全体の処理速度は一時期 2.4exaFLOPS[ii] を突破し、全世界の上位 500 台のスーパーコンピュータを合わせても勝る計算量を記録した [9]。Folding@home による分析結果は、ウェブページをはじめ出版されている論文 [10] から確認できる。

COVID-19 を含めた感染症の感染の過程を記述するにあたって、1 人の患者が何人に感染を広げる可能性があるかを示す基本再生産数（Basic Reproduction Number, $\mathcal{R}_0$）を推定することが重要である。しかし、COVID-19 の感染には多くの要因が複雑に絡み合っていることや推定手法が多様であることも相まって、なかなか基本再生産数の値の範囲を狭められていないのが現状である。感染初期の推定では基本再生産数は 0.8 から 5.0 の間とされており [11–13]、感染拡大からしばらく経った 2020 年 9 月の段階では基本再生産数は 2.39 から 3.44 とする研究が発表されている [14]。基本再生産数や導出の過程、背後にある疫学的モデルなどに関しては、第 3 章の新規感染者予測で詳しく取り上げる。

経済学の領域においては、COVID-19 は感染拡大と経済活動の両立というテーマが扱われる。感染症は、単純に外出などを一切禁じ接触などによる感染を抑えれば感染拡大は抑えられるが、そうした場合経済活動はほとんどすべて停止し、感染は抑えたものの経済の悪化によって人々の生活に悪影響を与えてしまうことになる。経済学の分野では、例として以下のような問いが扱われることが多い。

- COVID-19 の感染拡大は、経済（GDP や失業率、消費者行動など）にどのような影響を与えるか?
- テレワークなどの感染拡大を抑えた勤務体制の生産性は平常時に比べるとどれくらい違うか?
- 感染拡大と経済活動のトレードオフ関係はどのようにモデル化されるのか?

[i] 1 つのプログラムの個々の処理を複数のコンピュータ上で行い、それぞれがネットワークを介して結果の通信を行うことで全体としての処理を進行させる計算手法。

[ii] FLOPS は “FLoating point number Operations Per Second” の略で、1 秒間に浮動小数点演算ができる回数を示す。exa は 10^{18} を表す接頭辞である。つまり、2.4exaFLOPS とは 1 秒間で 240 京回の浮動小数点演算ができることを意味する。

- トレードオフを踏まえたうえで、感染の抑制と経済の維持を両立できるような政策は何か?

経済学者による COVID-19 と経済の関係性の分析は大まかに分けて、疫学モデルと経済モデルの融合による施策の提言と、事後データを用いた施策の評価の 2 つに分けられる。前者の例としては、人口を年齢別に分けて疫学モデルである SIR モデルを適用し、高齢者のみ外出規制をすることが最も感染抑制と経済活動の両立を実現できると主張した論文 [15] や、中央政府の所得の再分配機能に焦点を当て、疫学モデルを組み込んだ経済モデルを用いることにより再分配によるコストと外出規制による経済活動縮小のコストを比較した論文 [16]、消費者行動に焦点を当て、COVID-19 の死者などの情報が個人の消費や労働時間の削減にどのくらい影響を及ぼすか、ひいては社会全体としてどれくらい経済的に影響があるかを分析した論文 [17] などが挙げられる。後者の例としては、アメリカにおける COVID-19 感染初期のデータを扱い、マスク着用義務化の有無や、在宅勤務の義務化の有無によって感染者がどのように変わったかを分析した論文 [18] や、集団免疫の獲得を目指し感染抑制対策をほとんど行わなかったスウェーデンのデータを用いて、対策をした場合感染状況や経済状態がどれくらい好転しえたのかを論じる論文 [19] などが挙げられる。日本国内の分析については、日本国内の人々の行動の変化を、政府の外出自粛要請などの「政策効果」と政府の発表などにより危機感が高まり行動を変化させる「情報効果」の 2 つに分解し、どちらの効果のほうがより人々の行動変化に影響を及ぼしたかを論じる Tsutomu Watanabe と Tomoyoshi Yabu による論文 [20] の論文が例として挙げられる。また、直接政策と効果を結び付けた分析は行ってはいないものの、日本における政府の対応とその効果を詳細にまとめたものとして、前節でも紹介した書籍 [6] が非常に有用である。

第 3 章

COVID-19 の新規感染者の予測

2020 年度の竹内幹ゼミナールで Python のプログラミングを学習していた際に、日本国内最大規模の AI コンペティションサイトである SIGNATE というウェブサイトで、COVID-19 の新規感染者予測のコンペティションが行われていることを知った。SIGNATE の「COVID-19 Challenge」というコンペティションは、データセット構築を目指すフェーズ 1、統計的手法を用いてインサイトの抽出を目指すフェーズ 2 を経て、2020 年 4 月 16 日以降は国内全体の累積感染者数を予測するフェーズ 3 に突入しており[i]、このコンペティションは「日本国内収束（新規罹患者が 0 または 社会的受容）まで」行うとしている [21]。この章では、COVID-19 の新規感染者予測に使用した「SIR モデル」と「SARIMA モデル」の理論を検討し、それぞれの分析結果を比較することを目的とする。

3.1　モデルの分類

Christopher Avery, William Bossert, Adam Clark, Glenn Ellison, そして Sara Fisher Ellison が 2020 年 4 月に投稿したワーキングペーパー [22] では、疫病の感染を記述する数学的モデルを “機構的モデル（Mechanistic）” と “現象学的モデル（Phenomenological）” の 2 つに大別している。これは経済学でいう構造的モデル（Structural）と誘導形モデル（Reduced Form）に似ていると Avery et al. は指摘する。この 2 つのモデルの違いとしては、機構的モデルが疫病の感染拡大の背景にあるメカニズムやパラメータを推定する目的で使用されるのに対し、現象学的モデルはそういった背景にあまり立ち入らず将来の予測などに重点を置く。この 2 種類のモデルのさらなる比較は、Amy Hurford による記

[i] 予測期間は 2020 年 8 月 30 日までのラウンドは 2 週間だったが、それ以降は 4 週間に変更となった。

事 [23] を参考にされたい。

3.2 SIR モデルによる予測

SIR モデルは、W. O. Kermack と A. G. McKendrick が 1927 年に発表した論文 [24] に端を発する、感染症の短期的流行の過程を示した古典的なモデル方程式で、前節でいう機構的モデルに該当する。古典的な SIR モデルは 3 つの常微分方程式で示されるというシンプルかつわかりやすいものであり、その単純さゆえ多くの研究者により様々なモデルの拡張が存在する。

3.2.1 SIR モデルの理論

SIR モデルの基本的な記述は以下の通りである。

まずは年齢構造を考えない N 人の人口集団を考える。この人口集団は、以下の 3 つのグループのいずれかに所属する。

1. 感染性保持者（Suspectible，S）：感染する可能性のある者
2. 感染者（Infected，I）：感染しており、他者に感染させることができる者
3. 免疫保持者（Recovered / Removed，R）：感染から回復したか感染により死亡した者

このとき、流行のダイナミクスは以下のような常微分方程式で与えられる。

$$\frac{dS(t)}{dt} = -\beta \frac{S(t)I(t)}{N}, \tag{3.1}$$

$$\frac{dI(t)}{dt} = \beta \frac{S(t)I(t)}{N} - \gamma I(t), \tag{3.2}$$

$$\frac{dR(t)}{dt} = \gamma I(t). \tag{3.3}$$

この数式の様々な文字は以下の意味を持つ。

- β：感染率
- γ：回復率
  - 回復率は発症期間（Infectious Period：感染者になった者が他者へ感染を引き

起こす期間）を用いて以下のように表される[ii]。

$$\gamma = \frac{1}{i_p}. \tag{3.4}$$

特に、$\beta N/\gamma$ で表される無次元量を基本再生産数（Basic Reproduction Number, $\mathcal{R}_0$）と呼ぶ。つまり、サイズが N の集団に発生した感染者は、1 人あたり単位時間あたり βN の二次感染者を発生させる。ここに発症期間 $i_p = 1/\gamma$ をかけることによって、感染者が感染性でいる間に生産する二次感染者の総数が計算される。

また、SIR モデルは簡略化のために以下のような仮定を置いている。

- モデルは十分に短い期間を扱うため、出生による人口増加や自然死亡による人口減少は考えないものとする。つまり、(3.1)〜(3.3) 式について、

$$\frac{dS(t)}{dt} + \frac{dI(t)}{dt} + \frac{dR(t)}{dt} \equiv 0. \tag{3.5}$$

  が常に成り立つ。
- 一度感染し回復したものが再び感染することはない（感染による死亡者も当然再度感染することはないので、この 2 グループを合わせて免疫保持者としている）。

SIR モデルは単純で分かりやすいものではあるが、単純すぎるゆえ様々な感染症に完全に対応できない。そこで、これまで数多くの研究者が SIR モデルの拡張版を行っており、そのほとんどは S, I, R のそれぞれのグループをより細分化させたものである。その SIR モデルの派生版のいくつかについては本節の最後に紹介する。

3.2.2 SIR モデルのシミュレーション

以下のグラフは、人口 1.2 億人のうちある感染症の初期感染者が 100 人、免疫保持者が 0 人であるときの 100 日後までをプロットしたものである。感染率 β は 0.5、回復率 γ は 0.1 に設定してある。先の議論でいえば、この感染症の基本再生産数 $\mathcal{R}_0$ は $0.5/0.1 = 5$ であり、1 人の感染者が回復するまでに平均して 5 人にうつすという意味である。

[ii] 一般に確率密度関数が $f(X;\lambda) = \lambda e^{-\lambda X}$ で表されるような指数確率分布の期待値は $E(X) = \frac{1}{\lambda}$ で表される。

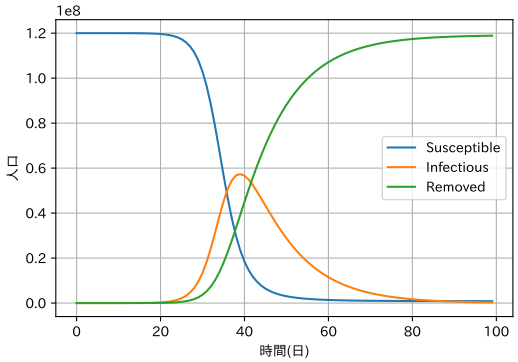


図 3.1　SIR モデルのシミュレーション

このグラフを見ると、初期の感染者は全人口に比べわずかでしかないにもかかわらず、20 日から 40 日前後にかけて指数関数的に感染者が増え、ピーク時には全人口の半分近くまで感染者が増えていることがわかる。しかし、感染者が増えるのに少し遅れて免疫保持者も増えている。免疫保持者は再度感染しないという仮定から、免疫保持者が増えれば増えるほど感染性保持者は減少し、感染者も減少する。100 日後には全人口のほとんどが免疫保持者になり、感染は収束する。

3.2.3　SIR モデルのパラメータ推定

前節では、初期値やパラメータをあらかじめ設定し、将来の感染の傾向をシミュレーションした。そこで逆に、すでにある感染者や免疫保持者の感染状況からパラメータを推定すれば、そのパラメータを使って将来の感染の傾向を予測することができそうである。そこで本項では、政府機関が公表しているデータセットを利用し、感染率 β と回復率 γ の 2 つのパラメータの値を推定する。

使用するデータセットについては、日毎の累計感染者数については内閣官房新型コロナウイルス感染症対策推進室のデータセット [25] を、日毎の累計の回復者及び死亡者については厚生労働省のオープンデータ [26] を用いる。データは 2020 年 5 月 8 日から執筆時点で最新である 2020 年 12 月 8 日までである。なお、全人口は 1.265 億人とする。

推定方法は、まずパラメータの初期値や値域を設定する。つぎに、あるパラメータで推定した日毎の感染者数、免疫保持者数を実際のデータの感染者数、免疫保持者数と差分を取るような誤差関数を定義する。最後に非線形最小二乗法の一つである Levenberg-Marquardt 法を用いて、誤差関数が最小となるようなパラメータを探索するというものである。より詳細な手順については、巻末付録の Python コードを参照されたい。

こうして得られた感染率 β や回復率 γ の推定値は以下のとおりである。

表 3.1　SIR モデルのパラメータ推定結果

パラメータ	推定値	標準誤差	相対誤差	初期値
β	0.0214	$\pm 9.0079 \times 10^{-5}$	0.42%	0.5
γ	0.0105	$\pm 1.0154 \times 10^{-4}$	0.97%	0.1

この表に初期値を掲載している理由は、Levenberg-Marquardt 法が初期値を出発し局所最適解を探索するものであるからである。したがって、初期値が大きく異なる場合、局所最適解が違うものになることもある。

最後に得られたパラメータを用いて、改めて SIR モデルのシミュレーションを行い実際の感染者の推移と比較する。

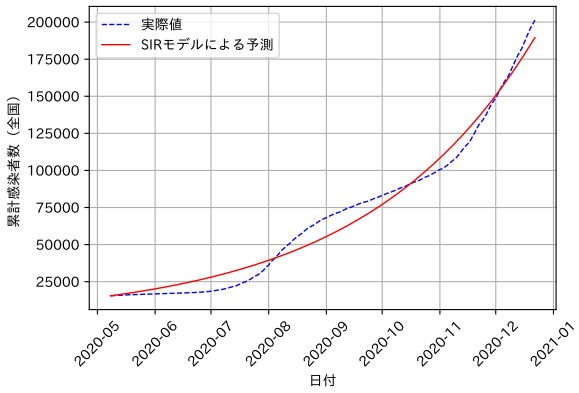


図 3.2　SIR モデルのシミュレーション

3.2.4　SIR モデルの拡張や問題点

前項の図 3.2 では、実際の感染者数とパラメータ推定を行った SIR モデルの感染者数を比較したが、これを「細かい誤差はあれど大体の傾向をつかめている」と肯定的に捉えるか「実際の感染者数との乖離が大きい」と否定的にとらえるかは意見が分かれるところである。SIR モデルはパラメータが 2 つしかなく、それぞれのパラメータの意味も分かりやすいものであるという利点は確かに存在するが、かなり単純化されたモデルであるため細かい変化を組み込むことは難しい。

SIR モデルの精度をより高くしようとする試みはこれまで数多くなされており、その

多くはすでにある感染性保持者、感染者、免疫保持者の 3 グループのいずれかの細分化によって得られる。例えば感染者をさらに「感染しているが潜伏期にある者（Exposed）」と「実際に発症した者（Infectious）」の 2 グループに分ける SEIR モデルや、免疫保持者を「感染症から回復した者（Recovered）」と「死亡した者（Dead）」の 2 グループに分ける SIRD モデル、あるいは免疫獲得を考慮せず、感染から回復しても感染性保持者に戻るような SIS モデルが SIR モデルの拡張のうち一般的なものである。COVID-19 の分析においても、SIR モデルを使った例 [27] や SIRD モデルを使った例 [28]、そして中には SEIR モデルをさらにソーシャル・ディスタンスを実施した者（Confined）、隔離された者（Quarantined）などを加えた SEIR+CAQH モデルなどといった例 [29] も存在する。このように、SIR モデルの拡張のバリエーションは枚挙にいとまがないが、より詳細な SIR モデル拡張の理論については Brauer, van Den Driessche, Wu による本 [30] が詳しい。

また、そもそも常微分方程式を用いた SIR モデル群が COVID-19 の傾向をとらえられないという指摘もある。James H. Stock が 2020 年 4 月に投稿した論文 [31] によると、そもそも感染者のデータによく用いられている数字はポリメラーゼ連鎖反応（PCR）による検査を受けた者のうち陽性となった人数であるが、PCR 検査の上限は時期や地域によって違う上、そもそも COVID-19 に近い症状が出た者が優先的に検査されるため、感染しても無症状である者が数字に含まれないと指摘したうえで、PCR 検査陽性者を SIR モデルにおける感染者として扱うことに異議を唱えている。同様に回復率 γ についても、無症状の人は回復にどれくらいかかるかが分かっていないため、推定される回復率 γ があくまで発症者の回復率であると指摘している。また、SIR モデルは初期値鋭敏性を持ち、初期値の設定が重要になってくると主張する論文 [32] もある。

