COVID-19データを分析する–Rの学習(1)

資　料

COVID-19データを分析する – Rの学習(1)[1]

遠　山　弘　徳

Ⅰ．モチベーション

COVID-19(新型コロナウィルス)は，2019年末，中国湖北省武漢市で発生し，瞬く間に中国全土に拡大し，感染者84,123人，死者4,638人と甚大な人的被害をもたらしました(2020年５月30日時点)．図１は３月１日と３月20日の２時点における感染者数を世界地図上に描いたものです．わずか３週間弱でアジアからヨーロッパ，アメリカへ，さらに全世界へと拡大して行ったことがわかります．2020年５月30日現在，世界全体で感染者数5,930,096人，死者数365,034人に達し，わたしたちははじめてリアルタイムでパンデミックの恐怖を経験しています[2]．

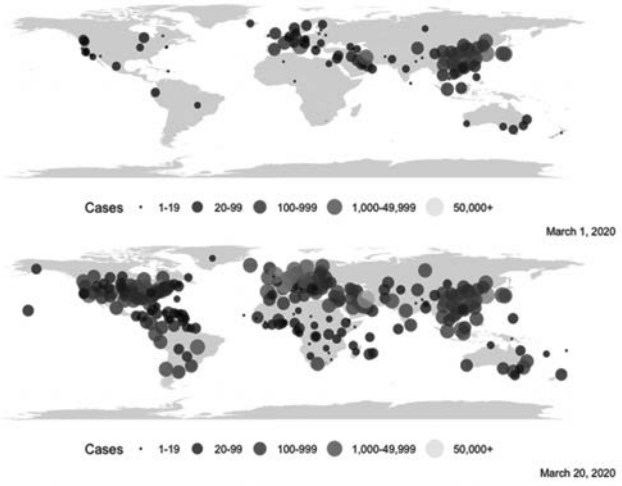


図１　感染者数の変化　2020年３月

注．データソースはジョン・ホプキンス大学システムサイエンス・エンジニアリングセンター(JHU CSSE)のGitHubサイト．図の作成にあたってはMap Visualization of COVID-19 Across the World with R (https://datascienceplus.com/map-visualization-of-covid19-across-world/)を参考にした．

[1] 本資料は主としてCOVID-19データを素材にRを学ぶために作成されたものです．このため以下のRを使ったデータ処理の解説においては，本資料作成時点(４月４日)のデータが中心となっています。

[2] データはJohn Hopkins, Coronavirus Resource Center(https://coronavirus.jhu.edu/map.html)．

中国では３月に入り，新規感染者が大幅に減少し，３月10日，中国政府はCOVDI-19の収束に成功したとのメッセージを発しています．中国では収束を見たものの，２月末には東アジア，３月初頭にはヨーロッパ，そして４月にはアメリカにおいて指数関数的に感染者が増加し，急速に死亡者数も増加しています．図２は，人口比を考慮せずに，アジア地域とヨーロッパ地域の国々におけるCOVID-19による死者の数を示しています．アジアでは，死者の増加がストップした中国を除けば，インド，インドネシア，フィリピンが上昇傾向を示しています．アジア以上に深刻なのはヨーロッパです．桁違いの死者を記録しています．中でもイタリア，スペイン，フランスおよびイギリスは突出した水準を示しています。さらに，ヨーロッパの状況を上回るのがアメリカ(US)です．直近のデータ(2020年５月30日)によれば，確認された感染者数は178万人，死者は10万人を超えています。いまや新型コロナウィルスの震源地はアメリカへと移っています。

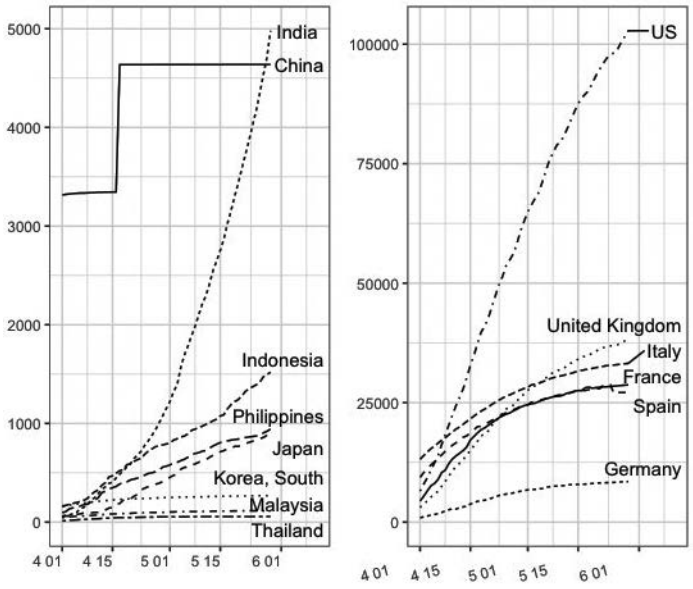


図２　COVID-19による死亡者数(累積)の推移 2020年４月１日～５月30日

注．データソースはJHU CSSEのGitHubサイト

現在，新型コロナウィルスの脅威が世界全体を覆い尽くしています．本資料は，COVID-19(新型コロナウィルス)データの分析を行いながらRおよびRStudioを学ぶということを目的に作成された資料です．もっとも，こうした学習によって新型コロナウィルスの脅威や経済活動との関連の一端も，同時に，窺い知ることができると思います．具体的には以下の６点を中心に解説していきます．

1．RとRStudioの基本的な使い方.
2．ggplot2パッケージを利用したグラフの作成方法.
3．COVID-19データの取得方法と読み込み方法，およびデータの加工方法.
4．世界銀行のWorld Development Indicatorsの取得方法，および経済指標とCOVID-19 – 回帰分析(以上本号)
5．COVID-19に対する社会経済的レジリエンス – 主成分分析を利用した次元の縮小(２号)
6．Twitter上のCOVID-19 – twitterデータの収集方法，および収集したテキストデータの分析(３号)

Ⅱ．Rとは

R(アール)はフリーの統計解析向けのプログラミング言語およびその開発実行環境です．多くの研究者，エンジニアたちの共同作業・協力によって開発され，日々進化を続けています．今ではRはデータを解析する者にとって「共有資源」となっています．Rの統合開発環境であるRStudio(アール・スタジオ)とあわせて利用することでデータ解析からレポート作成までの一連の作業をすべてRによって行うことができます[3]．

それでは早速，RとRStudioをパソコン(以下PC)にインストールしてみましょう．次のサイトにアクセスし，指示どおりに進めれば，ダウンロードできます．

R　https://cran.r-project.org
RStudio　https://rstudio.com/products/rstudio/download/

実際の作業はすべてRStudio上で行います．R本体で作業することはありません．しかし，RがPCにインストールされていないと，RStudioは使えませんので必ずインストールしてください．

Ⅱ．1．Rを使ってみる

RStudioをクリックし起動させてみましょう．図３のような画面が現れます．PCにRがインストールされていれば，Rを起動させることなく，Rの機能を利用できます．RStudioの画面は４つのウィンドウから構成されます．ソース(Source)，コンソール(Console)，環境等(Environment, History, Connections)，ファイル等(Files, Plots, Packages, Help, Viewer)の４つのウィンドウ(ペイ

[3] Rの解説はネット上にあふれています．ネット検索をすれば，Rに関するトラブルのほとんどが解決します．じっくり学びたい人はWickham & Grolemund(2017), Kabcoff(2011)がおすすめです．

ン)です．おそらく，最初に起動したときには，図3のように，3つの画面になっていると思いますが，とりあえず気にせず作業を進めて行きましょう．なお，プルダウンメニュー[RStudio]→[Preferenes]で環境設定を開くことができます．環境設定において4つの位置は自由に変更できますので，自分の使い勝手の良いように変更してください．

図3　RStudio画面

注．図の左側がコンソール，右上が環境等，右下がファイル等のウィンドウ．

いま，図4のような年齢と体重のデータを持っているとします．

年齢	体重
1	4.4
3	5.3
5	7.2
2	5.2
11	8.5
9	7.3
12	6
3	10.4

図4　データ例

RStudioのコンソール画面の「>」(プロンプト)の横に次のとおり入力してください．入力にあたっては必ず英数半角で行ってください．入力後エンターキーを押してください．

```
> age <- c(1, 3, 5, 2, 11, 9, 12, 3)
```

ageは「age」という名前をつけた「容れ物」―オブジェクトと呼ばれます―です．容れ物の名前は自分で好きなようにつけてかまいません．“<―”(代入演算子)は，矢印の右側のものを「容れ物age」に容れなさい，という命令です．cは「まとめて」ということを示しています[4]．したがって上の表現は，cのついたカッコ中の年齢データをまとめてageという容れ物に容れなさい，ということ示しています．

次に，コンソール画面に容れ物の名前ageを入力し，エンターキーを押してみてください．

```
> age
```

画面に，「1, 3, 5, 2, 11, 9, 12, 3」と容れ物の中身が表示されます．次に，もう１つ容れ物―weightという名前にします―を作ってみましょう．同じように，図４の体重データを次のように入力し，エンターキーを押して下さい．

```
> weight <- c(4. 4, 5. 3, 7. 2, 5. 2, 8. 5, 7. 3, 6, 10. 4)
```

これで２つの容れ物ができました．これを使って基本的な統計量を計算してみましょう．体重の平均値を取得したい場合はmean(容れ物の名前)，標準偏差を取得したい場合はsd(容れ物の名前)と入力します．さらに年齢と体重の相関係数はcor(容れ物の名前, 容れ物の名前)です．以下のように入力し，エンターキーを押してみてください．結果が[1]の後に表示されます．

```
> mean(weight)
[1] 6. 7875
```

```
> sd(weight)
[1] 1. 980936
```

```
> cor(age, weight)
[1] 0. 2829368
```

以上は表計算ソフトでも簡単に計算できます．これだけではRの有り難みがわかりませんので，

[4] cはconcatenateのcですので，連結するという意味です

体重と年齢の散布図を作成してみましょう．

```
> plot(age, weight)
```

これだけでx軸を年齢，y軸を体重とした散布図が自動的に作成され，RStudioの右側のプロット画面に図５のような散布図が表示されます．plot(x軸の変数,y軸の変数)は散布図を描く関数です．

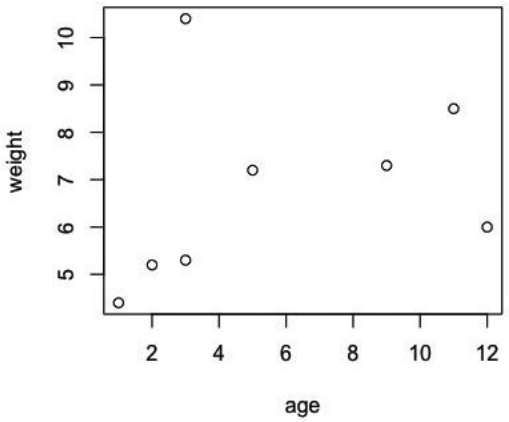


図５　ageとweightの散布図

このように簡単に散布図を作成することができます．しかし，これはとてもシンプルなグラフです．より表現に富んだエレガントなグラフを作成するためには，ggplot2と呼ばれるパッケージをRに組み込む必要があります．

Ⅱ．２．Rパッケージを組み込む

パッケージは，簡単に言えば，R上で動くアプリケーションソフトのようなものです．Rとパッケージは，Rを基本ソフトウェア(OS)，そしてパッケージをそのOS上で動くアプリケーションソフトにたとえることができます．Rだけでも基本的なことはできますが，パッケージを組み込むことによってRに新たな機能を追加できます．これによって自分でプログムを組んだり，関数を作ったりせずとも，簡単に，表現に富んだグラフを作ったり，さまざまな統計解析ができるようになります．この点がRのもっとも優れたところであり，魅力的な点です．

パッケージは多くの研究者・エンジニアによって開発され，Rの利用者に提供されています．Rが共有資源だということを実感できる特長です．たとえば，新型コロナウィルスの分析のための

パッケージ―nCov2019(付録２参照)―もすでに開発されています．ちなみに，上の２つの図１と図２の作成にあたってはこのパッケージも利用しています．

Rを使ったデータのグラフ化，統計解析作業にはパッケージを頻繁に利用することになります．ここで実際にパッケージをR本体に組み込んでみましょう．ここで組み込むパッケージは，ggplot2というエレガントなグラフを作成するためのパッケージです．図１と２の作成にあたっては，このパッケージを利用しています．コンソール画面に次のように入力し，エンターキーを押してください．

```
> install. packages(“ggplot2”)
```

コンソール画面にパッケージがインストール中であることが表示されます．Rにggplot2パッケージが組み込まれたかどうかを確認してみましょう．図３のRStudio画面の右下のウィンドウをみると，[Packages]というタブに気づくでしょう．そこをクリックすると，このRに組み込まれているパッケージ一覧が表示されます．下方向にスクロールしていくと，ggplot2という名前を確認することができます．これであなたのRにはggplot2が無事組み込まれました[5]．

Ⅱ．３．データフレームとオブジェクト

Rの操作を学ぶ上で,「データフレーム」と呼ばれるデータ構造と「オブジェクト」という概念を理解しておくことが重要になります．データフレームはRの基本的なデータ形式です．上述の体重と年齢の表(図４)をイメージしてかまいません．ただ，データフレームは数値データだけではなく，文字データも容れることができます．たとえば，上の表に名前のデータ(文字列データ)を容れたnameという「容れ物」を加えてみましょう．文字データは“”で囲む必要があります．コンソール画面に次のように入力し，エンターキーを押してください．

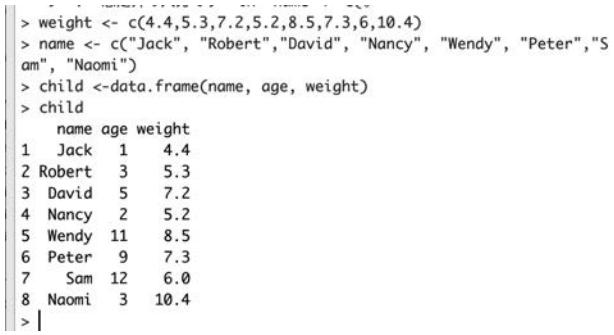
```
name <- c(“Jack”, “Robert”, “David”, “Nancy”, “Wendy”, “Peter”, “Sam”, “Naomi”)
```

それでは３つのデータ―体重，年齢および名前―を持つデータフレームを作成してみましょう．データフレームの名前はchildにしておきます．

[5] install. packages()コマンドを利用しなくとも，このメニューからもパッケージを組み込むことはできます．[install]をクリックすると，インストール画面が現れます．そこにggplot2と入力することでも同じ結果を得ることができます．

```
child <- data.frame(name, age, weight)
```

“data.frame”という文字は「データフレームを作成しなさい」という命令を表現しています．もう少し詳しく言うと，「name, age, wightという3つのオブジェクト(変数)を列とするデータフレームを作成しなさい」ということを表しています．そして“<-”(代入演算子)によって「作成したデータフレームをchildという容れ物に格納しなさい」という命令を表現しています．これでchildという名前の，図6のようなデータフレーム―表計算で言えばワークシート―ができます．コンソール画面にchildと入力し，エンターキーを押してみてください．図6のような結果がコンソール画面に表示されるはずです．



```
> weight <- c(4.4,5.3,7.2,5.2,8.5,7.3,6,10.4)
> name <- c("Jack", "Robert","David", "Nancy", "Wendy", "Peter","S
am", "Naomi")
> child <-data.frame(name, age, weight)
> child
    name age weight
1   Jack   1    4.4
2 Robert   3    5.3
3  David   5    7.2
4  Nancy   2    5.2
5  Wendy  11    8.5
6  Peter   9    7.3
7    Sam  12    6.0
8  Naomi   3   10.4
>
```

図6　データフレームchild

オブジェクトとは上述の例で言えば，「体重weight」や「年齢age」といった名前を付けた「容れ物」です．数学的に表現すれば変数ということになりますが，「オブジェクト」と呼ぶのはこの容れ物が一連の数値データだけではなく，文字のあつまり―名前，国名など―，数値と文字の混在したあつまりを一まとめにしたものも容れることができるからです．さらに表計算ソフトのシート全体も容れることができる「容れ物」です．

特定の操作を行うためにオブジェクトの中身を取り出す必要があるかもしれません．そのようなときコンソール画面に次のように入力してみてください．$をつけることによってオブジェクトchildの中身を取り出すことができます．「$」は「オブジェクトchildの中の～」ということを表現しています．

```
> child$
```

と入力すると，オブジェクトchildの中のname, age, weightが表示されます．ここではnameを選んでみましょう．つまり，次のように入力し，エンターキーを押してみてください．

```
> child$name
```

すると，容れ物の中身の名前一覧が表示されます．別の取り出し方もあります．データフレームの形式―表計算のワークシートをイメージ―を意識し，データフレームの１列目のデータを取り出す場合には，次のように入力し，エンターキーを押します．すると，一列目が表示されます．

```
> child[,1]
```

これはワークシートの行と列をイメージすれば，簡単に理解できます．Rでは行列は[行，列]で表現されます．[,1]では列は第１列が指定されていますが，行は何も指定されていません([]内のコンマの左側が空欄)．これはすべての行を意味します．

Ⅱ．４．スクリプト―再現性はとても大切

みなさんは表計算ソフトを使ったことがあると思います．そのさい，相関係数を求めたり，グラフを作成したりする作業はすべてプルダウン・メニューから行っていると思います．データ分析においてはグラフの作り直しや分析作業のやり直しが頻繁に求められます．この場合，いちいちメニューからたどるのは面倒ですし，効率的ではありません．何よりも結果を再現することがとても難しくなります．

Rを利用した分析においては，操作コード(指示・命令)のあつまりを記録したスクリプトと呼ばれるノートを作成します．一度，これを作成しておけば，何度でも簡単に操作をやり直せることができます．これまでの一連の作業をスクリプトに書いてみましょう．作業を一度スクリプトに記録し，保存しておけばいつでも作業経過と分析結果を再現できます[6]．

プルダウンメニュー[File]→[New File]→[R Script]で新しいスクリプトをオープンしてください．あるいはスクリプトはRstudioの左上の[＋]アイコンをクリックするだけでもオープンできます．

[6] COVID-19データは日々変化します．そのため作成したグラフや分析結果を常にアップデートする必要がありますが，一度，スクリプトで作成しておけば，とても簡単にアップデートできます．