そして、SIR モデルのパラメータが常に一定であることに異議を唱える研究者も多い。たとえば、COVID-19 における感染率 β や回復率 γ と組み合わせることによって導出される基本再生産数 $\mathcal{R}_0$ は気温や湿度、人口密度、接触の程度によって異なってくるはずであり、それを一定とすることによって予測の精度が下がるのはある種必然のことである。これの解決方法として、SIR モデルの連続時間的な常微分方程式を離散時間的な差分方程式に置き換える手法 [33], [34] や、実効再生産数 $\mathcal{R}$ （すでに感染が進んでいる集団内で、1 人の感染者が単位時間あたりに感染させる二次感染者数）をベイズ推定で更新していく手法 [35] 、実効再生産数 $\mathcal{R}$ を感染症患者の情報群から直接計算する手法 [36] などが挙げられる。特に差分方程式に置き換える手法は次章の外出自粛と感染率の関係において分析に使用される。

3.3　SARIMA モデルによる予測

SARIMA（Seasonal AutoRegressive Integrated Moving Average）モデルとは、時系列モデルである自己回帰（Auto Regressive; AR）モデル、移動平均（Movine Average; MA）モデル、そして和分（Integrated; I）を組み合わせた ARIMA モデルに、さらに季節変動を組み込んだ時系列モデルであり、3.1 節のモデルの分類における現象学的モデルに該当する。

なお、前節の SIR モデルとは異なり、SARIMA モデルでは日毎の累計感染者ではなく日毎の新規感染者を予測し、それを足し合わせることで累計感染者数を予測する。

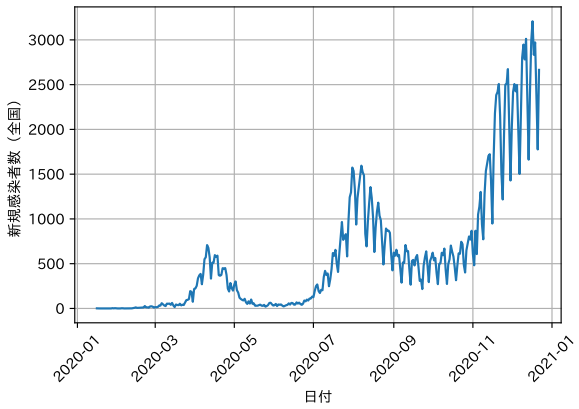


図 3.3　日本国内の日毎の新規感染者数の推移

上の図 3.3 を見ると、新規感染者数が周期的に上下している様子がうかがえる。これは新規感染者数が PCR 検査数と比例しているからであり、その PCR 検査数は曜日によって異なる（具体的には、平日は検査数が多く、休日は検査数が少ない）。新規感染者数の予測に季節変動を組み込んだ時系列モデルを適用するのはこのためである。

誤解を恐れずにいえば、SARIMA モデルは非定常過程（分散が時間を通じて一定でない過程）の観測データをトレンド（大まかな傾向）、季節変動（周期的に現れる変動）、そして残差の 3 つに分解する。予測をする際は、トレンドの予想値に季節変動を加えることで精度を高めることを可能にする。イメージとしては以下の図を参考にされたい。

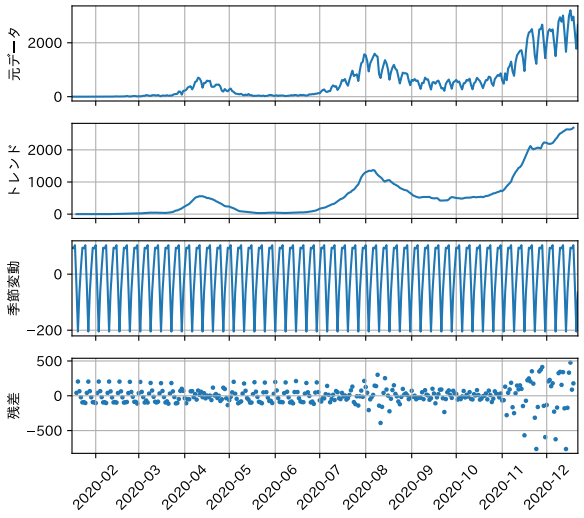


図 3.4　新規感染者データの成分分解

3.3.1　SARIMA モデルの理論

前述した通り、SARIMA モデルは ARIMA モデルに季節変動を加えたものになっている。

ARIMA モデルの土台になっているのは以下の 2 つのモデルである。

1. AR(p) モデル：値の回帰にその系列の p 期前までの過去の値を用いる自己回帰モデル

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \varepsilon_t \tag{3.6}$$

2. MA(q) モデル：値の回帰にその系列の q 期前までのノイズを用いる移動平均モデル

$$y_t = c + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{3.7}$$

この 2 モデルを組み合わせたのが、ARMA(p, q) モデルである。

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{3.8}$$

ARMA モデルは、主に時間によらず期待値や自己共分散が一定である定常過程の分析に使われるが、非定常過程であっても差分を取ることで定常過程になる場合がある（これを

d 次和分モデルと呼び、$I(d)$ と記す)。d 階差分をとった系列に対し ARMA(p,q) モデルを考えるとき、ARIMA(p,d,q) モデルと呼ぶ。

SARIMA モデルは、この ARIMA(p,d,q) モデルに季節変動周期 s を組み込んだものである。その際、季節変動周期 s を組み込むのは時系列方向の ARIMA(p,d,q) と季節差分方向の ARIMA(P,D,Q) となるため、次数は $(p,d,q),(P,D,Q),s$ の 7 つとなる。

3.3.2 SARIMA モデルのパラメータ推定

前項で、SARIMA モデルのパラメータは $(p,d,q),(P,D,Q),s$ の 7 つであるとしたが、COVID-19 の新規感染者の傾向で唯一分かっているのは PCR 検査数による季節変動の周期が 7 日であることだけである。そこで、残りのパラメータは総当たりで回帰し、AIC（赤池情報量規準）が最小であるものを最適な次数として選択した。総当たりのパターンについては AR の次数 p は 1 から 3 の 3 通り、MA の次数 q は 0 から 3 の 4 通り、そして残りの I の次数 d や季節差分方向の次数 (p,d,q) については 0 か 1 の 2 通りとし、計 192 通りとした。総当たりのパターン総数は組合せ爆発を起こすので、次数を増やす際は注意が必要である。

使用するデータは、SIR モデルでも使用した厚生労働省のオープンデータ [26] である。論文執筆時の最新情報である 2020 年 12 月 12 日までのデータを用いてパラメータ探索を行うと、$(p,d,q)=(2,1,1)$ 及び $(P,D,Q)=(0,1,1)$ が最も AIC の少ない値となり、より詳細な結果は以下の通りとなった。 ただし推定値はいずれも有意な係数である $(p<0.01)$。

表 3.2　SARIMA モデルのパラメータ推定結果

パラメータ	推定値	標準誤差	95% 信頼区間
AR(1)	0.5350	0.084	[0.370, 0.700]
AR(2)	-0.2448	0.042	[-0.327, -0.163]
MA(1)	-0.6048	0.090	[-0.781, -0.428]
S.MA(1)	-0.6043	0.029	[-0.662, -0.547]

以下は、得られたパラメータで 2021 年 1 月 31 日までの新規感染者を予測した図と、それらの総和として計算された累計感染者数の予測の図である。図を見てもわかるとおり、おおむね新規感染者の傾向を正しく捉えられているといえる。

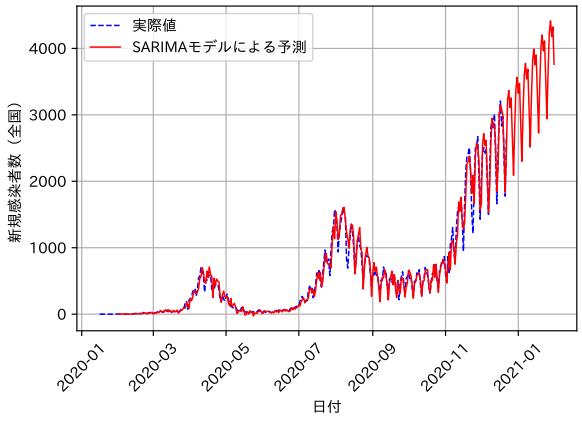


図 3.5　SARIMA モデルによる新規感染者の予測

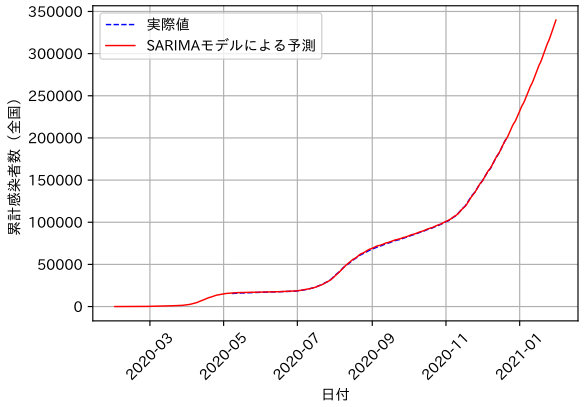


図 3.6　SARIMA モデルによる累計感染者の予測

3.3.3　SARIMA モデルの問題点

SARIMA モデルは、今回用いた COVID-19 の日本国内新規感染者データのように明確に季節変動があることが分かっているデータの扱いを得意とする。しかし、前項の図 3.2 の各係数の推定値を見ても直観的に新規感染者の増減の理由がわからないように、SARIMA モデルは現象の背後にどういうメカニズムがあるかについての説明に乏しい。これは、SARIMA モデルが予測に重点を置く現象学的モデルだからである。

そして、新規感染者数の総和をもって累計感染者数を算出する場合、本来は日毎の回復者や死者（いわゆる免疫保持者）を差し引かなければいけない。今回のデータは感染者数に比して差し引かなければいけない人の数は少なかったので無視したが、より精度の高い

分析を行うなら免疫保持者の傾向も分析し予測を行わなければならない。

3.4　2 つのモデルの比較

最後に、SIR モデルと SARIMA モデルの 2 つのモデルを用いて同じデータの分析を行い結果を比較する。パラメータ推定に充てる期間は 2020 年 5 月 8 日から 2020 年 10 月 31 日までとし、予測期間は 2020 年 11 月 1 日か 11 月 30 日までの 1 か月間とする。

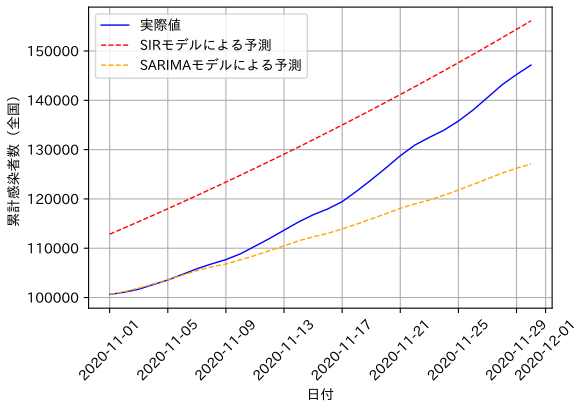


図 3.7　SIR モデルと SARIMA モデルの予測結果の比較

モデルの精度評価を、SIGNATE の「COVID-19 Challenge」と同じ MSE（平均二乗誤差）で計算したところ、SIR モデルが約 186,000,000、SARIMA モデルが約 89,200,000 となり、平均して SARIMA モデルの 1 日当たりの誤差が SIR モデルのそれよりも 5,000 人ほど少ないという結果となった。

ただし、この結果をもって SIR モデルよりも SARIMA モデルのほうが優れていると結論付けるのは間違っている。そもそも、SIR モデルは将来予測に使われるようなモデルではないので、将来予測に使われる SARIMA モデルと結果を比較すれば後者に軍配が上がるのは当然のことである。これは本章の冒頭に示した機構的モデルと現象学的モデルの違いとして説明される。また、学習及び予測に使う期間を変えれば、結果が違ってくる可能性も考えられる。

予測の精度という観点ではあまり優れなかった SIR モデルだが、このモデルには得られたパラメータの解釈がしやすいという SARIMA モデルにはない利点が存在する。そこで、次章では SIR モデルに手を加え、パラメータをより解釈しやすいものへと変更し分析を進めていく。

第 4 章

外出自粛と感染率の関係

この章では「外出をどのくらい減らせば COVID-19 の感染拡大を抑えたり、医療の崩壊を防ぐことができるか」という問いに回答を与えることを目的とする。そのために、まずは被説明変数となる感染の度合いを示す指標や説明変数となる外出を減らす度合いを定義し、線形回帰モデルを用いて分析を進めていく。

被説明変数となる感染の度合いを示す指標については、前章で取り扱った SIR モデルを離散時間的なものに置き換えることで計算するとし、説明変数となる外出を減らす度合いについては外部のデータを用いる。そして線形回帰を行なった後は、得られた推定量を用いて感染者数の今後の傾向を推測したり、それによって医療の崩壊がどの程度で起きるのかを考察する。

4.1 線形回帰モデルの構築

4.1.1 被説明変数：離散的 SIR モデルに基づく感染率

前章で取り扱った SIR モデルでは、期間全体を通じて感染率 β や回復率 γ が一定であると仮定した。しかし、これらのパラメータは期間を通じて一定であるよりも、気候状況や人々の外出の傾向などによって常に変化すると考える方が自然である。本節では Y. C. Chen、P. E. Lu、C. S. Chang、T. H. Liu の 4 名による論文 [34] にて紹介された手法に則り、日毎の感染率 β_t や回復率 γ_t の導出を行う。

まずは、連続時間的な SIR モデルである前章の (3.1)〜(3.3) 式を離散的な SIR モデル

に書き換える。

$$S_{t+1} - S_t = -\beta_t \frac{S_t I_t}{N}, \tag{4.1}$$

$$I_{t+1} - I_t = \beta_t \frac{S_t I_t}{N} - \gamma_t I_t, \tag{4.2}$$

$$R_{t+1} - R_t = \gamma_t I_t. \tag{4.3}$$

ここで、全人口に対する感染者数が十分に低いと仮定すると、$S_t \approx N$ とすることができる。これを (4.2) 式に代入すると

$$I_{t+1} - I_t = \beta_t I_t - \gamma_t I_t. \tag{4.4}$$

さらに、(4.3) 式を移行すると

$$\gamma_t = \frac{R_{t+1} - R_t}{I_t}. \tag{4.5}$$

最後に、(4.4) 式に (4.5) 式を代入すると、β_t を推定する式が得られる。

$$\beta_t = \frac{(I_{t+1} - I_t) + (R_{t+1} - R_t)}{I_t}. \tag{4.6}$$

日本国内の累計感染者や累計回復者などのデータは日毎に提供されているので、これを (4.6) 式に基づいて計算することによって、日毎の感染率が計算できる。

以下に計算した日本国内の感染率 β_t とその 7 日移動平均のグラフを載せる。

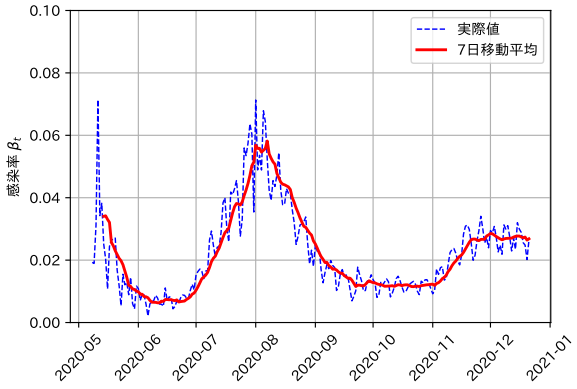


図 4.1　日本国内における日毎感染率 β_t の推移

4.1.2　説明変数：モバイル空間統計

説明変数となる外出自粛の度合いには、NTT ドコモ社が提供する「モバイル空間統計新型コロナウイルス感染症対策特設サイト」の人口増減率の推移のデータ [37] を用いる。データは全国 47 都道府県の 95 地点における人口増減率の感染拡大前比、緊急事態宣言前比、前年同月比、そして前日比の 4 つのデータが提供されている。今回は説明変数として、全 95 地点の日毎の感染拡大前比人口増減率の単純平均を取り、符号を逆転させたものを「移動削減率」と名付け説明変数とする。つまり、この移動削減率が大きければ大きいほど、人々は外出を減らし全国の地点で人口増減率の減り具合が大きいことを示す。