図7　新規スクリプトのオープン

新規スクリプトをオープンできたら，あとはワープロと同じように書き込んでいくだけです．これまでコンソール画面で実行してきた命令をまとめて書き込んでみましょう．

```
age <- c(1, 3, 5, 2, 11, 9, 12, 3)
weight <- c(4. 4, 5. 3, 7. 2, 5. 2, 8. 5, 7. 3, 6, 10. 4)
mean(weight)
sd(weight)
cor(age, weight)
plot(age, weight)
name <- c(“Jack”,“Robert”,“David”,“Nancy”,“Wendy”,“Peter”,“Sam”, “Naomi”)
child <-data.frame(name, age, weight)
```

それぞれの命令を実行するには，スクリプト画面の右上の[Run]をクリックします(ショートカットはCmd/Ctrl+Enter)．たとえば，一番最初の「ageに年齢データを容れなさい」という命令を実行するには，

手順1　age <- c(1, 3, 5, 2, 11, 9, 12, 3)の行にカーソルを持って行く
手順2　[Run]をクリック

これだけです．結果はコンソール画面に表示されます．同じように，それぞれの行を実行してみてください．スクリプトは必ず保存しておきましょう．スクリプト画面のフロッピーアイコンをクリックすると，保存先を尋ねてきますので，ファイル名を付けてPC上の適当な場所に保存してください．ここでは「script1」という名前で保存してください．

ここで１つ重要な注意です．この程度の長さのスクリプトならば記載されている内容はすぐに理解できますし，思い出すこともできます．しかし，数百行に及ぶスクリプトとなると，あとで見返しても，いったい何をやっているのか書いた本人でも分からなくなることがあります．そこでスクリプトにはコメントを書いておくことをお奨めします．Rは＃記号の後の部分(から改行する前まで)は無視しますので，＃をつけてコメントを書いておくと，あとで見返したとき，とても役に立ちます(付録のスクリプト一覧を参照してください)．

Ⅱ．５．RStudioを終了する

スクリプトscript1を保存できましたので，ここでいったん作業を終了しましょう．コンソール画面に次のように入力してくだい．

```
> q( )
```

グラフを作成した場合，Save workspace image…?　(イメージを保存しますか)と尋ねてきますが，すでにスクリプトが保存されてますから，グラフが失われてもまったく問題がありません．Noを選択しておきましょう(キーボードのNを押す)．スクリプトさえあれば，いつでも同じ作業や結果を再現できます．

Ⅲ．ggplot2の紹介

実際に，新型コロナウィルスのデータをグラフ化する前に，ここでごく簡単にグラフ作成パッケージggplot2を紹介しておきましょう．組み込まれたパッケージを利用するには，library()を使って宣言しておく必要があります．新規スクリプトをオープンし，スクリプトに次のように入力し，[Run]をクリックしてください．

```
library(ggplot2)
```

これでggplot2パッケージが使えるようになりました．ggplot2以外のパッケージの場合でも，パッ

ケージを利用する場合には，事前に，library()で利用するパッケージを呼び出しておく必要があります．覚えておきましょう．

Ⅲ．1．ggplot2の基礎

Rには当初からいくつかのデータセット(データフレーム)が用意されていますが，ここではcarsというデータセットを使ってggplotを使ってみましょう．carsは2つの変数―スピードspeedと距離dist―と50の観察値を持つ50行×2列のデータセットです(コンソール画面にオブジェクト名carsを入力しエンターキーで確認してみてください)．スピードと距離の散布図を描くためには，スクリプト画面に次のように入力します．

```
ggplot(cars)+
geom_point(aes(speed, dist))
```

この後，[Run]をクリックすると，スピードと距離の散布図が表示されます．これがもっとも基礎的なggplotの利用法です．ggplotでは描画の基本は2つの構成要素―レイヤー(層)と呼ばれています―からなります．第1のレイヤーは，利用するデータフレーム(データセット)を指定します．このレイヤーはもっとも基礎的な部分ですので，これは必ずRに与えないといけない指示です．書式は以下のとおりです．

```
ggplot(データフレーム名)
```

第2のレイヤーは，データの座標上の位置を何で表現するかを指示する部分です．geom_関数で示されます．ここではgeom_pointを使って「点」で表示するように指示しています．そしてそのデータフレームの中のどの変数をx軸とy軸にするかをaes( )で指示します．ここではaes(speed, dist)で指定します．つまりx軸にはスピード(speed)変数，y軸には距離(dist)変数を指定しています．

```
geom_point(aes(x軸変数名,y軸変数名))
```

この2つのレイヤーを結びつけるのは＋記号です．これは＋でどんどん新たなレイヤーを付け加えて行くことができることを示しています．注意しないといけないのは，＋は行末に置かなければならないということです．行頭に置くとエラーになります．ggplot2のグラフの作成方法は何

枚ものレイヤーを重ねて作成して行くというイメージです．これを見るために，１つ例を示しましょう．上で作成したグラフにさらにタイトルを付け，x軸とy軸のラベルを変えるグラフを作成してみましょう．このためにlabs()レイヤーを重ねます．

```
ggplot(cars)+                                                    ☜第１のレイヤー
geom_point(aes(speed, dist))+                                    ☜第２のレイヤー
labs(x="Speed",y="Distance",title ="Speed and Distance")         ☜第３のレイヤー
```

出力結果は以下の図８のようになります．お分かりのように，グラフにタイトル，x軸およびy軸ラベルがつきました．

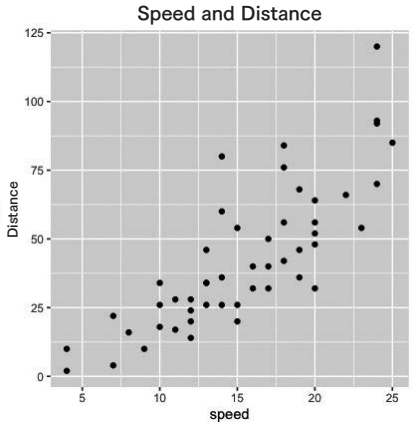


図８　スピードと距離の散布図

Ⅲ．２．レイヤーを重ねる

レイヤーを重ねてグラフを描くということを理解するために，さらに，いくつかのレイヤーを重ねて，グラフをさらにエレガントにしたいと思います．ここでは以下の作業を行います．

１）基本的なグラフを描く．	ggplot( )
２）タイトルをつける．軸ラベルを変更する．	labs( )
３）近似直線を描く．	geom_smooth( )
４）注釈を加える．	annotate( )
５）背景(テーマ)を変更する．	theme_bw( )

1)と2)はすでに上で行った作業です．そこでここでは3)以下の作業を行います．スピードと距離の散布図の作成にあたってはデータは点で表示されました．3)では同じスピードと距離のデータを近似曲線(スムーズな曲線)で表現したいと思います．このため，データ表現に利用する幾何関数はgeom_smoothです．次のようにスクリプト画面に入力し，[Run]を押して下さい．図9の中央の図が表示されます．

```
ggplot(cars)+
geom_smooth(aes(speed, dist))+
labs(x="Speed", y="Distance", title = "Speed and Distance")
```

このレイヤーを図9の左側の基本的な図に重ね，点と近似曲線の2つでデータを表現してみましょう．

```
ggplot(cars)+
geom_point(aes(speed, dist))+
labs(x="Speed", y="Distance", title = "Speed and Distance")+
geom_smooth(aes(speed, dist))                    ☜近似曲線のレイヤーを重ねる
```

この近似曲線表現のレイヤーを重ねることによって図9の右側の図が描かれます．

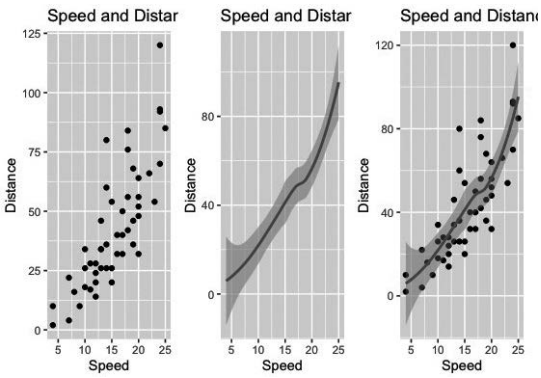


図9　レイヤーを重ねる

それでは次に4)の注釈レイヤーを重ねる作業に移ります．注釈のためにはannotate関数を使います．基本的な書式は

```
annotate(“text”, x=テキストのx座標, y=テキストのy座標, label=“表示するテキスト”)
```

です．注釈レイヤーを重ね，実行[Run]してみてください．注釈つきのグラフが作成されたと思います．

```
ggplot(cars)+
geom_point(aes(speed, dist))+
labs(x=“Speed”, y=“Distance”, title = “Speed and Distance”)+
geom_smooth(aes(speed, dist))+
annotate(“text”, x=10, y=100, label=“A simple graph”)                    ☜注釈レイヤー
```

最後にテーマを変更してみましょう．ggplot2にはデフォルトで８つのテーマが用意されています．ここではグリッドつきの白黒画面に変更してみます．テーマの変更にtheme_関数を利用します．白黒はtheme_bw()です．それではこのレイヤーを重ねてみましょう

```
ggplot(cars)+
geom_point(aes(speed, dist))+
labs(x=“Speed”, y=“Distance”, title = “Speed and Distance”)+
geom_smooth(aes(speed, dist))+
annotate(“text”, x=10, y=100, label=“A simple graph”)+
theme_bw()                                                    ☜テーマのレイヤーを重ねる
```

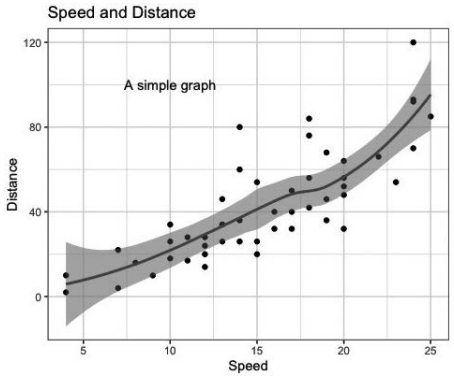


図10　白黒テーマ

以上のスクリプトはscript2として保存しておきましょう．ggplot2の機能は1冊の本でも語り尽くせないほど豊富です．より詳しくはChang(2013)やhttps://ggplot2.tidyverse.org/index.htmlにあたってみて下さい[7]．

Ⅳ．COVID-19の分析

これまでR，RStudioやパッケージggplot2の基本的な使い方をみてきました．それでは実際にCOVID-19のデータをダウンロードし，Rを使って分析してみたいと思います(「分析」と言ってもグラフ化が中心です)．

Ⅳ．1．データの編集とグラフ化

本節ではCOVID-19のデータを取得し，Rに読み込ませた上で，dplyr, tidyverseパッケージを使った基礎的なデータの加工法を解説します．その上ですでに紹介したggplot2パッケージを利用し，グラフを作成します．

Ⅳ．1．1．COVID-19データを取得(ダウンロード)する

新型コロナウィルスの最新のデータはジョン・ホプキンス大学システムサイエンス・エンジニアリングセンター(JHU CSSE) によって提供されています[8]．また，新型コロナウィルスの現状を把握するために，とても有益なRパッケージもすでに開発されています．Wu, Hu, Tun, GeおよびYu氏たちによって開発されたnCov2019です．彼らは各国のインタラクティブなプロットおよび時系列データを表示するウェブサイトも開発しています(http://www.bcloud.org/e/)．本サイトでも新型コロナウィルスの感染者数，死者数および退院者数に関して中国と世界の2種類のデータが提供されています．

ここではRパッケージcoronavirusを開発したRami Kripin氏のGitHubサイトからデータを取得します．同氏のGitHubサイト(https://github.com/RamiKrispin/coronavirus-csv)にアクセスし，直接，ファイルをクリックするか，[clone or downlod]▼をクリックし，Download ZIPを選択し，ダウンロードしてください[9]．データは自分のPC適当な場所に保存してください．この例では

[7] なお，同サイトにおいて提供されているCheatsheetはとても便利です．ggplot2の機能を網羅したシートです．
[8] https://github.com/CSSEGISandData/COVID-19にアクセスし，[clone or downlod]▼をクリックし，Download ZIPを選択し，ダウンロードしてください．フォルダの中にはいくつかのフォルダとファイルがありますが，[csse_covid_19_data]フォルダ，さらに[csse_covid_19_time_series]ファルダの中のtime_series_19-covid-Confirmed.csvです．これが感染者数のデータです．
[9] オリジナルなデータはJHU CSSEからのものです．

[Desktop]に保存しています．なお，データは常に更新されていますが，本資料では2020年４月４日時点のものを利用しています．

Ⅳ．１．２．COVID-19データを読み込む

データの準備はできました．あとはRの出番です．ダウンロードし，保存したデータをRに読み込みませるとしましょう．簡単な方法はRStudioの右上の窓にある[Import Dataset]を利用することです．

手順１　RStudioの右上画面の[Import Dataset]をクリック
手順２　プルダウンメニューから[From Text(readr)]を選択→[Browse..]ボタンをクリック→保存したデータファイルを指定し開く
手順３　右下の[Import]をクリック

これでデータが読み込まれ，読み込むと同時にViewウィンドウが自動的に開きデータを見ることができます．しかし，忘れてはいけないことは分析作業・結果の再現性です．コンソール画面に次のような表示が出力されているはずです．最初はcsvファイルを読み込むために必要なパッケージの名前です．２番目は読みこんだデータを，coronavirus_datasetという名前のオブジェクトに格納しています．なお，この例ではファイルは[Desktop]に保存されています．

```
> library(readr)
> coronavirus_dataset <- read_csv(“Desktop/coronavirus_dataset.csv”)
```

作業の再現のためにこれをそっくりコピーし，スクリプトにベーストしておきましょう．次の作業からはいちいち[import Dataset]のプルダウンメニューを利用せずとも，スクリプト画面の[Run]をクリックすれば，データを読み込むことができます．

ちなみに，データを格納するオブジェクト名は自由です．このままでは長くて入力が面倒なので coronavirus_dataset ではなく，covidに名前を変更しておきます．スクリプト上で次のように書き換えておきましょう．

```
covid <-   read_csv(“Desktop/coronavirus_dataset.csv”)
```

これで新型コロナウィルスのデータはcovidに格納されます．データの中を見たい場合，いくつ

かの方法がありますが，ここでは２つ紹介しておきます．コンソール画面もしくはスクリプト画面，いずれに入力しても同じです．再現する必要のある作業ならスクリプトに記録しておくのが良いでしょう．

1．head(covid)　これを実行すると，コンソール画面にデータの先頭部分が表示されます．ちなみに，データの末尾部分をみるためのコードはtail(covid)です．
2．View(covid)　これを実行すると，View画面が開き，スクリプト画面に全データが表示されます．

このデータの読み込みスクリプトはscript3という名前をつけて保存しておきましょう．

Ⅳ．１．３．COVID-19データを加工する

多くの場合，取得したデータはそのままでは分析に適しないことがあります．そのため自分の分析目的にあった形にデータを編集し直す必要があります．これが意外と厄介なのですが，この厄介な作業を簡単にするためdplyrパッケージがあります．

最初に，Views(covid)でオブジェクトcovidの中をみてみましょう．

Filter

	Province.State	Country.Region	Lat	Long	date	cases	type
1	NA	Afghanistan	33.0000	65.0000	2020-01-22	0	confirmed
2	NA	Afghanistan	33.0000	65.0000	2020-01-23	0	confirmed
3	NA	Afghanistan	33.0000	65.0000	2020-01-24	0	confirmed
4	NA	Afghanistan	33.0000	65.0000	2020-01-25	0	confirmed
5	NA	Afghanistan	33.0000	65.0000	2020-01-26	0	confirmed
6	NA	Afghanistan	33.0000	65.0000	2020-01-27	0	confirmed
7	NA	Afghanistan	33.0000	65.0000	2020-01-28	0	confirmed
8	NA	Afghanistan	33.0000	65.0000	2020-01-29	0	confirmed
9	NA	Afghanistan	33.0000	65.0000	2020-01-30	0	confirmed
10	NA	Afghanistan	33.0000	65.0000	2020-01-31	0	confirmed
11	NA	Afghanistan	33.0000	65.0000	2020-02-01	0	confirmed
12	NA	Afghanistan	33.0000	65.0000	2020-02-02	0	confirmed
13	NA	Afghanistan	33.0000	65.0000	2020-02-03	0	confirmed
14	NA	Afghanistan	33.0000	65.0000	2020-02-04	0	confirmed
15	NA	Afghanistan	33.0000	65.0000	2020-02-05	0	confirmed
16	NA	Afghanistan	33.0000	65.0000	2020-02-06	0	confirmed
17	NA	Afghanistan	33.0000	65.0000	2020-02-07	0	confirmed
18	NA	Afghanistan	33.0000	65.0000	2020-02-08	0	confirmed

図11　Rami Kripin氏編集COVID-19データ
注．データ・ソースはJHU CSSEのGitHubサイト

図11と同一のデータが表示されると思います．[Counrty.Region]は国・地域，[date]は日時を示す変数です．[cases]変数は確認感染者(confirmed)，死亡者(death)の人数を示しています．[type]変数は感染者(confirmed)か死者(death)かの２つの値を持ちます．くわえて位置情報の経度(Lat)

と緯度(Long)が提供されています．このおかげで図１のように地図上にデータを表示することができます．

このデータをもとに日本の１日あたりの感染者推移をグラフ化しみましょう．しかし，そのためにはデータを事前に加工する必要があります．Rパッケージにはdplyrという便利なパッケージがあります．このパッケージはtidyverseパッケージに入っていますので，このパッケージを組み込みましょう．パッケージのインストールはコンソール画面に次のように入力し，エンターキーを押せばOKです．

```
> install. packages(“tidyverse”)
```

インストールを終えたら，このパッケージを利用するために，先ほどのスクリプトscript3を開き，次のように入力し実行[Run]します．

```
library(tidyverse)
```

それではdplyrを使って上のテーブルを次のように編集してみましょう．作業はおもにdplyrのfilter( )関数を使います．使い方はfilter(データフレーム名，データフレームにフィルタをかける条件)です．作業としては次の３つになります．

１．Country.Region変数から観察値Japanの行を取り出す　filter( )
２．Case変数から観察値Confirmedの行を取り出す
３．日本の感染者数Confirmedの時系列グラフ(図12)を描く