以下に移動削減率とその 7 日移動平均のグラフを載せる。グラフを見るとわかるとおり、移動削減率は緊急事態宣言が全国的に発令されていた 2020 年 5 月初旬から徐々に減少を続けており、7 月から 9 月にかけて 20% 前後で推移していた。その後 10 月に入ってからは 10% 前後を推移している。

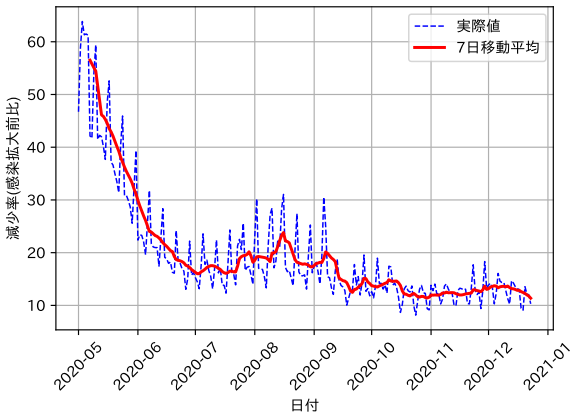


図 4.2　日本国内における移動削減率の推移

余談だが、モバイル空間統計と似た指標として Google 社が提供する “COVID-19 Community Mobility Reports” [38] が挙げられる。これは世界各地のモビリティ（訪問数や滞在時間）を曜日別の基準値（2020 年 1 月 3 日〜2 月 6 日の 5 週間の曜日別中央値）と比較した増減率である。このデータは世界各国のデータが用意されているうえに、モビリティも retail_and_recreation（小売、娯楽）、grocery_and_pharmacy（食料品店、薬局）、parks（公園）、transit_stations（乗換駅）、workplaces（職場）、residential（住宅）の 6 つに分かれて集計されているのが利点である反面、どのような方法で計算されているかが不明で曜日間のつながりがないので今回の日本国内の分析においては不採用とした。

4.1.3 「効果は 2 週間後」の妥当性

本章で扱う問いは「移動削減率をどれくらいにすれば感染率を抑えることができるか」というものであるが、移動を削減した翌日に感染率がすぐに下がることはない。政府の発表や様々なニュースを見ていると「外出自粛の効果が表れるのは 2 週間後」という表現がよくみられるが[i]、著者の知る限りこれに明確な根拠はない。恐らく COVID-19 の感染可能期間が 2 週間とされていることから、2 週間経てば COVID-19 に罹患している人が二次感染者を確実に発生させることがなくなるというを根拠にしていると予想される。本項では改めてこの主張の妥当性を検討したい。

検討の手法は以下のとおりである。前項で論じた移動削減率を説明変数に、そして離散的 SIR モデルに基づく日毎感染率 β_t を被説明変数に置き線形回帰を行う。その際被説明変数の日毎感染率 β_t を 1 日毎にずらし、0 日後の感染率、1 日後の感染率、2 日後の感染率、$\cdots$、28 日後の感染率の計 28 個の被説明変数のデータセットを作る。そのそれぞれについて線形回帰を行い、回帰係数や 95% 信頼区間を得る。

被説明変数を 0 日から 28 日（4 週間）シフトしたときの回帰係数及び 95% 信頼区間の推移は以下のとおりである。

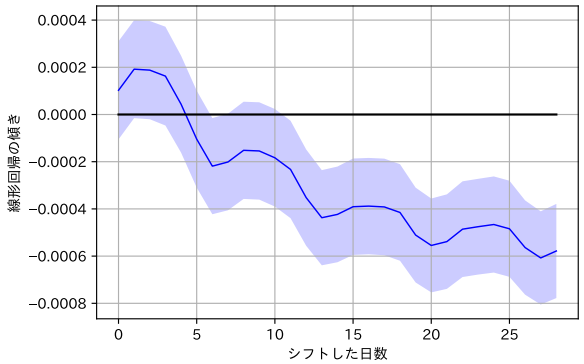


図 4.3　シフトした日数と回帰係数の推移

この図 4.3 を見ると、回帰係数の推定値はシフト日数が 5 日以上であるときに負となる。これはつまり「今日の移動削減率が、5 日後以降の感染率に負の影響を及ぼす」と言い換えられる。また、95% 信頼区間をみると、シフト日数が 11 日以上になると 95% 信頼区間の全域にわたって負の値を取る。このため、本論文においてもシフト日数を 14 日

[i] 例えば日経新聞の記事 [39] や朝日新聞の記事 [40] など。

とし「今日の移動削減率と 14 日後（2 週間後）の感染率の関係性」を分析していくこととする。

こうして改めて移動削減率と 2 週間後の感染率 β_t を散布図にしてグラフ化すると興味深い関係性が浮かび上がってくる。以下の図 4.4 は横軸に移動削減率、縦軸に 2 週間後の感染率 β_t を取り、時系列順に各点を彩色したものである。つまり、この図におけるデータの点は時系列順に青 → 水色 → 黄緑 → 黄色 → オレンジ → 赤 → 茶色の順に発生している。

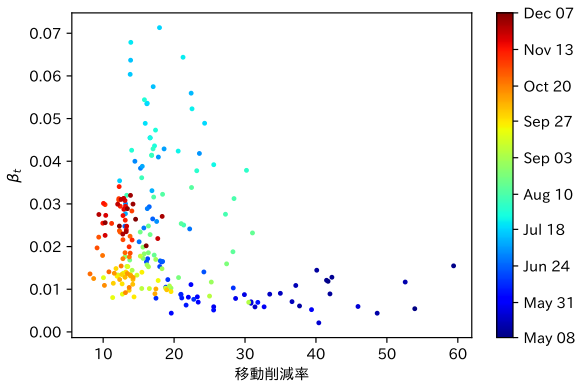


図 4.4　移動削減率と 2 週間後の感染率 β_t の散布図

図 4.4 を見ると、当初高移動削減率・低感染率で推移していた点（青色の点）あたりが、8 月に入り低移動削減率・高感染率（水色の点）に推移していった。そして 9 月に入ると低移動削減率を維持したまま感染率だけが減少し（黄色とオレンジ色の点）、最近ではまた上昇に転じている。そして、散布図を見ると移動削減率と 2 週間後の感染率は右下がりの傾向にあるらしいことがわかる。

4.2　線形回帰の結果と詳細な考察

4.2.1　回帰結果とシミュレーション

本節でも、前節で扱った「移動削減率と 2 週間後の感染率」の線形回帰から出発する。以下に、より詳細な結果を表示する。

表 4.1　移動削減率と 2 週間後感染率の回帰結果

パラメータ	推定値 （標準誤差）
移動削減率	-0.0004 (0.000)
定数項	0.0296 (0.002)
N	205
R^2	0.071

線形回帰の結果を散布図と重ね合わせたのが以下の図 4.5 である。図 4.5 の回帰直線上における点は、回帰直線上を移動削減率について 5% おきに取ったもので、10% から 55% までの 10 点ある。ここで、それぞれの点における感染率を再度 SIR モデルに組み込んで感染者の今後の推移をシミュレーションさせてみたのが以下の図 4.6 である[ii]。ここでは左の図 4.5 のそれぞれの点の色が、右の図 4.6 のグラフの色に対応している。

ここからわかることは、データが回帰直線に従うならば、移動削減率を上げれば 2 週間後の感染率 β_t が減少し、その結果 SIR モデルによる感染者の増加もより緩やかになるということである。例えば移動削減率が 10% だった場合、ピーク時の感染者が 3,000 万人近くまであるのに対し、移動削減率を 40% まで減らすと感染者数は最高でも 500 万人に満たない結果となる。

[ii] SIR モデルのもう一つのパラメータである回復率については $\gamma = 0.01$ で固定している。

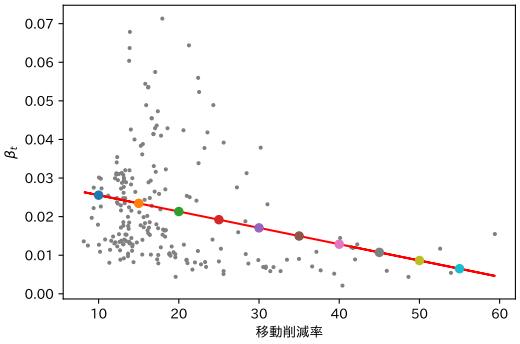


図 4.5　移動削減率と 2 週間後の感染率 β_t の回帰結果

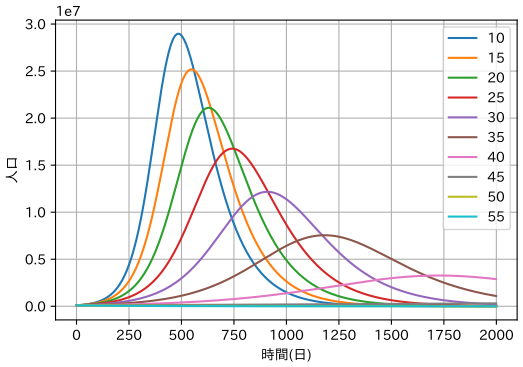


図 4.6　回帰直線上の点における SIR モデルのシミュレーション

また回帰直線上の点だけでなく、実際のデータの点も感染率 β_t や回復率 γ_t を SIR モデルに組み込むことでシミュレーションを行うことが可能である。そこで次ページの図 4.7 は実際のデータ点を移動削減率について 5% おきにグループ化し、それぞれの感染率 β_t と回復率 γ_t を SIR モデルに基づいてシミュレーションしたグラフである。例えば図の左上は、移動削減率（inv_mobi）が 5% 以上 10% 未満であるようなデータ点（7 つ）のそれぞれの感染率 β_t と回復率 γ_t をシミュレーションし、7 本のグラフを同じ図に描画したものである。

この図 4.7 を見ると、移動削減率が 10% 以上 15% 未満及び 15% 以上 20% 未満のグループは該当するデータの個数が多いため、図の線の本数も多くなっている。そのうちほとんどのグラフはピークが 2,000 万人以上となっており、中にはピーク時の感染者数が 4,000 万人以上となるようなグラフも存在する。移動削減率が 25% 以上となったあたりからグラフの本数が減少し、ピーク時の感染者数が減少したなだらかな曲線になっている。そして移動削減率が 45% 以上になるとほとんど感染者の増加は見受けられなくなる。より詳細な分析は次項で扱うこととする。

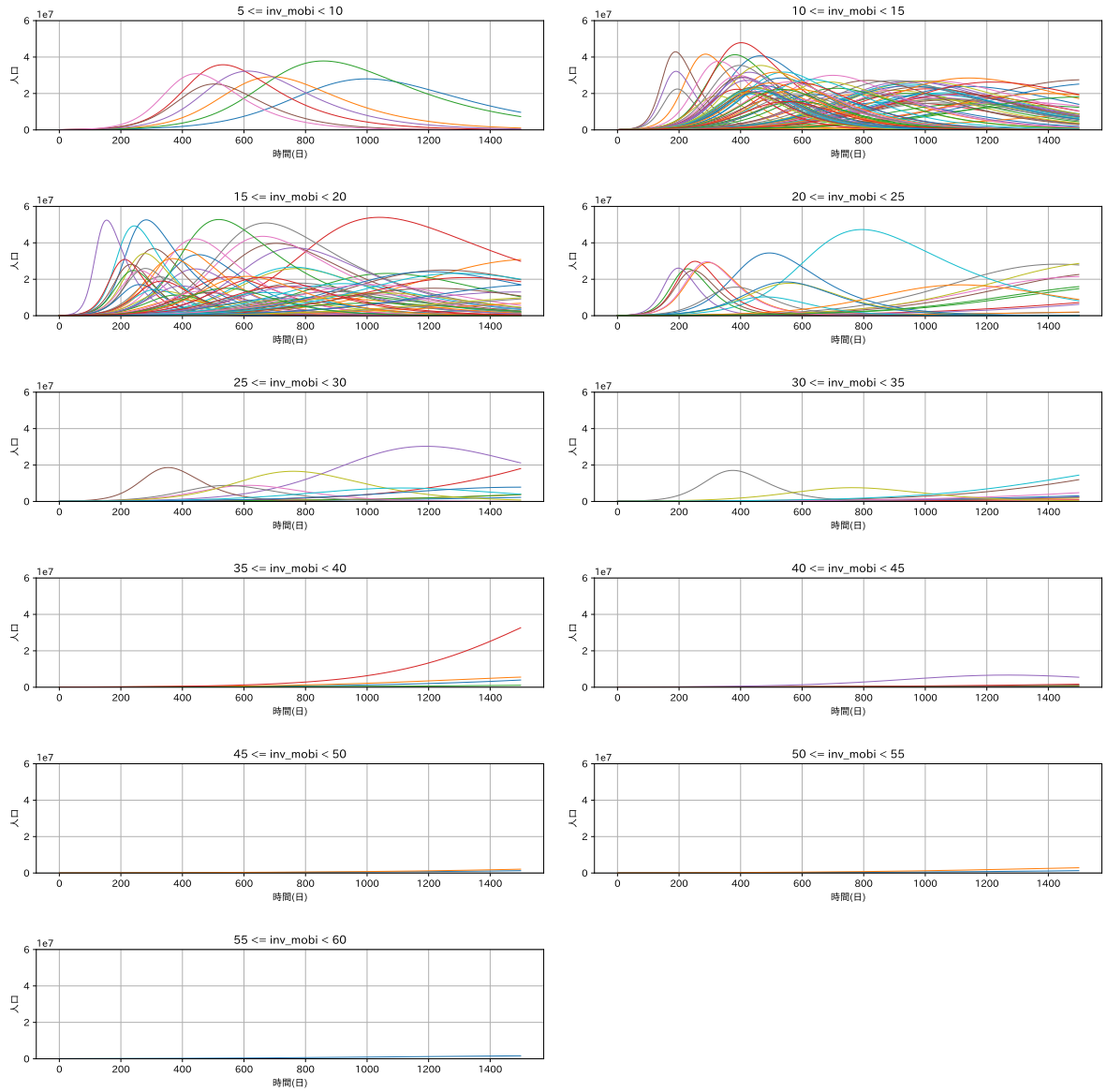


図 4.7　実際のデータ点における SIR モデルのシミュレーション

4.2.2　感染率と医療の逼迫度の関係

前項の最後では、感染率を SIR モデルに組み込むによって感染者数の推移をシミュレーションでき、感染率が高ければ高いほど感染者の増加も急なものになった。COVID-19 については、経済活動を完全に停止し人々が一切の外出をしないかワクチンが開発されない限り、早かれ遅かれ感染拡大は避けられないものである。そこで代わりに焦点となるのは、いかに感染の増加の割合を減らし COVID-19 が重症化した患者の数が医療機関の

キャパシティを逼迫することを避けるかということである。そこで本項では、独自に医療が逼迫するような閾値を設定し、それにより医療の逼迫度を示す度合いを定義する。最後に、改めて前項の移動削減率と 2 週間後の感染率の関係性から医療の逼迫度を計算し、医療を逼迫させないような移動削減率の程度を考察する。