スクリプト３に次のように入力して下さい．

```
filter(covid, Country.Region == “Japan”)
```

このコードを実行すると，コンソール画面に条件Country.Region == “Japan”を満たす行だけが抽出されます．書式はfilter(データフレーム名，抽出条件)です．この結果は，代入演算子<-を使ってjpと名前をつけたオブジェクトに保存します．

```
jp <- filter(covid, Country.Region == “Japan”)
```

１の作業はこれで終わりましたが，さらに，２の作業つまり感染者Confirmedだけ取り出す必要があります．jpにさらにfilter( )をかけて良いのですが，filter( )には２つの条件を入れることができますので，上のfilter( )条件を次のように書き換えて下さい．

```
jp <- filter(covid, Country.Region == “Japan”, case == “Confrimed”)
```

これで１と２の作業が終了です．それでは最後に３の作業を行ってみましょう．これはすでに学習済みのggplot( )を利用することになります．ただし，ここでは棒グラフでデータを表現します．このためgeom_オブジェクトはgeom_colを使います.geom_col(aes(xが離散変数, yは連続変数))です．

```
ggplot(jp)+                          ☜第１のレイヤー　データフレーム名を指定
geom_col(aes(date,cases))    +       ☜第２のレイヤー　幾何オブジェクトを棒グラフにする
theme_bw()                           ☜第３のレイヤー　テーマを白黒にする
```

これを実行[Run]すると，図12の上のようなグラフが描かれます．なお，図12の下のグラフはⅢ節で説明したggplot2の使い方を応用して描いたものです．

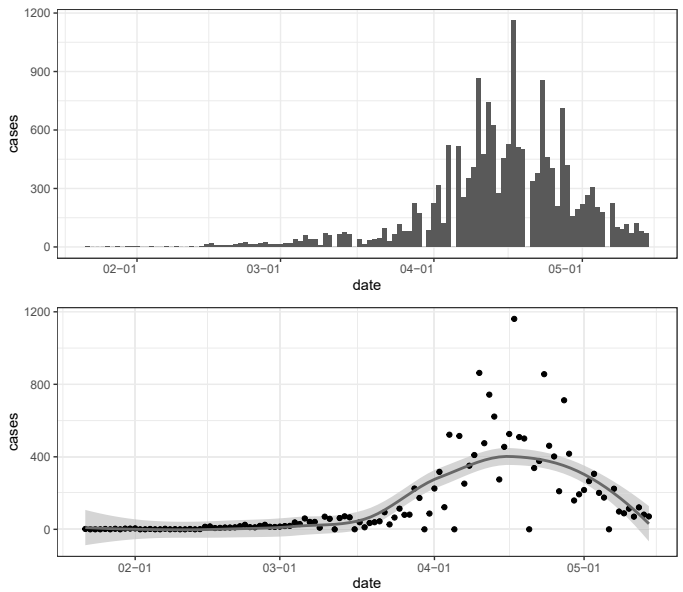


図12　日本の１日あたりの確認感染者数　2020年５月15日時点

注．下の図は同じデータを点(geom_point()関数)で表現し，近似曲線のレイヤーを重ねたものである．

データの操作は基本的に次の５つにまとめられますが，第１の操作はすでに紹介しました．

- 行の取り出し＝観察値から特定の観察値を取り出す　filter( )
- 並びかえる　arrange( )
- 列の取り出し＝特定の変数を選ぶ　select( )
- 新たな変数を作る　mutate( )
- 多数の値から単一の要約量を作る　summarize( )

こうしたデータ変換を学ぶために世界の感染者数の国別ランキングデータを作成し，トップ25か国について図14のような横棒のグラフを作成することを考えてみましょう．作業は次のようになります，

１）type変数の中のConfirmedの行を取り出す
２）国ごとにグループ化する　group_by( )
３）感染者数を集計する　summarize( )
４）大きい順に並べる　arrange( )
５）感染者数上位25か国を選択する　head( )
６）上位25か国の感染者数をグラフにする

それでは続けてスクリプト３に次のように入力し，実行[Run]してみて下さい．これで１)〜５)の作業が実行されます．

```
covid %>%
filter(type == “confirmed”) %>%
group_by(Country.Region) %>%
summarise(total = sum(cases)) %>%
arrange(-total) %>%
head(25)
```

このスクリプトでは“%>%”が活躍します．これはパイプと呼ばれ，操作を連結するものです．日本語で言えば「そして」です．このおかげで，たとえば，filter( )の中に操作対象のオブジェクト名(データフレーム名)covidを記す手間が省かれます[10]．パイプを使う場合の表現は次のように

[10] この点は本節で紹介されるデータの操作方法のすべてに妥当します．%>%がないと，オブジェクトがみつからないというエラーメッセージが返ってきます．

なります，

```
データフレーム　%>%　filter(条件)
```

このスクリプトは次のような操作を行っています．

- １行目と２行目はcovidとfilter( )がパイプ(%>%)で連結されていますので，データフレームcovidにfilter(type == “confirmed”)の操作を行いなさい，という命令です．
- ２行目から３行目にかけてもパイプで繋がれていますので，さらに，フィルタをかけた結果にgroup_by( )の処理をしなさい，という命令です．group_by( )はカッコ内の変数によってグループ化する関数です．これで国ごとにグループ化されます．
- ４行目のsummarize( )は要約関数です．ここではそのうちのsum( )関数を使って集計値を求めています．sum( ) の( )の中にcases変数を指定していますのでcasesを集計しています．ただし，この場合，すでにcases変数の中身はfilter( )によって感染者confirmedだけになっていますので，感染者数を集計することになります．そしてsum( )で集計した感染者数confirmedを = でtotalという変数を作成し，そこに容れています．totalは自由につけてかまいません．たとえば，syukei, gokeiでも何でもOKです．
- さらに，パイプでつながっていますので，ここまでの結果をarrange( )関数に渡し，( )の中の変数totalの大きさにしたがって並び替えを行っています．arrange( )はデフォルトでは昇順で並び替えた結果を返します．そこで変数totalの前にマイナス記号をつけ，降順で並び替えるようにしています．
- 最後の，head()によって先頭の25行を取り出しています．

スクリプトを実行すると，コンソール画面に図13のように表示されます．この結果を保存しておきましょう．代入演算子<-を使ってtop25と名付けたオブジェクトに容れておきます．top25には一連の操作の結果つまり図13に示された25行×２列のデータが格納されます．

```
top25 <- covid %>%
  filter(type == “confirmed”) %>%
  group_by(Country.Region) %>%
  summarise(total = sum(cases)) %>%
  arrange(-total) %>%
  head(25)
```

```
> covid %>%
+   filter(type == "confirmed") %>%
+   group_by(Country.Region) %>%
+   summarise(total = sum(cases)) %>%
+   arrange(-total) %>%
+   head(25)
# A tibble: 25 x 2
   Country.Region  total
   <chr>           <dbl>
 1 US             243453
 2 Italy          115242
 3 Spain          112065
 4 Germany         84794
 5 China           82432
 6 France          59929
 7 Iran            50468
 8 United Kingdom  34173
 9 Switzerland     18827
10 Turkey          18135
# … with 15 more rows
```

図13　各国の感染者数
注．2020年４月４日現在．

最後に，６)の作業―図13のテーブルのグラフ化―を行いましょう．ggplot2によるグラフ作成方法はほぼ同一です．ここでデータを表現するために横棒グラフを使います．棒グラフはgeom_col( )を利用します．geom_col(aes(x=x軸の変数,y=y軸の変数))です．

```
ggplot(top25)+                                  ☜データフレームを指定するレイヤー
geom_col(aes(Country.Region, total))+           ☜棒グラフ作成レイヤー
coord_flip()+                                   ☜棒グラフを横に描くためのレイヤー
theme_bw()                                      ☜テーマを指示するレイヤー
```

coord_flip( )は棒グラフの図を横にする操作(座標変換)です．以上で，図14のような横棒のグラフが描かれます．しかし，これだけですと，大きい順に描かれません．図14の左側の図になるだけです．そこでさらに，x軸の表記方法に感染者数の集計値の大きい順に並びかえます．これはreorder( )によって実現できます．コードはreorder(x軸変数，並び替えの基準となる変数つまりy軸変数名)です．

```
ggplot(top25)+
geom_col(aes(reorder(Country.Region,total), total))+
coord_flip()+
theme_bw()
```

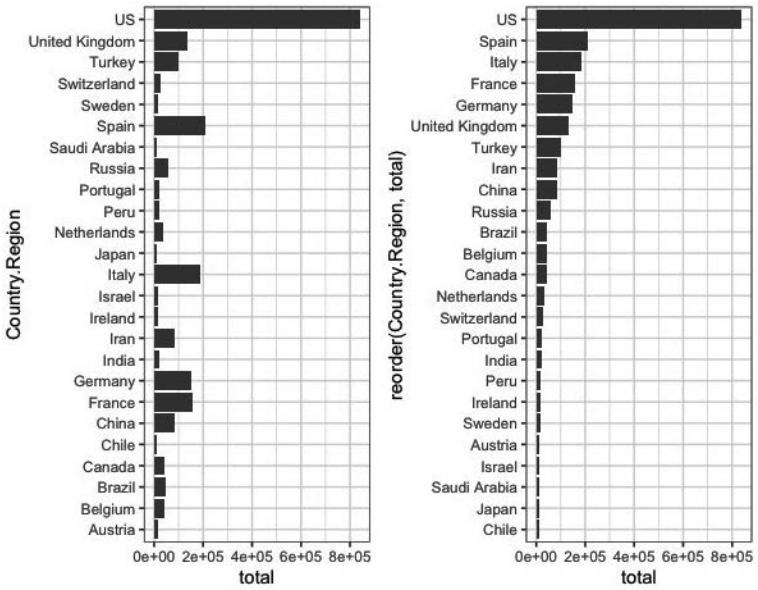


図14　各国の感染者数 reorder( )なしのケースとありのケース

注．データは2020年４月４日時点．

ここまで毎日の感染者数および感染者数の累計を計算し，それらをグラフ化してきました．ここでさらに累積的な増加を計算し，それをグラフ化して比較してみましょう．このために新たな変数を作成する関数mutate( )，必要な変数だけを選択するためにselect()を利用します．また変数名を変更するrename( )も使います．それではスクリプトに次のように入力して下さい．

```
cum_dead <- covid %>%
  rename(country = Country.Region) %>%
  group_by(country) %>%
  filter(type == “death”,country %in% c(“US”, “Italy”, “Spain”, “Germany”))%>%
  mutate(cum_d = cumsum(cases)) %>%
  select(country, date, cum_d)
```

２行目にrename( )関数を使い，変数名 Country.Regionをcountryという変数名に変更しています．この関数の表現式は，

```
rename(新しい変数名＝古い変数名)
```

3行目ではgroup_by( )を使って国別にグループ化しています．4行目はfilter( )関数ですが，抽出条件に2つ指定しています．第1に，type==“death”でtype変数の値がdeathのものを選択します．さらに2つ目の抽出条件が，country %in% c(“US”, “Italy”, “Spain”, “Germany”)です．これは，変数countryのうちc(“US”, “Italy”, “Spain”, “Germany”)の中のどれかに等しい行を選びなさい，を意味しています．論理演算で言えば論理和です．これで4か国が選ばれました．

```
x %in% y　xがyの中のどれかに等しい行(観察値)を選択
```

5行目はmutate( )を使って新しい変数を作成しています．新しい変数名は自分の好きなようにつけてOKです．ここではcumsum( )を使ってcasesの累積を計算し，その変数名をcum_dとしています．mutate( )の表現は次のようになります，

```
mutate(新しい変数名＝計算式)
```

6行目ではselect( )を利用し，3つの変数を取り出しています．

```
select(取り出す変数名)
```

最後に，第1行目にもどると，以上の操作結果をcum_deadというオブジェクトに格納しています．View()でcum_deadの中をみると，4か国について3つ変数が表示されます．国別の死者の累積数をグラフでみてましょう．次のように入力して下さい．

```
ggplot(cum_dead)+                          ☜データフフレームを指定するレイヤー
geom_line(aes(date,cum_d, lty = country))+ ☜データを折線で表現するレイヤー
theme_bw()                                 ☜白黒テーマを指定するレイヤー
```

2番目のレイヤーの中に“lty = country”を入れています．これは国をlytすなわち線種line typeで区別して表現させるコードです．

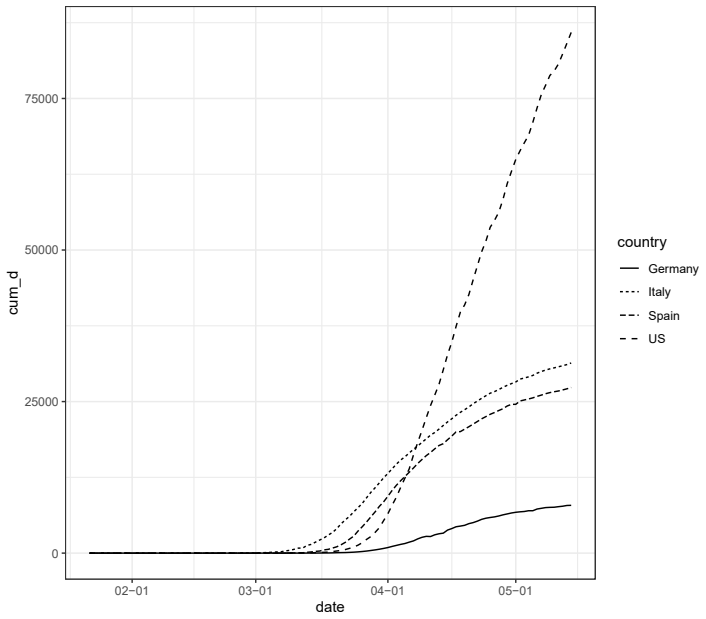


図15　主要国の死亡者数の推移(累積数)　2020年５月15日時点

Ⅳ．２．経済活動とCOVID-19

図16は，情報通信技術(ICT)財のサプライチェーンを示しています．この図から容易に理解されるように，中国経済は世界の工場であり，グローバルネットワークの中心に位置します．中国での製造業生産の停滞はほぼすべての国の製造業にサプライショックをもたらします(Baldwin and di Mauro, 2020)．

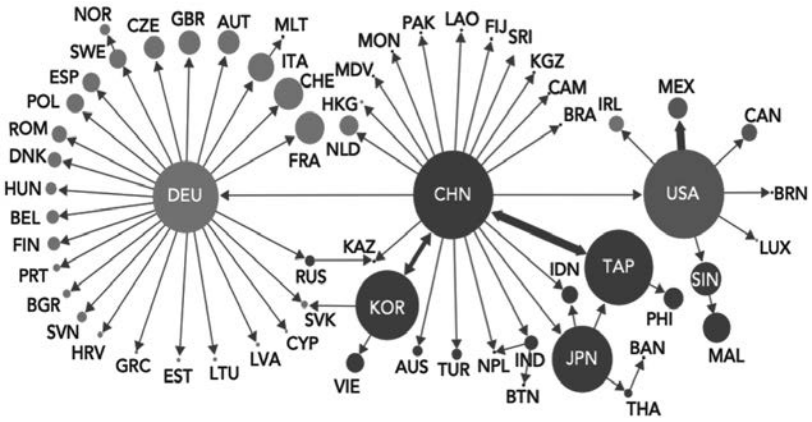


図16　ICTセクターにおけるサプライチェーン

出所：Global Value Chain Development Report (2019) p. 29.
注．円の大きさはその国の付加価値貿易，円を結びつける線の太さは貿易パートナー間の付加価値フローの大きさを示す．

経済活動のグローバル化は同時に人々のグローバルな移動も意味します．また，このサプライチェーンには地域的な性格―ハブの存在―も鮮明に現れています．こうしたつながりをみると，当初，中国で発生した新型コロナウィルスが韓国，日本において急増したことは当然のことかもしれません．ヨーロッパではドイツがネットワークのハブです．イタリア，フランスおよびイギリスが新型ウィルスの猛威にさらされているのも理解できます．また，もう１つの大きなハブはアメリカです．アメリカで感染者数は中国を超え(図14)，また死者も急増しています(図15)．

本節ではいくつかの経済活動指標とCOVID-19の関連を見ていきます．そのために，最初に，世界銀行のデータを取得するRパッケージWDIの利用法を説明します．じっさいのデータ分析にあたっては複数のデータセット(データフレーム)が必要になることが多々あります．そこで第２に，COVID-19データセットと世銀データセットの結合方法を解説します．

Ⅳ．２．１．世界銀行World Development Indicator

世界銀行によって数多くの経済指標が提供されています．R にはそうした指標を簡単に取得できるパッケージWDIがあります．それではこれまでと同じように，このパッケージを組み込みます．コンソールに次のように入力して下さい，

```
> install.packages(“WDI”)
```

利用するためにはスクリプト画面にlibrary( )で呼び出します．

```
library(WDI)
```

どの国のデータ，もしくはどのようなデータシリーズが利用可能かを見るためには次のように入力します．

```
> WDI_data$country      もしくはView( WDI_data$country)
> WDI_data$series       もしくはView(WDI_data$series)
```

WDI_data$countryによってWDIで利用可能な国の一覧が表示されます．また国別コード，緯度経度情報などが提供されています．WDI_data$seriesによって利用可能なデータ系列名が表示されます．indicatorが変数名，nameが変数の簡単な説明です．変数の詳しい内容を知るためには，世銀のサイト http://datatopics.worldbank.org/world-development-indicators/#archivesをチェック

して下さい．

それでは具体的にデータを取得してみましょう．次のようにスクリプトに入力し，実行[Run]を押してみて下さい．ここでは「対GDP比貿易額(%)」を世銀からダウンロードします．このデータ名は“NE.TRD.GNFS.ZS”です．そしてそれをtradeという名のオブジェクトに入れます．

```
trade <-WDI(country = c('USA','DEU','ESP','KOR','ITA','FRA'), indicator = "NE.TRD.GNFS.ZS", start = 1980,end = 2017)
```

WDI関数の使い方は次のようになります．

```
WDI(country = "all", indicator = "NY.GNS.ICTR.GN.ZS", start = NULL, end = NULL, extra = FALSE, cache = NULL)
```

1．country=“国名”．country=“all”とすると，すべての利用可能な国のデータが取得されます．また，国名はisoの２文字(もしくは３文字)コードを利用します．たとえば，日本ならばJP(JPN)です．
2．indicator=“世銀データの指標の名前”．例では“NY.GNS.ICTR.GN.ZS”．
3．start=開始年，end終了年です．start=NULLの場合，1950年です．end=NULLの場合，現在の年になります。
4．extra = TRUEの場合，地域region，国コードiso3c，所得水準といった追加変数を返します．
5．cache ＝NULL(オプション)．WDIcache( )によって作成されたWDI indicatorの更新可能な指標リストです．
6．なお，複数の国，指標を取り出したい場合は，例のように，c( )を使います．