まずは、医療を逼迫させるような感染者数の閾値を考察する。COVID-19 が医療の現場を逼迫させる要因は、COVID-19 の対応に医療従事者があてがわれ人員が不足することや、COVID-19 の罹患者の病床を確保するためにその他の患者の病床が不足することなどが考えられるが、今回は COVID-19 の人工呼吸器が必要となるような重症患者数に対し、人工呼吸器の数が不足することを医療の逼迫だと捉える。つまり、COVID-19 の感染者のうち、一定の割合で重症患者が発生すると仮定し、重症患者全員に人工呼吸器を装着すると仮定したとき、その数が日本全国の人工呼吸器の数を超過したときに医療が逼迫すると定義し、超過の程度が大きければ大きいほど医療をより強く圧迫するとする。

日本呼吸療法学会が 2020 年 5 月 14 日に発表した「国内の病院における人工呼吸器等の取扱台数推計値」という調査 [41] では、日本国内の病院における人工呼吸器の取扱台数は 45,293 台であった。また、厚生労働省が 2020 年 11 月 27 日に公表した情報 [42] によると、COVID-19 と診断された人のうち、重症化する割合は約 1.62% であった。従って、この数字を用いて逆算すると医療を逼迫させるような感染者数の閾値は $45{,}293 \div 1.62 \times 100 \approx 2{,}800{,}000$ (0.28×10^7) 人とする。医療の逼迫の度合いは、この閾値を超えた人数を日毎に足し合わせていったものと定義する。

次ページの図 4.8 は、前ページの図 4.7 に赤線で医療を逼迫する感染者の閾値の線を引き、各データのグラフが閾値を超えた部分を赤く塗ったものである。この赤く塗られた面積が、前段落で定義した医療の逼迫の度合いに相当する。また、1 年以内に COVID-19 に対するワクチンが開発され、医療の逼迫が完全になくなるという希望的観測をもって、面積を集計する期間は 1 年間とする。

図 4.8 を見てみると、移動削減率（inv_mobi）が 10% 以上 25% 未満であるような 3 つの図においては、多くのグラフが閾値を超えており、面積の色が重なり濃い赤色となっている。移動削減率（inv_mobi）が 25% 以上 35% 未満であるような 2 つの図においては、閾値を超えるようなグラフが 1 本か 2 本しかなく、移動削減率（inv_mobi）が 35% 以上だと閾値を超えるようなグラフは一切現れない。

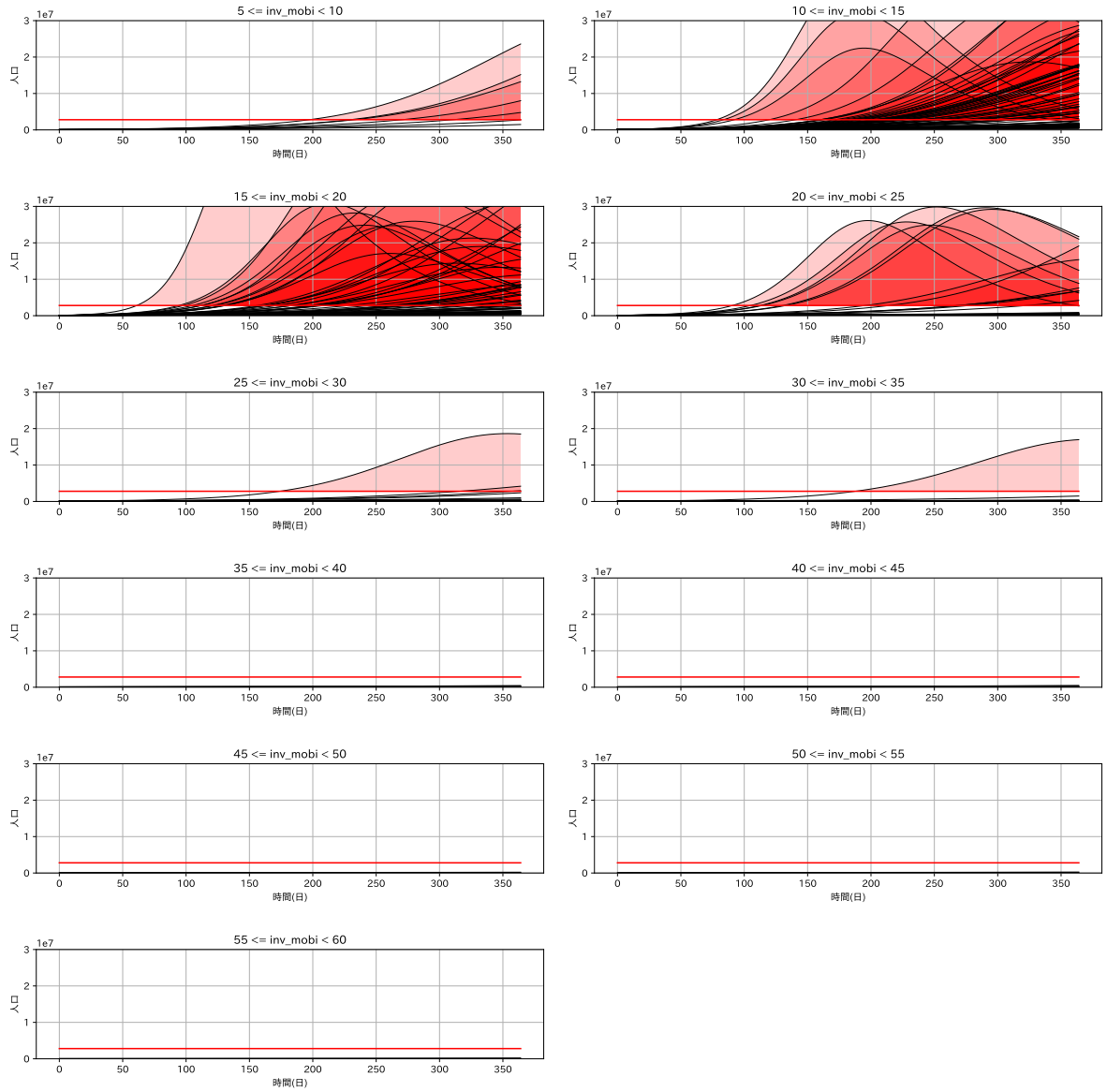


図 4.8　実際のデータ点における今後の医療の逼迫の程度

最後に、5 %おきにグループ分けした移動削減率のグループそれぞれの閾値を超過した合計面積を、データの個数で割ることによって、平均してどのくらい医療を逼迫させるかという指標を作ることができ、これを移動削減率に対する平均医療逼迫度を呼ぶことにする。例えばあるグループの平均医療逼迫度が 3,650,000 であるなら、医療を逼迫させる閾値（280 万人）を平均して 1 日あたり 10,000 人超過しているということになる。以下に、5 %おきにグループ分けした移動削減率のグループそれぞれの平均医療逼迫度を示す。

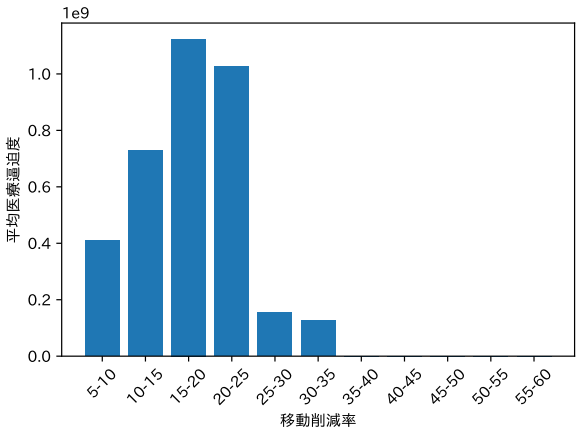


図 4.9　移動削減率と平均医療逼迫度の関係

この図 4.9 を見ると、移動削減率 25% 以上のグループは 25% 未満のグループと比較して平均医療逼迫度が大きく減少しており、移動削減率が 35% を超えると平均医療逼迫度はゼロとなる。以上の議論から、COVID-19 の感染者が医療を逼迫させないような理想の移動削減率は 35% 以上であることが結論付けられる。

4.3　都道府県別データへの拡張

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras

nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

第 5 章

おわりに

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum

ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa. [15]

5.1 今後の研究課題

外出自粛→感染率はここで示された。外出自粛（経済活動の抑制）→経済的な影響はすでにたくさんの論文が出ている。

→この 2 つを組み合わせれば、より詳細に感染率⇔経済のトレードオフが検討できる。

5.2 謝辞

I would like to thank my arms for always being by my side,
my legs for always supporting me,
and my fingers because I can always count on them.
Anonymous

本論文の執筆にあたって、実に多くの方のご支援をいただいた。こういう謝辞ではどこまで感謝すればいいのかわからないものだが、冗長でない程度に感謝の意をここに記したい。

修士ゼミナールの指導教官である竹内幹准教授には、終始適切な助言を賜り、丁寧かつ熱心なご指導ご鞭撻をいただいた。そもそもプログラミング言語である Python は、修士ゼミナールで竹内准教授とともに学習していたことであり、こういった形で修士ゼミナールの成果を竹内准教授にお見せできることを非常に嬉しく思う。竹内准教授がゼミナール内で発する「いいね！」や「グッド！」といった言葉のおかげで、研究のモチベーションが大きく上がったのは言うまでもない。また、ゼミナールの時間に質問や助言をくれた同期や後輩の諸君にも感謝したい。

2019 年 10 月から仕事の関係で日本に在住している両親には精神面（と温かいお風呂と美味しい食事と清潔な寝床）において多くの励ましと協力を賜った。また、在住している寮の友人たちや大学の友人たちについても、修士論文が忙しいといっても聞く耳を持たず、執拗に遊びに誘うことによって著者の危機感を煽ってくれたという意味で特に感謝の意を示したい。できればもう少し落ち着いてるときに誘ってほしいものである。

本論文は、2020 年末という COVID-19 がいまだ猛威を振るっている時期に執筆されたものである。この謝辞を書いている時点で、まだ COVID-19 がいつ収束するかの目処も全く立っていない。願わくば、数年経ったあとこの論文を読み返したときに「そういえばそんなこともあったな」と思い出すくらいには、将来の世界が明るいものでありますように。

2020 年 12 月 31 日

大晦日で誰もいない一橋大学マーキュリータワーの研究室にて

参考文献

[1] World Health Organization. Naming the coronavirus disease (COVID-19) and the virus that causes it. https://www.who.int/emergencies/diseases/novel-coronavirus-2019/technical-guidance/naming-the-coronavirus-disease-(covid-2019)-and-the-virus-that-causes-it. Accessed: 2020-12-31.

[2] WHO Kobe. コロナウイルス病 2019（COVID-19）に関する WHO-中国合同ミッション報告書. 2020. Accessed: 2020-12-31.

[3] G. Apolone, E. Montomoli, A. Manenti, M. Boeri, F. Sabia, I. Hyseni, L. Mazzini, D. Martinuzzi, L. Cantone, G. Milanese, S. Sestini, P. Suatoni, A. Marchianó, V. Bollati, G. Sozzi, and U. Pastorino. Unexpected detection of SARS-CoV-2 antibodies in the prepandemic period in Italy. *Tumori*, p. 300891620974755, 2020.

[4] 日本放送協会. 新型コロナウイルス特設サイト 時系列ニュース. https://www3.nhk.or.jp/news/special/coronavirus/chronology/. Accessed: 2020-12-31.

[5] World Health Organization. Listings of WHO’s response to COVID-19. https://www.who.int/news/item/29-06-2020-covidtimeline. Accessed: 2020-12-31.

[6] 一般財団法人アジア・パシフィック・イニシアティブ. 新型コロナ対応・民間臨時調査会 調査・検証報告書. ディスカヴァー・トゥエンティワン, 2020.

[7] Na Zhu, Dingyu Zhang, Wenling Wang, Xingwang Li, Bo Yang, Jingdong Song, Xiang Zhao, Baoying Huang, Weifeng Shi, Roujian Lu, Peihua Niu, Faxian Zhan, Xuejun Ma, Dayan Wang, Wenbo Xu, Guizhen Wu, George F. Gao, and Wenjie Tan. A novel coronavirus from patients with pneumonia in China, 2019.

New England Journal of Medicine, Vol. 382, No. 8, pp. 727–733, 2020. PMID: 31978945.

[8] Fan Wu, Su Zhao, Bin Yu, Yan-Mei Chen, Wen Wang, Zhi-Gang Song, Yi Hu, Zhao-Wu Tao, Jun-Hua Tian, Yuan-Yuan Pei, Ming-Li Yuan, Yu-Ling Zhang, Fa-Hui Dai, Yi Liu, Qi-Min Wang, Jiao-Jiao Zheng, Lin Xu, Edward C. Holmes, and Yong-Zhen Zhang. A new coronavirus associated with human respiratory disease in China. *Nature*, Vol. 579, No. 7798, pp. 265–269, 2020.

[9] 中村真司. Folding@home が TOP 500 の全スパコンを超える 2.4EFLOPS に到達. `https://pc.watch.impress.co.jp/docs/news/1246939.html`. Accessed: 2020-12-31.

[10] Maxwell I. Zimmerman, Justin R. Porter, Michael D. Ward, Sukrit Singh, Neha Vithani, Artur Meller, Upasana L. Mallimadugula, Catherine E. Kuhn, Jonathan H. Borowsky, Rafal P. Wiewiora, Matthew F. D. Hurley, Aoife M Harbison, Carl A Fogarty, Joseph E. Coffland, Elisa Fadda, Vincent A. Voelz, John D. Chodera, and Gregory R. Bowman. SARS-CoV-2 simulations go exascale to capture spike opening and reveal cryptic pockets across the proteome. *bioRxiv*, 2020.

[11] Qun Li, Xuhua Guan, Peng Wu, Xiaoye Wang, Lei Zhou, Yeqing Tong, Ruiqi Ren, Kathy S.M. Leung, Eric H.Y. Lau, Jessica Y. Wong, Xuesen Xing, Nijuan Xiang, Yang Wu, Chao Li, Qi Chen, Dan Li, Tian Liu, Jing Zhao, Man Liu, Wenxiao Tu, Chuding Chen, Lianmei Jin, Rui Yang, Qi Wang, Suhua Zhou, Rui Wang, Hui Liu, Yinbo Luo, Yuan Liu, Ge Shao, Huan Li, Zhongfa Tao, Yang Yang, Zhiqiang Deng, Boxi Liu, Zhitao Ma, Yanping Zhang, Guoqing Shi, Tommy T.Y. Lam, Joseph T. Wu, George F. Gao, Benjamin J. Cowling, Bo Yang, Gabriel M. Leung, and Zijian Feng. Early transmission dynamics in Wuhan, China, of novel coronavirus–infected pneumonia. *New England Journal of Medicine*, Vol. 382, No. 13, pp. 1199–1207, 2020. PMID: 31995857.

[12] Steven Sanche, Yen Ting Lin, Chonggang Xu, Ethan Romero-Severson, Nick Hengartner, and Ruian Ke. The novel coronavirus, 2019-nCoV, is highly contagious and more infectious than initially estimated. *medRxiv*, 2020.

[13] Julien Riou and Christian L. Althaus. Pattern of early human-to-human transmission of Wuhan 2019 novel coronavirus (2019-nCoV), December 2019 to January 2020. *Eurosurveillance*, Vol. 25, No. 4, 2020.

[14] Md. Arif Billah, Md. Mamun Miah, and Md. Nuruzzaman Khan. Reproductive number of coronavirus: A systematic review and meta-analysis based on global level evidence. *PLOS ONE*, Vol. 15, No. 11, pp. 1–17, 2020.

[15] Daron Acemoglu, Victor Chernozhukov, Iván Werning, and Michael D Whinston. Optimal targeted lockdowns in a multi-group SIR model. Working Paper 27102, National Bureau of Economic Research, 2020.

[16] Andrew Glover, Jonathan Heathcote, Dirk Krueger, and José-Víctor Ríos-Rull. Health versus wealth: On the distributional effects of controlling a pandemic. Working Paper 27046, National Bureau of Economic Research, 2020.

[17] Martin S Eichenbaum, Sergio Rebelo, and Mathias Trabandt. The macroeconomics of epidemics. Working Paper 26882, National Bureau of Economic Research, 2020.

[18] Victor Chernozhukov, Hiroyuki Kasahara, and Paul Schrimpf. Causal impact of masks, policies, behavior on early COVID-19 pandemic in the U.S. *Journal of Econometrics*, Vol. 220, No. 1, pp. 23 – 62, 2021. Themed Issue: Pandemic Econometrics / Covid Pandemics.

[19] Dirk Krueger, Harald Uhlig, and Taojun Xie. Macroeconomic dynamics and reallocation in an epidemic: Evaluating the "Swedish solution". Working Paper 27047, National Bureau of Economic Research, 2020.

[20] Tsutomu Watanabe and Tomoyoshi Yabu. Japan's voluntary lockdown. *Covid Economics, Vetted and Real-Time Papers*, Vol. 46, No. 1, pp. 1–31, 2020.

[21] COVID-19 challenge. `https://signate.jp/covid-19-challenge/`. Accessed: 2020-12-31.

[22] Christopher Avery, William Bossert, Adam Clark, Glenn Ellison, and Sara Fisher Ellison. Policy implications of models of the spread of coronavirus: Perspectives and opportunities for economists. *Covid Economics, Vetted and Real-Time Papers*, Vol. 12, No. 2, pp. 21–68, 2020.

[23] Amy Hurford. Mechanistic models: what is the value of understanding? `https://theartofmodelling.wordpress.com/2012/02/19/mechanistic-models-what-is-the-value-of-understanding/`, 2012. Accessed: 2020-12-31.

[24] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A,*

Containing Papers of a Mathematical and Physical Character, Vol. 115, No. 772, pp. 700–721, 1927.

[25] 新型コロナウイルス感染症対策推進室. 新型コロナウイルス感染症対策. `https://corona.go.jp/dashboard/`. Accessed: 2020-12-31.

[26] 厚生労働省. オープンデータ. `https://www.mhlw.go.jp/stf/covid-19/open-data.html`. Accessed: 2020-12-31.

[27] Glenn Ellison. Implications of heterogeneous SIR models for analyses of COVID-19. Working Paper 27373, National Bureau of Economic Research, 2020.

[28] Jesús Fernández-Villaverde and Charles I Jones. Estimating and simulating a SIRD model of COVID-19 for many countries, states, and cities. Working Paper 27128, National Bureau of Economic Research, 2020.

[29] Wladimir Lyra, José-Dias do Nascimento, Jaber Belkhiria, Leandro de Almeida, Pedro Paulo M. Chrispim, and Ion de Andrade. COVID-19 pandemics modeling with SEIR(+CAQH), social distancing, and age stratification. the effect of vertical confinement and release in brazil. *medRxiv*, 2020.

[30] Fred Brauer, Pauline van Den Driessche, and Jianhong Wu. *Mathematical epidemiology*. Springer, Berlin, 2008.

[31] James H. Stock. Data gaps and the policy response to the novel coronavirus. *Covid Economics, Vetted and Real-Time Papers*, Vol. 3, No. 1, pp. 1–11, 2020.

[32] Stefanella Boatto, Catherine Bonnet, Bernard Cazelles, and Frédéric Mazenc. SIR model with time dependent infectivity parameter : approximating the epidemic attractor and the importance of the initial phase. working paper or preprint, 2018.

[33] Mehrdad Kiamari, Gowri Ramachandran, Quynh Nguyen, Eva Pereira, Jeanne Holm, and Bhaskar Krishnamachari. COVID-19 risk estimation using a time-varying SIR-model, 2020.

[34] Y. C. Chen, P. E. Lu, C. S. Chang, and T. H. Liu. A time-dependent SIR model for COVID-19 with undetectable infected persons. *IEEE Transactions on Network Science and Engineering*, 2020.

[35] Luís M. A. Bettencourt and Ruy M. Ribeiro. Real time bayesian estimation of the epidemic potential of emerging infectious diseases. *PLOS ONE*, Vol. 3,

No. 5, pp. 1–9, 2008.

[36] Hiroshi Nishiura and Gerardo Chowell. The effective reproduction number as a prelude to statistical estimation of time-dependent epidemic trends. *Mathematical and Statistical Estimation Approaches in Epidemiology*, pp. 103–121, 2009.

[37] NTT ドコモ. モバイル空間統計 新型コロナウイルス感染症対策特設サイト. https://mobaku.jp/covid-19/download/%E5%A2%97%E6%B8%9B%E7%8E%87%E4%B8%80%E8%A6%A7.csv. Accessed: 2020-12-31.

[38] Google LLC. COVID-19 community mobility reports. https://www.google.com/covid19/mobility/index.html. Accessed: 2020-12-31.

[39] 日本経済新聞. 外出自粛、2 週間後に効果？ 「市中感染」減少. https://www.nikkei.com/article/DGXMZO58592950Y0A420C2AC8Z00. Accessed: 2020-12-31.

[40] 朝日新聞. 【詳報】首相が緊急事態宣言「2 週間後に感染者を減少」. https://www.asahi.com/articles/ASN467JFGN46UTFK01R.html. Accessed: 2020-12-31.

[41] 日本呼吸療法医学会. 国内の病院における人工呼吸器等の取扱台数推計値. https://www.jsicm.org/news/upload/jsicm_info_ventilator_200514.pdf. Accessed: 2020-12-31.

[42] 厚生労働省. （2020 年 11 月時点）新型コロナウイルス感染症の“いま”についての 10 の知識. https://www.mhlw.go.jp/content/000699304.pdf. Accessed: 2020-12-31.

[43] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 1999.

付録 A

Python によるプログラムコード

> Any fool can write code that a computer can understand. Good programmers write code that humans can understand.
> ——どんな馬鹿でもコンピュータが理解できるコードは書ける。しかし人間が理解できるコードをかけるのは優秀なプログラマだけである。
>
> Martin Fowler [43]

本論文の分析においては、プログラミング言語のひとつである Python を使用した。残念ながら筆者は Python を使用しはじめてから 1 年も経っていないので、コンピュータが理解できるコードを書くのが精いっぱいで、ここに記されているコードはお世辞にも人間が理解できるコードとは言い難い。しかし、数多あるプログラミングの入門書やインターネットの存在に助けられ、なんとか一連の分析を終えることができた。ここでは、コードの内容もかいつまんで説明しながら論文内の分析や図表の出力を与えるコードを掲載する。

また、一連のコードを載せた Jupyter Notebook も著者の Google Drive[i]にアップロードしているので積極的に活用されたい。

[i] https://うんたらかんたら

A.1　導入

Python でのプログラミングは Anaconda に同梱される Jupyter Notebook で行った。バージョンの情報は以下のとおりである。

余談だが、以下のうち `lmfit` や `statsmodels`、`japanize_matplotlib`[ii]などといったパッケージは Anaconda の初期パッケージに含まれていない。そのため、あらかじめ `conda install` や `pip install` コマンドなどでインストールする必要がある。

- In[1]:

```
import sys
import numpy as np
import pandas as pd
import scipy as sp
from scipy.integrate import odeint
import lmfit
from lmfit import minimize, Parameters, report_fit
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
import sklearn
from sklearn.metrics import mean_squared_error
import matplotlib
import matplotlib.pyplot as plt
import japanize_matplotlib
import datetime
import warnings
np.set_printoptions(threshold=0)
warnings.filterwarnings('ignore')

print('Python version: {}'.format(sys.version))
print('NumPy version: {}'.format(np.__version__))
print('pandas version: {}'.format(pd.__version__))
print('SciPy version: {}'.format(sp.__version__))
print('LMFIT version: {}'.format(lmfit.__version__))
print('statsmodels version: {}'.format(sm.__version__))
print('sklearn version: {}'.format(sklearn.__version__))
print('matplotlib version: {}'.format(matplotlib.__version__))
```

- Out[1]:

```
Python version: 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64
    )]
NumPy version: 1.19.1
pandas version: 1.1.1
```

[ii] `matplotlib` に日本語の名前などを入力した際文字化けするのを防ぐ、個人が開発したパッケージである。すべて英語でグラフを作成する際にはこのパッケージは必要ない。

```
SciPy version: 1.5.0
LMFIT version: 1.0.1
statsmodels version: 0.12.0
sklearn version: 0.23.2
matplotlib version: 3.3.1
```

A.2 感染者データの可視化

ここでは、2 章で掲載した全世界及び日本の感染者や死者のデータをグラフにして出力する。全世界のデータについては “Our World in Data” より、日本国内のデータについては内閣官房新型コロナウイルス感染症対策推進室より取得している。

- In[2]:

```
world_data = pd.read_csv('https://covid.ourworldindata.org/data/owid-covid-data.
    csv',parse_dates=True)
world_data = world_data[['date','continent','location','total_cases','new_cases
    ','total_deaths','new_deaths']].set_index('date')

continents = ['Asia', 'Europe', 'Africa', 'North America', 'South America', '
    Oceania']
date_range = pd.date_range(world_data.sort_index().index[0],
                           world_data.sort_index().index[-1],
                           freq='d')

res_t_cases = pd.DataFrame(index=continents, columns=date_range)
res_t_deaths = pd.DataFrame(index=continents, columns=date_range)

for i in continents:
    subset = world_data[world_data['continent'] == i]
    date = pd.date_range(subset.sort_index().index[0],subset.sort_index().index
        [-1],freq='d')
    for j in date:
        datetime = j.strftime('%Y-%m-%d')
        subsubset = subset[subset.index.str.contains(datetime)]
        res_t_cases.at[i,j] = subsubset['total_cases'].sum()
        res_t_deaths.at[i,j] = subsubset['total_deaths'].sum()

res_n_cases = pd.DataFrame(index=date_range, columns=['new_cases'])
res_n_deaths = pd.DataFrame(index=date_range, columns=['new_deaths'])

for k in date_range:
    datetime = k.strftime('%Y-%m-%d')
    subset = world_data[world_data.index.str.contains(datetime)]
    res_n_cases.at[k,'new_cases'] = subset['new_cases'].sum()
    res_n_deaths.at[k,'new_deaths'] = subset['new_deaths'].sum()
```

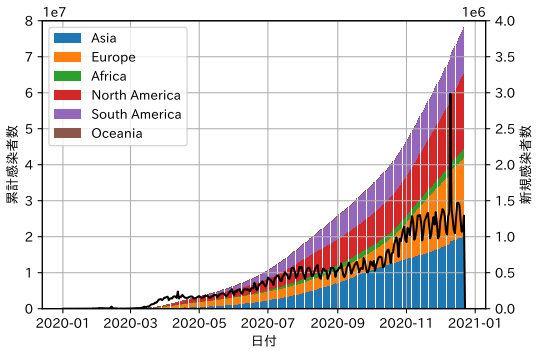
まずは全世界のデータを図に出力する。累計感染者数については大陸別の積み上げ棒グ

ラフ、新規感染者は折れ線グラフとし、1 つの図に落とし込む。この図は本文 7 ページの図 2.1 に該当する。

- In[3]:

```
fig, ax1 = plt.subplots(1,1)
ax2 = ax1.twinx()
for i in range(len(res_t_cases)):
    ax1.bar(res_t_cases.columns, res_t_cases.iloc[i], bottom=res_t_cases.iloc[:i
        ].sum())
ax1.set(xlabel='日付', ylabel='累計感染者数')
ax1.legend(res_t_cases.index, loc=2)
ax2.plot(res_n_cases['new_cases'],linestyle='solid',color='k')
ax2.set(ylabel='新規感染者数')
ax1.set_ylim(0,8e+7)
ax2.set_ylim(0,4e+6)
ax1.grid()
fig.savefig('figure/globalcases.pdf', bbox_inches='tight')
fig.show()
```

- Out[3]:



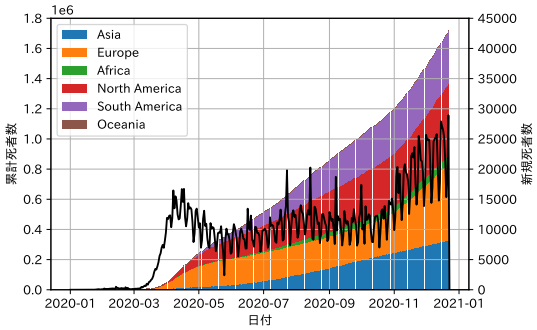
全世界の累計死者数及び新規死者数についても同様に集計する。この図は本文 7 ページの図 2.2 に該当する。

- In[4]:

```
fig, ax1 = plt.subplots(1,1)
ax2 = ax1.twinx()
for i in range(len(res_t_deaths)):
    ax1.bar(res_t_deaths.columns, res_t_deaths.iloc[i], bottom=res_t_deaths.iloc
        [:i].sum())
ax1.set(xlabel='日付', ylabel='累計死者数')
ax1.legend(res_t_deaths.index, loc=2)
ax2.plot(res_n_deaths['new_deaths'],linestyle='solid',color='k')
ax2.set(ylabel='新規死者数')
ax1.set_ylim(0,1.8e+6)
ax2.set_ylim(0,45000)
ax1.grid()
```

```
fig.savefig('figure/globaldeaths.pdf', bbox_inches='tight')
fig.show()
```

- Out[4]:

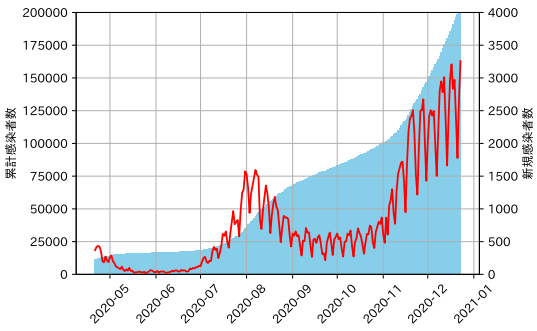


最後に日本国内のデータをグラフにする。この図は本文 9 ページの図 2.3 に該当する。

- In[5]:

```
japan_data = pd.read_json('https://data.corona.go.jp/converted-json/covid19japan
    -npatients.json', dtype='float')
japan_data = japan_data.set_index('date')

fig, ax1 = plt.subplots(1,1)
ax2 = ax1.twinx()
ax1.bar(japan_data.index, japan_data['npatients'], color='skyblue')
ax1.set_xticklabels('日付', rotation=45)
ax1.set_ylabel('累計感染者数')
ax2.plot(japan_data['adpatients'],linestyle='solid',color='r')
ax2.set(ylabel='新規感染者数')
ax1.set_ylim(0,200000)
ax2.set_ylim(0,4000)
ax1.grid()
fig.savefig('figure/japancases.pdf', bbox_inches='tight')
fig.show()
```

- Out[5]:



A.3　SIR モデル

A.3.1　SIR モデルのシミュレーション

まずは、SIR モデルの常微分方程式を関数として定義する。

- In[6]:

```
def SIR_model(z, t, beta, gamma):
    S, I, R = z
    N = S + I + R
    dSdt = -beta*S*I/N
    dIdt = beta*S*I/N - gamma*I
    dRdt = gamma*I
    return [dSdt, dIdt, dRdt]
```

初期値を入力した常微分の数値解析には、SciPy パッケージに含まれる odeint モジュールを用いる[iii]。第 1 引数に常微分方程式、第 2 引数に初期値、第 3 引数に時点列、そして第 4 引数にその他必要なパラメータの値を渡すことによって計算される。

- In[7]:

```
tspan = np.arange(0, 100)
initI, initR, initN = 100, 0, 120000000
initS = initN - (initI + initR)
beta, gamma = 0.5, 0.1

result = odeint(SIR_model, [initS, initI, initR], tspan, args=(beta, gamma))
result[:5]
```

- Out[7]:

```
array([[1.19999900 e+ 08, 1.00000000 e+ 02, 0.00000000 e+ 00],
       [1.19999839 e+ 08, 1.49182390 e+ 02, 1.22956142 e+ 01],
       [1.19999747 e+ 08, 2.22553785 e+ 02, 3.06385026 e+ 01],
       ...,
       [8.49283849 e+ 05, 3.30251480 e+ 05, 1.18820465 e+ 08],
       [8.48170520 e+ 05, 2.99882506 e+ 05, 1.18851947 e+ 08],
       [8.47160837 e+ 05, 2.72304971 e+ 05, 1.18880534 e+ 08]])
```