図17は対GDP比貿易額(%)を主要な感染国についてプロットしたものです。アメリカと日本を除けば，ほとんどの国で1990年以降急速に上昇してきたことが理解できます。貿易の大きさがGDPの半分を占めるようになっています。とりわけ，ドイツと韓国はGDPの70%を超えており，グローバル化がきわめて進んだ国だということが理解できます。

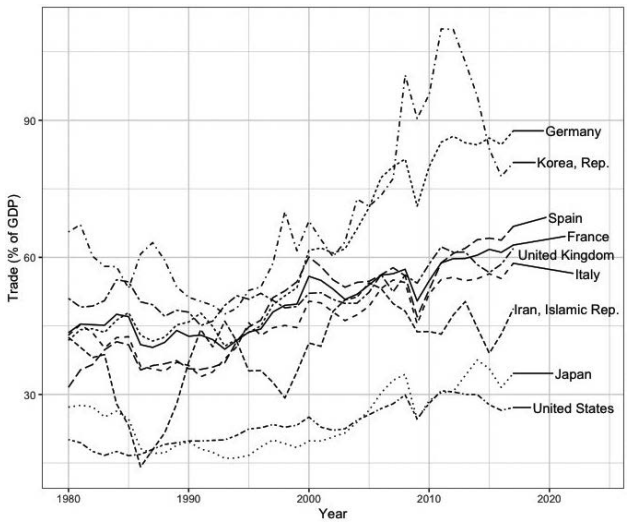


図17　主要感染国の対GDP貿易額(%)2018年.

これだけグローバル化が進んでいれば，人の移動も活発に行われることになります．いったん，貿易ネットワークのハブでCOVID-19のような感染症が発生すれば，瞬く間に世界中に拡大して行くことが容易に想像できます．なお，一連の操作はscript4というスクリプト名で保存します。

Ⅳ．2．2．経済のグローバル化とCOVID-19 – 回帰分析lm ( )

次に，経済活動のグローバル化を踏まえ，感染者数がグローバリゼーションの関数だというアイデアを検討してみましょう。グローバリゼーションはモノ・サービスのグローバル化，ヒトのグローバル化，そして生産のグローバル化にブレークダウンできます[11]．第１のモノ・サービスのグローバル化は対GDP貿易(%)によって代理させます。第２のヒトのグローバル化はインバウンド訪問者数によって表現します。第３の生産のグローバル化は国外直接投資によって代理されると仮定します．

ここで検証するモデルは，感染者数を対GDP貿易(%)，インバウンドの訪問者数および国外直接投資に回帰させるモデルです。理屈なしの直感的モデルですが，ここではRで回帰分析[12]を行う練習だと理解して下さい。こうした仮説の検討のために，最初に，世銀WDIから以下の３つの変数をダウンロードし，それぞれwd_trade, wd_fdi, wd_inbという名前のデータフレームに容れておきます。取得年数は，現在のグローバリゼーションの程度を知るために，直近の2018年もしくは

[11] ここでは金融のグローバル化は省略します。
[12] 回帰分析については上藤一郎先生が訳された『数式なしでわかるデータサイエンス ビッグデータ時代に必要なデータリテラシー』の第６章を参照して下さい。

2017年を指定します。

変数名	NE.TRD.GNFS.ZS	対GDP貿易額 (% of GDP)
変数名	BX.KLT.DINV.WD.GD.ZS	国外直接投資,純流入 (% of GDP)
変数名	ST.INT.ARVL	ツーリズム，訪問者数

サンプル国は感染者数トップ100の国とします。そこで最初に以下のスクリプトで感染者トップの国を選択しています。

```
top100 <- covid %>%
  filter(type == “confirmed”) %>%
  group_by(Country.Region) %>%
  summarise(total = sum(cases)) %>%
  arrange(-total) %>%
  rename(country = Country.Region) %>%          ☜変数名を変更
  filter(country != “Diamond Princess”)%>%      ☜ダイヤモンド・プリンセス号を除外
  head(100)
```

これでtop100という名前のデータフレームの中に感染者の数で見てトップ100に入る国のデータが格納されます。このスクリプトにおいてダイヤモンド・プリンセス号のケースを除外するために，“!=”を利用していますが，これは論理演算子と呼ばれるもので「等しくない」を意味します。他に，“==”は「等しい」，“>=”は「以上」，“<=”は「以下」を意味します。

次に，世銀からグローバリゼーションに関係するデータを取得しますが，その場合“country=”に国名もしくはiso2コードが必要なります。しかし，“country=”の後に100か国の国名を入力するのはとても面倒です。そこで国名・国コード変換パッケージcountrycode[13]を利用し，top100の中の国名をiso2(国名を２文字の英語表記)に変換します。そしてそれをiso2と名付けたオブジェクトに入れます。

```
iso2   <- countrycode(top100[[1]], origin = ‘country.name’,destination = ‘iso2c’)
```

[13] 本文では省略してますが，install.packages(“countrycode”)でパッケージをインストールし，library(countrycode)で呼び出しておいて下さい。

なお，countrycode( )の書式は次のようになります。

```
countrycode(国コードもしくは国名，origin=変換前のコードスキーム，destination=変換後のコードスキーム )
```

最初の引数にはtop100[[1]]と入力していますが，これはデータフレームtop100の第１列を選択することを意味します。ここには国名―Japan, China等など100か国の国名―が入っています。ですので，変換前のコードスキームは国名ですからorigin = ‘country.name’と指定しています。そしてこれをdestination = ‘iso2c’で英語２文字の国表現コードに変換するよう指示しています。

それではこのiso2を利用し，感染者トップ100か国のグローバリゼーション代理変数を取得します。

```
wd_trade <-  WDI(country = iso2, indicator = “NE.TRD.GNFS.ZS”, start = 2018, end = 2018,extra = F, cache = NULL) %>% rename(trade=NE.TRD.GNFS.ZS)
wd_fdi <-  WDI(country = iso2, indicator = “BX.KLT.DINV.WD.GD.ZS”, start = 2017, end = 2017,extra = F, cache = NULL) %>%  rename(fdi=BX.KLT.DINV.WD.GD.ZS)
wd_inb <-  WDI(country = iso2, indicator = “ST.INT.ARVL”, start = 2018, end = 2018,extra = F, cache = NULL) %>%
rename(inbound= ST.INT.ARVL)
```

あとはこの３つのデータフレームと感染者データの入ったtop100を結合させるだけです。それぞれ変数countryが入っていますので，これをキーに結合させるとしましょう．しかし，１つだけ厄介なことにtop100の中のアメリカの国名だけが“US”で表記されています[14]．アメリカだけがcountryをキーで結合できなくなります．そこでデータフレームtop100の１行，１列目に入っている“US”を“United States”に入れ替えます．

```
top100[1, 1] <- “United States”
```

これでcountryをキーに４つのデータフレームを結合できます。

[14] JH CSEESではアメリカの国名がUSで表現されていますが，World BankではUnited Statesで表記されています。このために発生する問題です。

```
world <- top100 %>%
arrange(country) %>%                          ☜countryの順で並べる
left_join(wd_fdi) %>%
left_join(wd_inb,by="country") %>%            ☜左側からbyによってデータフレームを結合
left_join(wd_trade,by="country")%>%
select(iso2c.x, country, total, fdi, inbound, trade)      ☜select( )によって必要な変数のみ選択
```

以上で回帰分析を行うのに必要なデータセットworldができあがりました。回帰分析のためにはlm( )関数を使います。書式は次のとおりです。

```
lm(目的変数〜説明変数，データセット名)
```

これにしたがって，各変数を入力してください。ただし，解釈を容易にするために変数はlog( )によって自然対数に変換してあります。その上で，回帰分析の結果を代入演算子<-を使ってres.lmというオブジェクトに容れます。説明変数が複数の場合(重回帰分析の場合)，＋記号を使って追加します。

```
res.lm   <- lm(log(total)〜log(trade)+log(inbound)+log(fdi),world)
```

コンソール画面にres.lmを入力すると，シンプルな結果が返ってきます。これはあまり役に立ちませんので，summary(オブジェクト名)関数を使います。

```
summary(res.lm)
```

結果は図18のように表示されます。最初に係数(Estimateの列)をみると，貿易tradeも直接投資fdiもマイナスの符号をとっています．これは貿易の拡大も直接投資の伸びも感染者数にマイナスの効果を与えるという予想に反した結果を示しています．もっともP値(Prの列)をみると，その係数がゼロであるという帰無仮説は棄却できないようです．

```
> summary(res.lm)

Call:
lm(formula = log(total) ~ log(trade) + log(inbound) + log(fdi),
    data = world)

Residuals:
     Min       1Q   Median       3Q      Max
-1.93534 -1.02048 -0.01379  0.81492  2.58708

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.70112    2.26241  -1.636   0.1063
log(trade)   -0.64511    0.30651  -2.105   0.0389 *
log(inbound)  0.86192    0.11018   7.823 3.88e-11 ***
log(fdi)     -0.02336    0.13067  -0.179   0.8587
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.197 on 70 degrees of freedom
Multiple R-squared:  0.498,     Adjusted R-squared:  0.4765
F-statistic: 23.15 on 3 and 70 DF,  p-value: 1.613e-10
```