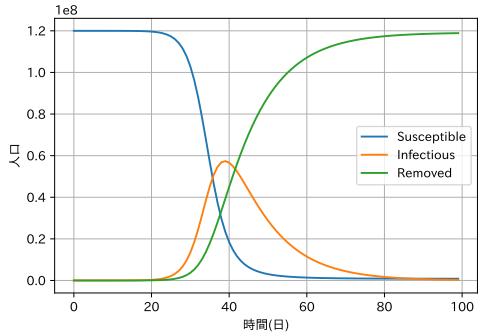
16 ページの図 3.1 はこの結果をグラフにして出力したものである。

[iii] 最近ではこのモジュールの代わりに scipy.integrate.solve_ivp モジュールを使うべきとされている。

- In[8]:

```
plt.plot(tspan, result)
plt.legend(['Susceptible', 'Infectious', 'Removed'])
plt.xlabel('時間(日)')
plt.ylabel('人口')
plt.grid()
plt.savefig('figure/SIRsimu.pdf')
plt.show()
```

- Out[8]:



A.3.2 SIR モデルのパラメータ推定

まずは、実際の COVID-19 に関するデータをインターネットより取得し、データを成型する。

- In[9]:

```
patients = pd.read_json('https://data.corona.go.jp/converted-json/covid19japan-
    npatients.json', dtype='float')
recovery = pd.read_csv('https://www.mhlw.go.jp/content/recovery_total.csv',
                       index_col='日付' , dtype='float', parse_dates=True)
death = pd.read_csv('https://www.mhlw.go.jp/content/death_total.csv',
                    index_col='日付' , dtype='float', parse_dates=True)

cml_df = patients.set_index('date').drop(columns='adpatients')
                 .rename(columns={'npatients':'infected'})
cml_df['removed'] = recovery['退院、療養解除となった者']+death['死亡者数']
cml_df = cml_df['2020-05-08':]
cml_df = cml_df.dropna(how = 'any')
print(cml_df)
```

- Out[9]:

```
            infected   removed
date
```

```
2020-05-08   15462.0    8710.0
2020-05-09   15581.0    8889.0
2020-05-10   15630.0    9135.0
2020-05-11   15706.0    9546.0
2020-05-12   15854.0   10519.0
...              ...       ...
2020-12-08  164203.0  141543.0
2020-12-09  166928.0  143958.0
2020-12-10  169890.0  145992.0
2020-12-11  172638.0  148520.0
2020-12-12  175608.0  150301.0
```

次に、あるパラメータを初期値とした数値解析の解と実際の数値の差分を取るような誤差関数を定義する。つまり、パラメータの推定とはこの誤差関数を最小化するようなパラメータを選ぶことである。

- In[10]:

```
def SIR_solver(t, initial_conditions, params):
    initI, initR, initN = initial_conditions
    beta, gamma = params['beta'].value, params['gamma'].value
    initS = initN - (initI + initR)
    res = odeint(SIR_model, [initS, initI, initR], t, args=(beta, gamma))
    return res
```

- In[11]:

```
def error(params, initial_conditions, tspan, data):
    sol = SIR_solver(tspan, initial_conditions, params)
    return (sol[:, 1:3] - data).ravel()
```

インターネット上から取得したデータのうち、誤差の計算に使われる部分だけを取り出す。その他、数値解析に必要な初期値や時点列などを定義する。

- In[12]:

```
initN = 126500000
initI = cml_df.at[cml_df.index[0], 'infected']
initR = cml_df.at[cml_df.index[0], 'removed']
initial_conditions = [initI, initR, initN]

period = cml_df.index[-1]-cml_df.index[0]
days = period.days
tspan = np.arange(0, days+1)

data = cml_df[['infected','removed']].values
data
```

- Out[12]:

```
array([[ 15462.,   8710.],
       [ 15581.,   8889.],
       [ 15630.,   9135.],
       ...,
       [172638., 148520.],
       [175608., 150301.],
       [177960., 151975.]])
```

誤差関数の最小化には、lmfit パッケージの minimize モジュールを使うが、その際 Parameters モジュールでパラメータの初期値や値の取りうる範囲を指定する。最小化を行った後は report_fit モジュールで詳細な結果を確認できる。この結果の一部は 17 ページの表 3.1 に使用されている。

- In[13]:

```
params = Parameters()
params.add('beta', value=beta, min=0, max=10)
params.add('gamma', value=gamma, min=0, max=10)
params['beta'].value = 0.5
params['gamma'].value = 0.1
```

- In[14]:

```
result = minimize(error, params, args=(initial_conditions, tspan, data))
report_fit(result)
```

- Out[14]:

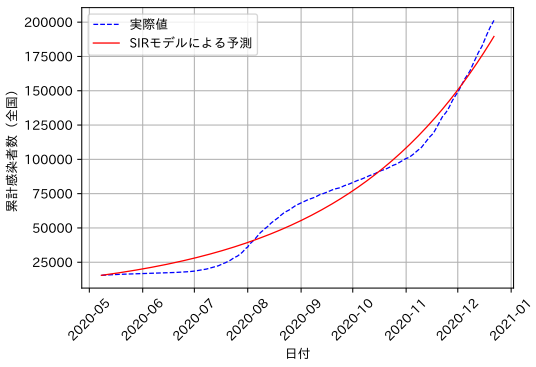
```
[[Fit Statistics]]
    fitting method     = leastsq
    function evals     = 28
    data points        = 440
    variables          = 2
    chi-square         = 2.3648e+10
    reduced chi-square = 53991885.5
    Akaike info crit   = 7835.90694
    Bayesian info crit = 7844.08049
[[Variables]]
    beta:   0.02135509 +/- 8.3942e-05 (0.39%) (init = 0.5)
    gamma:  0.01041938 +/- 9.4597e-05 (0.91%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(beta, gamma) =  0.928
```

最後に、得られたパラメータによる SIR モデルの推定を行い、結果を出力する。この結果は 17 ページの図 3.2 として使われている。

- In[15]:

```
SIR_pred = data + result.residual.reshape(data.shape)
SIR_pred = pd.DataFrame(SIR_pred)
SIR_pred['date'] = pd.date_range(start=cml_df.index[0],
                                 end=cml_df.index[-1], freq='D')
SIR_pred = SIR_pred.set_index('date')

plt.plot(cml_df['infected'], color='blue',  linestyle='dashed',
         linewidth = 1.0, label='実際値')
plt.plot(SIR_pred[0], color='red',  linestyle='solid',
         linewidth = 1.0, label='SIRモデルによる予測')
plt.xlabel('日付')
plt.ylabel('累計感染者数（全国）')
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.savefig('figure/SIRest.pdf', bbox_inches='tight')
plt.show()
```

- Out[15]:



A.4 SARIMA モデル

A.4.1 データの読込・成分分解

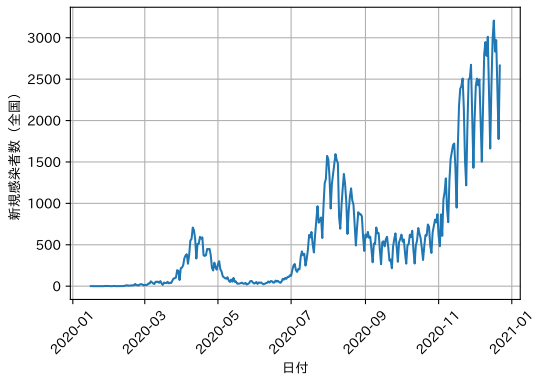
SARIMA モデルは累計感染者数ではなく新規感染者数を予測するので新たにデータを取得する。新規感染者の図は 19 ページの図 3.3 に使用されている。

- In[16]:

```
new_df = pd.read_csv('https://www.mhlw.go.jp/content/pcr_positive_daily.csv',
                     index_col='日付' , dtype='float', parse_dates=True)

plt.plot(new_df)
plt.xlabel('日付')
```

```
plt.ylabel('新規感染者数（全国）')
plt.grid()
plt.xticks(rotation=45)
plt.savefig('figure/newinfected.pdf', bbox_inches='tight')
plt.show()
```

- Out[16]:



次に、20 ページの図 3.4 にある成分分解のグラフを作る[iv]。

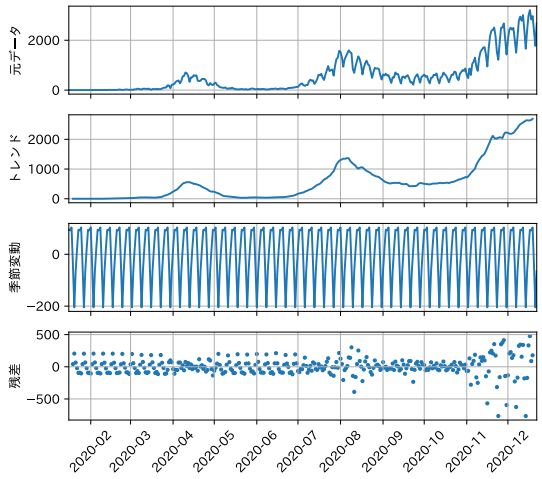
- In[17]:

```
decomposition = seasonal_decompose(new_df, period=7)

fig = plt.figure(figsize=(6, 5))

t = pd.date_range(new_df.index[0],new_df.index[-1])

ax1 = fig.add_subplot(4, 1, 1)
ax2 = fig.add_subplot(4, 1, 2)
ax3 = fig.add_subplot(4, 1, 3)

ax1.plot(t, new_df)
ax2.plot(t, decomposition.trend)
ax3.plot(t, decomposition.seasonal)

axs = plt.gcf().get_axes()
for ax in axs:
    ax.axes.get_xaxis().set_ticklabels([])

ax4 = fig.add_subplot(4, 1, 4)
ax4.scatter(t, decomposition.resid, s=5)

axs = plt.gcf().get_axes()
for ax in axs:
    ax.set_xlim(new_df.index[0],new_df.index[-1])
```

[iv] 本来は seasonal_decompose.plot モジュールを使えばわざわざ図を 1 つずつ出力する必要はないが、縮尺が固定され図が見づらかったので、著者が新たにプロットした。

```
    ax.grid()

ax1.set_ylabel('元データ')
ax2.set_ylabel('トレンド')
ax3.set_ylabel('季節変動')
ax4.set_ylabel('残差')

fig.tight_layout()
plt.xticks(rotation=45)
plt.savefig('figure/SeaDec.pdf', bbox_inches='tight')
plt.show()
```

- Out[17]:



A.4.2 SARIMA モデルのパラメータ推定

SARIMA モデルのパラメータ探索は総当たり法によって行われる。まずは各パラメータの探索の最大値を入力し、パターンの数を計算する。

- In[18]:

```
max_p = 3
max_q = 3
max_d = 1
max_sp = 1
max_sq = 1
max_sd = 1

pattern = max_p*(max_q + 1)*(max_d + 1)*(max_sp + 1)*(max_sq + 1)*(max_sd + 1)

modelSelection = pd.DataFrame(index=range(pattern), columns=["model", "aic"])
pattern
```

- Out[18]:

```
192
```

総当たりについては、ループ処理を行うことで 1 つずつ地道に計算していく。

- In[19]:

```
num = 0
for p in range(1, max_p + 1):
    for d in range(0, max_d + 1):
        for q in range(0, max_q + 1):
            for sp in range(0, max_sp + 1):
                for sd in range(0, max_sd + 1):
                    for sq in range(0, max_sq + 1):
                        sarima = sm.tsa.SARIMAX(new_df, order=(p,d,q),
                             seasonal_order=(sp,sd,sq,7), enforce_stationarity =
                              True, enforce_invertibility = True).fit(method='
                             bfgs', maxiter=300, disp=False)
                        print("order=(" + str(p) + ","+ str(d) + ","+ str(q) +
                             "), season=("+ str(sp) + ","+ str(sd) + "," + str(
                             sq) + "), AIC=" + str(sarima.aic))
                        modelSelection.iloc[num]["model"] = "order=(" + str(p) +
                              ","+ str(d) + ","+ str(q) + "), season=("+ str(sp)
                              + ","+ str(sd) + "," + str(sq) + ")"
                        modelSelection.iloc[num]["aic"] = sarima.aic
                        num = num + 1
```

- Out[19]:

```
order=(1,0,0), season=(0,0,0), AIC=4403.694589801746
order=(1,0,0), season=(0,0,1), AIC=4297.549573769679
order=(1,0,0), season=(0,1,0), AIC=4110.841897705888
     (中略)
order=(3,1,3), season=(1,0,1), AIC=4131.061265686723
order=(3,1,3), season=(1,1,0), AIC=4052.4768162772184
order=(3,1,3), season=(1,1,1), AIC=4030.867634973866
```

AIC が最小となるようなパラメータは以下のとおりである。

- In[20]:

```
modelSelection[modelSelection.aic == min(modelSelection.aic)]
```

- Out[20]:

```
                             model      aic
107 order=(2,1,1), season=(0,1,1)   4016.8
```

得られたパラメータを SARIMA モデルに入力し、回帰分析を行う。ここで得られた結果の一部は 21 ページの表 3.2 に使われている。

- In[21]:

```
bestSARIMA = sm.tsa.SARIMAX(new_df, order=(2,1,1), seasonal_order=(0,1,1,7),
                            enforce_stationarity = True,
                            enforce_invertibility = True).fit()
print(bestSARIMA.summary())
```

- Out[21]:

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                PCR 検査陽性者数(単日)   No. Observations:                  333
Model:          SARIMAX(2, 1, 1)x(0, 1, 1, 7)   Log Likelihood               -2003.402
Date:                          Mon, 14 Dec 2020   AIC                           4016.804
Time:                                  18:50:58   BIC                           4035.723
Sample:                              01-16-2020   HQIC                          4024.354
                                   - 12-13-2020
Covariance Type:                            opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.5402      0.081      6.656      0.000       0.381       0.699
ar.L2         -0.2341      0.042     -5.572      0.000      -0.316      -0.152
ma.L1         -0.6201      0.086     -7.238      0.000      -0.788      -0.452
ma.S.L7       -0.5901      0.030    -19.936      0.000      -0.648      -0.532
sigma2      1.311e+04    533.735     24.557      0.000    1.21e+04    1.42e+
     04
==============================================================================
Ljung-Box (L1) (Q):                   0.03   Jarque-Bera (JB):              1703.76
Prob(Q):                              0.86   Prob(JB):                         0.00
Heteroskedasticity (H):               8.47   Skew:                            -1.16
Prob(H) (two-sided):                  0.00   Kurtosis:                        13.97
==========================================================================================
```

SARIMA モデルの予測は、一般的な機械学習のモジュールと同じ predict モジュールで行うことができる。

- In[22]:

```
SARIMA_pred = bestSARIMA.predict('2020-02-01', '2021-01-31')
SARIMA_pred
```

- Out[22]:

```
2020-02-01       0.795466
2020-02-02       0.439540
2020-02-03      -0.052382
2020-02-04       1.970694
```

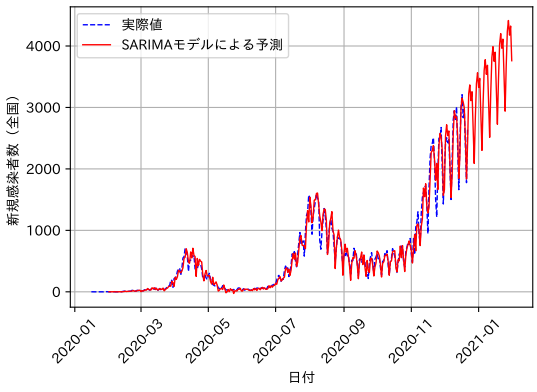
```
2020-02-05       0.923579
                 ...
2021-01-27    4345.239278
2021-01-28    4546.770967
2021-01-29    4468.610902
2021-01-30    4617.999649
2021-01-31    4079.957177
Freq: D, Name: predicted_mean, Length: 366, dtype: float64
```

最後に、実際の数値と SARIMA モデルによる予測値を重ね合わせたグラフを出力する。1 つ目は 22 の図 3.5 であり、2 つ目は同ページの図 3.6 である。

- In[23]:

```
plt.plot(new_df, color='blue',  linestyle='dashed', linewidth = 1.0,
         label='実際値')
plt.plot(SARIMA_pred, color='red', linewidth = 1.0,
         label='SARIMAモデルによる予測')
plt.xlabel('日付')
plt.ylabel('新規感染者数（全国）')
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.savefig('figure/SARIMAestNEW.pdf', bbox_inches='tight')
plt.show()
```

- Out[23]:

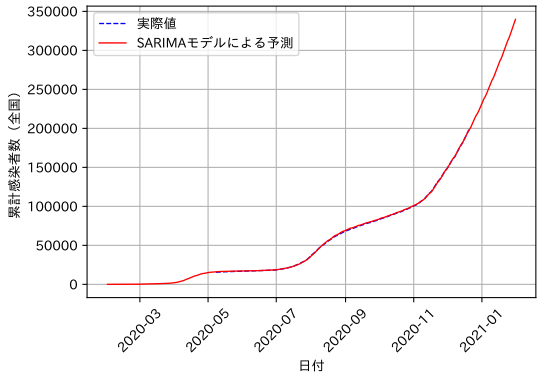


- In[24]:

```
SARIMA_cml = pd.DataFrame(SARIMA_pred)
SARIMA_cml['cmlsum'] = pd.DataFrame.cumsum(SARIMA_cml['predicted_mean'])

plt.plot(cml_df['infected'], color='blue',  linestyle='dashed', linewidth = 1.0,
         label='実際値')
plt.plot(SARIMA_cml['cmlsum'], color='red',  linestyle='solid', linewidth = 1.0,
         label='SARIMAモデルによる予測')
```

```
plt.xlabel('日付')
plt.ylabel('累計感染者数（全国）')
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.savefig('figure/SARIMAestCML.pdf', bbox_inches='tight')
plt.show()
```

- Out[24]:



A.5　SIR モデルと SARIMA モデルの比較

まずは学習期間となる 2020 年 5 月 8 日〜10 月 31 日までのデータセットを作り、前節で使ったコードを再度使い結果を出力していく。

- In[25]:

```
cml_df_oct = cml_df['2020-05-08':'2020-10-31']

initN = 126500000
initI = cml_df_oct.at[cml_df_oct.index[0], 'infected']
initR = cml_df_oct.at[cml_df_oct.index[0], 'removed']
initial_conditions = [initI, initR, initN]

period = cml_df_oct.index[-1]-cml_df_oct.index[0]
days = period.days
tspan = np.arange(0, days+1)

data = cml_df_oct[['infected','removed']].values

params = Parameters()
params.add('beta', value=beta, min=0, max=10)
params.add('gamma', value=gamma, min=0, max=10)
params['beta'].value = 0.1
params['gamma'].value = 0.01

result = minimize(error, params, args=(initial_conditions, tspan, data))
```

```
beta, gamma = result.params['beta'].value, result.params['gamma'].value
tspan = np.arange(0, days+31)
SIR_NOV_cml = odeint(SIR_model, [initS, initI, initR], tspan, args=(beta, gamma
    ))
SIR_NOV_cml = pd.DataFrame(SIR_NOV_cml)
SIR_NOV_cml = SIR_NOV_cml.iloc[SIR_NOV_cml.index[-30]:SIR_NOV_cml.index[-1]+1]
SIR_NOV_cml['Date'] = pd.date_range('2020-11-01','2020-11-30')
SIR_NOV_cml = SIR_NOV_cml.set_index('Date')
SIR_NOV_cml = SIR_NOV_cml[1]
SIR_NOV_cml
```

- Out[25]:

```
Date
2020-11-01    112840.692326
2020-11-02    114112.172166
2020-11-03    115397.922998
                  ...
2020-11-28    152676.660164
2020-11-29    154394.662801
2020-11-30    156131.895024
Name: 1, dtype: float64
```

- In[26]:

```
new_df_oct = new_df['2020-05-08':'2020-10-31']

num = 0
for p in range(1, max_p + 1):
    for d in range(0, max_d + 1):
        for q in range(0, max_q + 1):
            for sp in range(0, max_sp + 1):
                for sd in range(0, max_sd + 1):
                    for sq in range(0, max_sq + 1):
                        sarima = sm.tsa.SARIMAX(new_df_oct, order=(p,d,q),
                             seasonal_order=(sp,sd,sq,7), enforce_stationarity =
                              True, enforce_invertibility = True).fit(method='
                             bfgs', maxiter=300, disp=False)
                        modelSelection.iloc[num]["model"] = "order=(" + str(p) +
                              ","+ str(d) + ","+ str(q) + "), season=("+ str(sp)
                              + ","+ str(sd) + "," + str(sq) + ")"
                        modelSelection.iloc[num]["aic"] = sarima.aic
                        num = num + 1

modelSelection[modelSelection.aic == min(modelSelection.aic)]
```

- Out[26]:

```
                            model       aic
107 order=(2,1,1), season=(0,1,1)   2036.43
```

- In[27]:

```
bestSARIMA = sm.tsa.SARIMAX(new_df_oct, order=(2,1,1), seasonal_order=(0,1,1,7),
      enforce_stationarity = True, enforce_invertibility = True).fit()
SARIMA_NOV_new = bestSARIMA.predict('2020-11-01', '2020-11-30')

SARIMA_NOV_cml = pd.DataFrame(SARIMA_NOV_new)
SARIMA_NOV_cml['cmlsum'] = pd.DataFrame.cumsum(SARIMA_NOV_cml['predicted_mean'])
      + cml_df_oct.iat[-1, 0]
```

MSE を計算する mean_squared_error は sklearn パッケージ内に含まれている。

- In[28]:

```
y_true = cml_df.loc['2020-11-01':'2020-11-30', 'infected']
y_SIR = SIR_NOV_cml
y_SARIMA = SARIMA_NOV_cml['cmlsum']

print('SIRモデルの平均二乗誤差(MSE): {}'.format(mean_squared_error(y_true, y_SIR
     )))
print('SARIMAモデルの平均二乗誤差(MSE): {}'.format(mean_squared_error(y_true,
     y_SARIMA)))
```

- Out[28]:

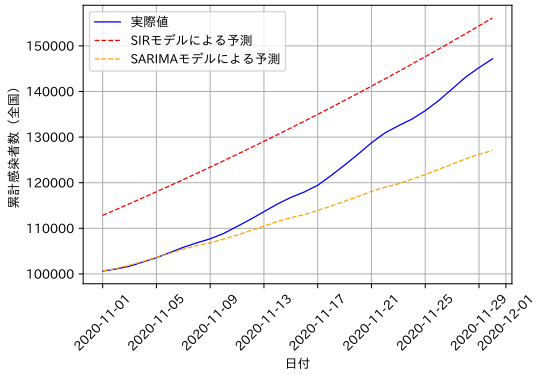
```
SIRモデルの平均二乗誤差(MSE): 185928779.21116066
SARIMAモデルの平均二乗誤差(MSE): 89231716.65574993
```

最後に、結果をグラフにプロットする。この図は 23 ページの図 3.7 として使われている。

- In[29]:

```
plt.plot(y_true, color='blue',  linestyle='solid', linewidth = 1.0,
    label='実際値')
plt.plot(y_SIR, color='red',  linestyle='dashed', linewidth = 1.0, label='
     SIRモデルによる予測')
plt.plot(y_SARIMA, color='orange',  linestyle='dashed', linewidth = 1.0, label='
     SARIMAモデルによる予測')
plt.xlabel('日付')
plt.ylabel('累計感染者数（全国）')
plt.legend()
plt.grid()
plt.xticks(rotation=45)
plt.savefig('figure/NOVest.pdf', bbox_inches='tight')
plt.show()
```

- Out[29]:



A.6　外出自粛と感染率の関係

A.6.1　被説明変数と説明変数の準備

本文で導出した方法により、被説明変数として日毎の累計感染者数及び免疫保持者から日毎の感染率を計算しグラフにする。グラフは 25 ページの図 4.1 で使用されているものである。

- In[30]:

```
df = cml_df
df['inf+1d'] = df['infected'].shift(-1)
df['rem+1d'] = df['removed'].shift(-1)
df = df.dropna(how='any')
df['beta_t'] = ((df['inf+1d']-df['infected'])+(df['rem+1d']-df['removed']))/df['
    infected']
df['gamma_t'] = (df['rem+1d']-df['removed'])/df['infected']
df['R_t'] = df['beta_t']/df['gamma_t']
df = df.drop(columns=['inf+1d','rem+1d'])
print(df)
```

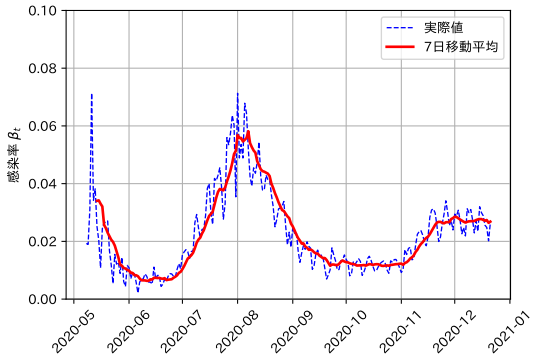
- Out[30]:

```
            infected   removed    beta_t   gamma_t       R_t
date
2020-05-08   15462.0    8710.0  0.019273  0.011577  1.664804
2020-05-09   15581.0    8889.0  0.018933  0.015788  1.199187
2020-05-10   15630.0    9135.0  0.031158  0.026296  1.184915
2020-05-11   15706.0    9546.0  0.071374  0.061951  1.152107
2020-05-12   15854.0   10519.0  0.034250  0.030844  1.110429
...              ...       ...       ...       ...       ...
2020-12-15  182326.0  156658.0  0.032003  0.015220  2.102703
2020-12-16  185386.0  159433.0  0.029959  0.013620  2.199604
2020-12-17  188415.0  161958.0  0.028888  0.013560  2.130333
2020-12-18  191303.0  164513.0  0.025598  0.010773  2.376031
2020-12-19  194139.0  166574.0  0.024807  0.011322  2.191083
```

- In[31]:

```
rolmean = df['beta_t'].rolling(window = 7).mean()
plt.plot(df['beta_t'], color='blue',  linestyle='dashed', linewidth = 1.0,
    label='実際値')
plt.plot(rolmean, color='red',  linestyle='solid', linewidth = 2.0,
    label='7日移動平均')
plt.ylabel(r'感染率 $\beta_t$')
plt.ylim(0, 0.1)
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.savefig('figure/betatRolling.pdf', bbox_inches="tight")
plt.show()
```

- Out[31]:



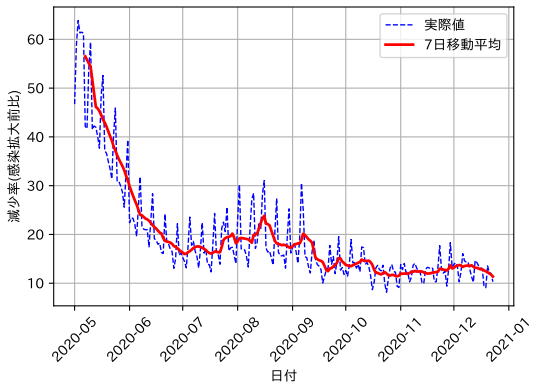
説明変数はモバイル空間統計を使う。日毎感染率同様、日毎の値と移動平均をグラフにする。グラフは 26 ページの図 4.2 で使用されているものである。

- In[32]:

```
mobaku = pd.read_csv('https://mobaku.jp/covid-19/download/%E5%A2%97%E6%B8%9B%E7
    %8E%87%E4%B8%80%E8%A6%A7.csv', encoding="SHIFT-JIS",
    index_col='エリア')
mobaku['percentage'] = mobaku['各日15時時点増減率(%)']
mobaku = mobaku.query('percentage == "感染拡大前比"')
mobaku = mobaku.drop(columns=['メッシュ','各日15時時点増減率(%)','percentage']).
    T
mobaku.index = pd.to_datetime(mobaku.index)

mobaku['average'] = mobaku.mean(axis='columns')
mobaku['inv_avg'] = -mobaku['average']

rolmean = mobaku['inv_avg'].rolling(window = 7).mean()
plt.plot(mobaku['inv_avg'], color='blue',  linestyle='dashed', linewidth = 1.0,
    label='実際値')
plt.plot(rolmean, color='red',  linestyle='solid', linewidth = 2.0,
    label='7日移動平均')
plt.xlabel('日付')
```

```
plt.ylabel('減少率(感染拡大前比)')
plt.xticks(rotation=45)
plt.legend()
plt.grid()
plt.savefig('figure/MobakuRolling.pdf', bbox_inches="tight")
plt.show()
```

- Out[32]:



A.6.2　1 日ずつシフトさせた回帰分析

本文 27 ページの図 4.3 にある図を作成する。具体的には、結果を格納する DataFrame を作ったあとに、for 構文でシフトする日数を決めループ処理を行う。そして最後に、グラフに出力するために格納した値をすべて浮動小数点として保存する。

- In[33]:

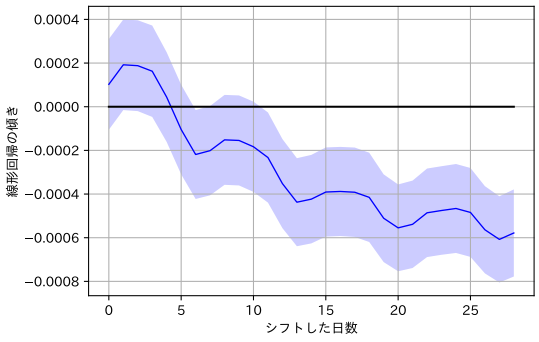
```
data = df[['beta_t','gamma_t']]
data['inv_mobi'] = mobaku['inv_avg']
data = data.dropna(how='any')

result = pd.DataFrame(columns=['coef','lower','upper','f_pvalue','std_err'],
     index=np.arange(0,29))
for i in range(29):
    x = 0
    y = 0

    x = data[['inv_mobi']]
    x = sm.add_constant(x)
    y = data['beta_t'].shift(-i)

    OLS = sm.OLS(y, x, missing='drop').fit()
    coef = OLS.params['inv_mobi']
    lower = OLS.conf_int().iat[1, 0]
    upper = OLS.conf_int().iat[1, 1]
    f_pvalue = OLS.f_pvalue
```

```
    std_err = OLS.bse['inv_mobi']

    result.at[i,'coef'] = coef
    result.at[i,'lower'] = lower
    result.at[i,'upper'] = upper
    result.at[i,'f_pvalue'] = f_pvalue
    result.at[i,'std_err'] = std_err

result['coef'] = result['coef'].astype(float)
result['lower'] = result['lower'].astype(float)
result['upper'] = result['upper'].astype(float)
result['f_pvalue'] = result['f_pvalue'].astype(float)
result['std_err'] = result['std_err'].astype(float)
print(df)
```

- In[34]:

```
zero_line = np.full(29,0)
tspan = np.arange(0,29)
plt.plot(result['coef'], c='blue', linestyle='solid', linewidth = 1.0)
plt.plot(zero_line, c='black')
plt.fill_between(tspan, result['lower'], result['upper'], facecolor='blue',alpha
    =0.2)
plt.xlabel('シフトした日数')
plt.ylabel('線形回帰の傾き')
plt.grid()
plt.savefig('figure/ShiftedBeta.pdf', bbox_inches="tight")
plt.show()
```

- Out[34]:



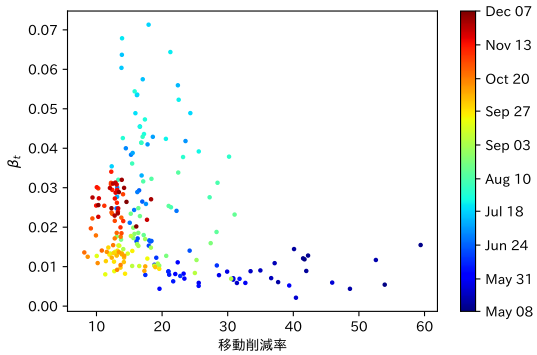
A.6.3 回帰分析の結果とシミュレーション

まずは移動削減率と日毎感染率の散布図を作成する。時間経過による傾向を把握するために、時期によって色がグラデーションで変わるように設定し、右に凡例を設ける。得られたグラフは 28 ページの図 4.4 で使用されているものである。

- In[35]:

```
dt = data
dt['beta'] = dt['beta_t'].shift(-14)
dt['gamma'] = dt['gamma_t'].shift(-14)
dt = dt.dropna(how='any')

x = dt[['inv_mobi']]
y = dt[['beta']]

N_TICKS = 10
smap = plt.scatter(x, y, s=5, c=dt.index, cmap=cm.jet)
indexes = [dt.index[i] for i in np.linspace(0,dt.shape[0]-1,N_TICKS).astype(int
    )]
cb = plt.colorbar(smap, orientation='vertical',ticks= dt.loc[indexes].index.
    astype(int))
cb.ax.set_yticklabels([index.strftime('%b %d') for index in indexes])
plt.xlabel('移動削減率')
plt.ylabel(r'$\beta_{t}$')
plt.savefig('figure/ScatterMobiBeta.pdf', bbox_inches="tight")
plt.show()
```

- Out[35]:



次に、本文 29 ページの表 4.1 で使用する結果を出力する。

- In[36]:

```
X = sm.add_constant(x)
print(sm.OLS(y,X).fit().summary())
```

- Out[36]:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   beta   R-squared:                       0.074
Model:                            OLS   Adj. R-squared:                  0.069
Method:                 Least Squares   F-statistic:                     16.74
Date:                Tue, 22 Dec 2020   Prob (F-statistic):           6.11e-05
Time:                        18:14:12   Log-Likelihood:                613.48
```

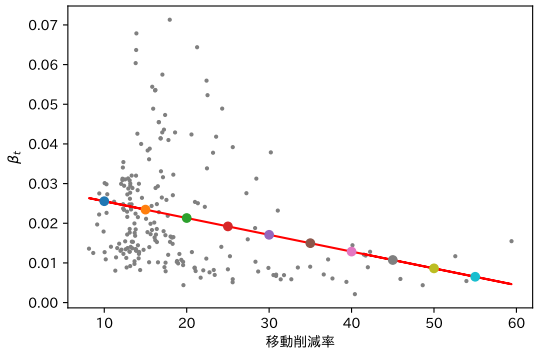
```
No. Observations:                 212   AIC:                             -1223.
Df Residuals:                     210   BIC:                             -1216.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0298      0.002     13.804      0.000       0.026       0.034
inv_mobi      -0.0004      0.000     -4.091      0.000      -0.001      -0.000
==============================================================================
Omnibus:                       50.839   Durbin-Watson:                   0.196
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               81.418
Skew:                           1.333   Prob(JB):                     2.09e-18
Kurtosis:                       4.452   Cond. No.                         48.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
      specified.
```

散布図に回帰直線を重ねる（見づらくなるため、時系列による色のグラデーションはなくした)。回帰直線上に移動削減率 5% おきに点を取る。このときの点の色を、この次の図の色と対応させるため、彩色に使われるカラーマップを指定しておく。このグラフは 30 ページの図 4.5 で使用されているものである。

- In[37]:

```
model_lr = LinearRegression()
model_lr.fit(x, y)

beta_1 = np.full(10,model_lr.coef_[0][0])
beta_0 = np.full(10,model_lr.intercept_[0])

mobi = np.arange(10,60,5)
beta_hat = np.around(beta_1*mobi+beta_0, 5)
test = pd.DataFrame(mobi, columns = ['mobi'])
test['beta_hat'] = beta_hat

plt.plot(x, y, 'o', c='gray', markersize=2)
plt.plot(x, model_lr.predict(x), c='red', linestyle="solid")
cmap = plt.get_cmap("tab10")
for i in range(len(mobi)):
    plt.plot(test['mobi'].iat[i], test['beta_hat'].iat[i],'o', c=cmap(i),
        markersize=6)
plt.xlabel('移動削減率')
plt.ylabel(r'$\beta_{t}$')
plt.savefig('figure/LinReg.pdf', bbox_inches="tight")
plt.show()
```

- Out[37]:

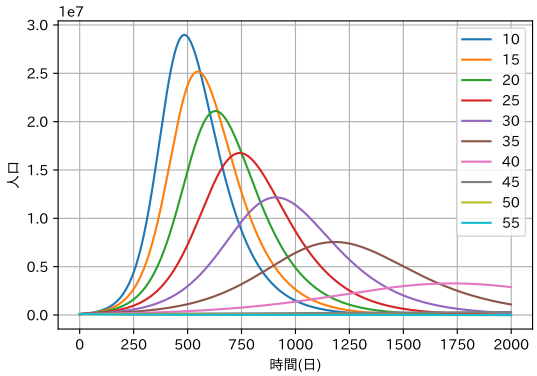


移動削減率を 5% おきに取った点の日毎感染率を取得し、SIR モデルに組み込むことによってシミュレーションを行う。このグラフは 30 ページの図 4.6 で使用されているものである。

- In[38]:

```
tspan = np.arange(0, 2000)
initI, initR, initN = 100000, 100000, 120000000
initS = initN - (initI + initR)
gamma = 0.01

res = pd.DataFrame()
for beta in test['beta_hat']:
    ode = odeint(SIR_model, [initS, initI, initR], tspan, args=(beta, gamma))
    ode_df = pd.DataFrame(ode)
    res[str(beta)]=ode_df.iloc[:,1]

plt.plot(tspan, res)
plt.legend(mobi)
plt.xlabel('時間(日)')
plt.ylabel('人口')
plt.grid()
plt.savefig('figure/BetaSimulation.pdf', bbox_inches="tight")
plt.show()
```

- Out[38]:



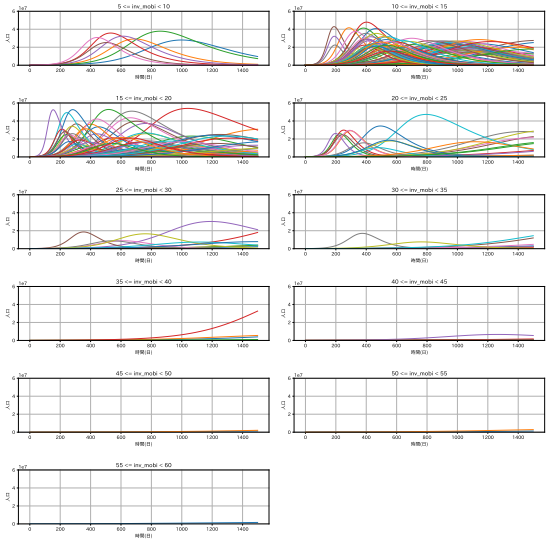
A.6.4　医療の逼迫度の計算

まずは移動削減率を 5% 以上 10% 未満、10% 以上 15% 未満、···、55% 以上 60% 未満の 5% おきに 11 個のグループに分ける。まずはそれぞれのグループを `for` 構文で反復処理させ該当するようなデータ点を抽出したあと、今度は抽出したデータ点について `for` 構文で SIR モデルのシミュレーションを行いグラフにプロットしていく。ここでは 1 つの図に複数のグラフを載せるために `fig.add_subplot` モジュールを使って領域内に各グラフを載せていく。このグラフは 31 ページの図 4.7 で使用されているものである。

- In[39]:

```
group = ['5 <= inv_mobi < 10', '10 <= inv_mobi < 15',
         '15 <= inv_mobi < 20', '20 <= inv_mobi < 25',
         '25 <= inv_mobi < 30', '30 <= inv_mobi < 35',
         '35 <= inv_mobi < 40', '40 <= inv_mobi < 45',
         '45 <= inv_mobi < 50', '50 <= inv_mobi < 55',
         '55 <= inv_mobi < 60']

tspan = np.arange(0, 1500)
initI, initR, initN = 100000, 100000, 120000000
initS = initN - (initI + initR)

fig = plt.figure(figsize=(20,20))

for i in range(11):
    ax = fig.add_subplot(6, 2, i+1)

    grp = group[i]
    sub = dt.query(grp)
    subdf = pd.DataFrame(sub[['beta','gamma']].values, columns = ['beta','gamma
        '])
    res = pd.DataFrame()

    for beta in subdf['beta']:
        ode = odeint(SIR_model, [initS, initI, initR], tspan, args=(beta, subdf[
            subdf['beta'] == beta].gamma))
        ode_df = pd.DataFrame(ode)
        res[str(beta)]=ode_df.iloc[:,1]

    ax.plot(tspan, res, lw=1)
    ax.set_ylim([0, 6e+7])
    ax.set_title(grp)
    ax.set_xlabel('時間(日)')
    ax.set_ylabel('人口')
    ax.grid()

plt.subplots_adjust(wspace=0.1, hspace=0.7)
plt.savefig('figure/BetaGrouped.pdf', bbox_inches="tight")
plt.show()
```

- Out[39]:



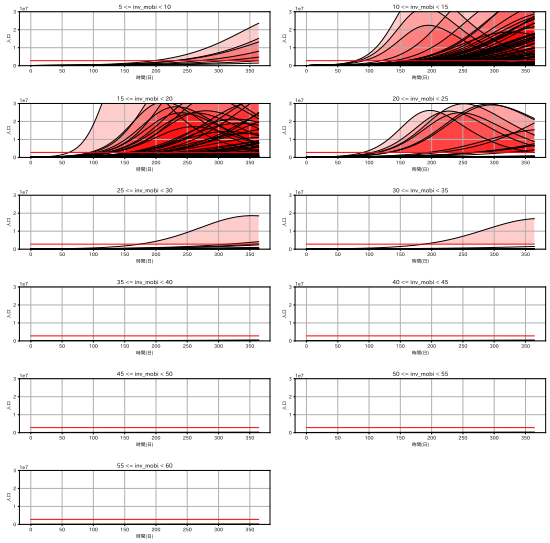
今度は、医療の逼迫の度合いを示す 33 ページの図 4.8 を作成する。大まかな部分は先の図と変わらないが、期間を 365 日にしたこと、各データ点の曲線を黒色にしたこと、データに基づいて閾値を定めたこと、閾値を超えた領域を赤色で塗ったことなどが変更点である。

- In[40]:

```
tspan = np.arange(0, 365)
initI, initR, initN = 100000, 100000, 120000000
initS = initN - (initI + initR)

fig = plt.figure(figsize=(20,20))

for i in range(11):
    ax = fig.add_subplot(6, 2, i+1)

    grp = group[i]
    sub = dt.query(grp)
    subdf = pd.DataFrame(sub[['beta','gamma']].values, columns = ['beta','gamma
        '])
    res = pd.DataFrame()

    limit = np.full(365, 0.28e+7)
    for beta in subdf['beta']:
        ode = odeint(SIR_model, [initS, initI, initR], tspan, args=(beta, subdf[
            subdf['beta'] == beta].gamma))
        ode_df = pd.DataFrame(ode)
        res[str(beta)]=ode_df.iloc[:,1]
        ax.plot(tspan, res[str(beta)], c='k',lw=0.5)
        ax.fill_between(tspan,res[str(beta)],limit,where=res[str(beta)]>limit,
            facecolor='r',alpha=0.2)
```

```
    ax.plot(limit, c='r')
    ax.set_ylim([0, 3e+7])
    ax.set_title(grp)
    ax.set_xlabel('時間(日)')
    ax.set_ylabel('人口')
    ax.grid()

plt.subplots_adjust(wspace=0.1, hspace=0.7)
plt.savefig('figure/BetaArea.pdf', bbox_inches="tight")
plt.show()
```

- Out[40]:



次に本文で定義した「平均医療逼迫度」の計算を行う。まずは結果を格納する DataFrame を作成する。移動削減率 5 %おきのグループでループ処理を行い、各データ点で 365 日分の累計感染者数を日毎に計算し、閾値を超えた分の和を計算する。最後に超えた分の総和をデータ点の数で割れば「平均医療逼迫度」が計算される。

- In[41]:

```
tspan = np.arange(0, 365)
initI, initR, initN = 100000, 100000, 120000000
initS = initN - (initI + initR)

res = pd.DataFrame({'Average Exceed Area': ['0', '0', '0', '0', '0', '0', '0',
     '0', '0', '0', '0'],},
                   index=group)

for i in group:
    sub = dt.query(i)
    subdf = pd.DataFrame(sub[['beta','gamma']].values, columns = ['beta','gamma
         '])
```

```
    limit = 0.28e+07

    summ = 0
    count = 0
    for beta in subdf['beta']:
        ode = odeint(SIR_model, [initS, initI, initR], tspan, args=(beta, subdf[
            subdf['beta'] == beta].gamma))
        infected = ode[:, 1]

        count = count + 1
        diff = 0
        for j in infected:
            if j > limit:
                diff = diff + (j - limit)
        summ = summ + diff

      final = summ / count
      print("{} ... datapoints : {}, Average exceed area : {:.3f}".format(i,
          count, final))
      res.loc[i] = final
```

- Out[41]:

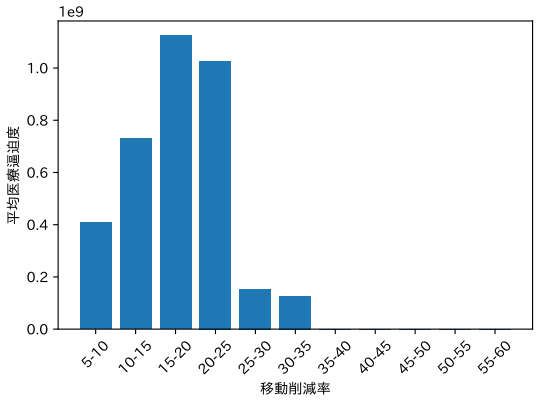
```
5 <= inv_mobi < 10 ... datapoints : 7, Average exceed area : 409065055.726
10 <= inv_mobi < 15 ... datapoints : 83, Average exceed area : 736514872.230
15 <= inv_mobi < 20 ... datapoints : 64, Average exceed area : 1124106488.888
20 <= inv_mobi < 25 ... datapoints : 23, Average exceed area : 1027277553.967
25 <= inv_mobi < 30 ... datapoints : 11, Average exceed area : 153910688.511
30 <= inv_mobi < 35 ... datapoints : 10, Average exceed area : 126697578.372
35 <= inv_mobi < 40 ... datapoints : 4, Average exceed area : 0.000
40 <= inv_mobi < 45 ... datapoints : 6, Average exceed area : 0.000
45 <= inv_mobi < 50 ... datapoints : 2, Average exceed area : 0.000
50 <= inv_mobi < 55 ... datapoints : 2, Average exceed area : 0.000
55 <= inv_mobi < 60 ... datapoints : 1, Average exceed area : 0.000
```

最後に、得られた結果を棒グラフにして出力する。このグラフは 34 ページの図 4.9 で使用されているものである。

- In[42]:

```
label = ['5-10', '10-15', '15-20', '20-25', '25-30', '30-35', '35-40', '40-45',
  '45-50', '50-55', '55-60']
plt.bar(label, res.iloc[::,0].values.astype(np.float), align="center")
plt.xlabel('移動削減率')
plt.ylabel('平均医療逼迫度')
plt.xticks(rotation=45)
plt.savefig('figure/ExceedArea.pdf', bbox_inches="tight")
plt.show()
```

- Out[42]:


1e9
1.0
0.8
0.6
0.4
0.2
0.0
平均医療過迫度
5-10
10-15
15-20
20-25
25-30
30-35
35-40
40-45
45-50
50-55
55-60
移動削減率