図18　回帰分析の結果

注．回帰分析にあたっては中国は除いてある．

唯一，予想どおりの符号を示し，かつ統計的に有意な結果を示したのは，インバウンド変数inboundです。インバウンドの１ポイント上昇が感染者数を0.86ポイント上昇させるという，非常に大きな効果を有していることが分かります．COVID-19が人の活動を介在して拡大して行くことを考えれば，こうした結果は驚くにあたらない結果です．

こうしたナイーブな回帰分析の結果を踏まえ，図19は感染者数とインバウンド者数の散布図を描いています．両者の関係は正の相関を示します．グラフの右端には感染者が急増しているアメリカ(US)，イタリア(IT)，スペイン(ES)，フランス(FR)，ドイツ(DE)およびイギリス(GB)を見ることができます。これはインバウンド業界には厳しい結果かもしれません[15]．

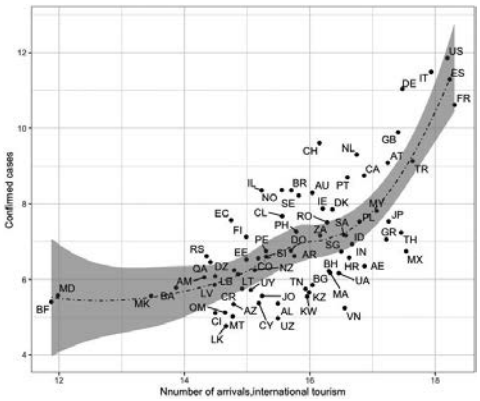


図19　感染者数とインバウンド者数の関連

注．データは対数表示．感染者(Confirmed cases)直近のデータ，インバウンドは2018年データ．

[15] なお，Ⅳ．２．２で行ったデータ処理は付録の【スクリプト一覧】においてscript5という名前でまとめられています．

ジョン・ホプキンス大学システムサイエンス・エンジニアリングセンター(JHU CSSE)が提供するCOVID-19データは常時更新されています。本資料で利用されたデータはすでに古いものとなっていますので，RおよびRStudioを使って最新のデータを分析することをお勧めします。

【参考文献・URL】

Baldwin, R. and di Mauro, B.W.(eds.) Economics in the Time of Covid-19, A CEPR Press VoxEU.org eBook. 2020.

Chang, W.(石井弓美子，河内崇，瀬戸山雅人，古畠敦訳)『Rグラフィックスクックブック―ggplot2によるグラフ作成のレシピ集』オライリー・ジャパン，2013年.

Global Value Chain Development Report, 2019.

Kabcoff,R.I. (2011) *R in Action: Data analysis and graphics with R,* Manning.

Ng. A. and Soo, K.(上藤一郎訳)『数式なしでわかるデータサイエンス ビッグデータ時代に必要なデータリテラシー』オーム社，2019年.

Tianzhi Wu , Erqiang Hu , Xijin Ge , Guangchuang Yu (2020) Open-source analytics tools for studying the COVID-19 coronavirus outbreak doi: https://doi.org/10. 1101/2020. 02. 25. 20027433.

Wickham, H. and Grolemund, G. (黒川利明訳，大橋真也技術監修)『Rではじめるデータサイエンス」オライリー・ジャパン，2017年．本書の英語版がhttps://r4ds.had.co.nzで公開されている.

Zhao, Y. (2020) COVID-19 Data Analysis with R - China, Coronavirus data analysis – China, http://www.rdatamining.com/docs/Coronavirus-data-analysis-china.pdf

Zhao, Y. (2020) COVID-19 Data Analysis with R - Worldwide, http://www.rdatamining.com/docs/Coronavirus-data-analysis-china.pdf

Visualize the Pandemic with R #COVID-19(https://towardsdatascience.com/visualize-the-pandemic-with-r-covid-19-c3443de3b4e4)

Map Visualization of COVID-19 Across the World with R(https://datascienceplus.com/map-visualization-of-covid19-across-world/)

ggplot2パッケージ ggplot2(https://ggplot2.tidyverse.org/index.html)

【データ・ソース】

- COVID-19(新型コロナウィルス)のデータ
  - ジョン・ホプキンス大学システムサイエンス・エンジニアリングセンター(JHU CSSE)

のGitHubサイト (https://github.com/CSSEGISandData/COVID-19)

- 以下のデータセットのソースはすべてJHU CSSEである.
  - nCov2019パッケージ (https://guangchuangyu.github.io/nCov2019)
  - Coronavirus COVID-19 outbreak statistics and forecast v0. 82 (http://www.bcloud.org/e/)
  - Rami Kripin　GitHubサイト (https://github.com/RamiKrispin/coronavirus-csv)
- 世界銀行World Development Indicator (http://datatopics.worldbank.org/world-development-indicators/#archives).

【Rパッケージ一覧】

ggplot2, tidyverse, dplyr, readr, maps, viridis, patchwork, WDI, nCov2019, coronavirus, countrycoe, ggrepel

【付録１　スクリプト一覧】

```
# ==================================================================
#   script1　練習
# ==================================================================
    age <- c(1, 3, 5, 2, 11, 9, 12, 3)
    weight <- c(4. 4, 5. 3, 7. 2, 5. 2, 8. 5, 7. 3, 6, 10. 4)
    mean(weight)                                          # weightの平均を計算
    sd(weight)
    cor(age, weight)
    name <- c(“Jack”, “Robert”, “David”, “Nancy”, “Wendy”, “Peter”, “Sam”, “Naomi”)
    child <-data.frame(name, age, weight)
    plot(age, weight)                                     # 散布図を描く
# ==================================================================
#   script2　グラフを描く
# ==================================================================
    library(ggplot2)
      ggplot(cars)+
      geom_point(aes(speed, dist))+
      labs(x=“Speed”, y=“Distance”, title = “Speed and Distance”)+
      geom_smooth(aes(speed, dist))+
```

```
        annotate(“text”, x=10, y=100, label=“A simple graph”)+
        theme_bw()
# ================================================================
#   script3   COVID-19のデータの読み込み＆グラフ化
# ================================================================
library(readr)
library(tidyverse)
#   COVID-19のデータの読み込み
        covid<-read_csv(“Desktop/coronavirus_dataset.csv”)
      head(covid)
      View(covid)
#   日本＆感染者数を抽出
        jp <- filter(covid, Country.Region == “Japan”, case == “Confirmed”)
#   日本の感染者数推移のグラフ
        ggplot(jp)+
        geom_col(aes(date,cases))   +
        theme_bw()
#   世界の感染者数トップ25
        top25 <- covid %>%
        filter(type == “confirmed”) %>%
        group_by(Country.Region) %>%
        summarise(total = sum(cases)) %>%
        arrange(-total) %>%
        head(25)
#   世界の感染者数トップ25のグラフ化
        ggplot(top25)+
        geom_col(aes(reorder(Country.Region,total), total))+
        coord_flip()+
        theme_bw()
#   主要感染国の死亡者数(累積)
        cum_dead <- covid %>%
        rename(country = Country.Region) %>%
```

```
        group_by(country) %>%
        filter(type == “death”,country %in% c(“US”, “Italy”, “Spain”, “Germany”))%>%
        mutate(cum_d = cumsum(cases)) %>%
        select(country, date, cum_d)
# 死亡者数の推移(累積)
        ggplot(cum_dead)+
        geom_line(aes(date,cum_d, lty = country))+
        theme_bw()
# ====================================================================
#   script4   世銀データの読み込み & グラフ
# ====================================================================
library(WDI)
library(ggplot2)
#   WDIデータの表示
        View( WDI_data$country)
        View(WDI_data$series)
#   WDIデータの読み込み
        trade <-WDI(country = c(‘USA’,‘DEU’,‘ESP’,‘KOR’,‘ITA’,‘FRA’), indicator = “NE.TRD.GNFS.
        ZS”,start = 1980,end = 2017)
# 読み込んだデータのグラフ表示
        ggplot(trade)+
        geom_line(aes(year,NE.TRD.GNFS.ZS , lty=country))  +
        xlab(‘Year’) + ylab(‘Trade (% of GDP)’)+
        theme_bw()
# ====================================================================
#   script5   データフレームの結合，回帰分析
# ====================================================================
# サンプルの感染者トップ100か国を抽出
        top100 <- covid %>%
        filter(type == “confirmed”) %>%
        group_by(Country.Region) %>%
        summarise(total = sum(cases)) %>%
```

```
        arrange(-total) %>%
        rename(country = Country.Region) %>%
            filter(country != “Diamond Princess”)%>%
    head(100)
# データフレームtop100の中の国名をiso2に変換
        iso2   <- countrycode(top100[[1]], origin = ‘country.name’,destination = ‘iso2c’)
# WDIからtop100の国のデータのみをダウンロード
        wd_trade <-     WDI(country = iso2, indicator = “NE.TRD.GNFS.ZS”, start = 2018, end =
        2018,extra = F, cache = NULL) %>% rename(trade=NE.TRD.GNFS.ZS)
        wd_fdi <-   WDI(country = iso2, indicator = “BX.KLT.DINV.WD.GD.ZS”, start = 2017, end =
        2017,extra = F, cache = NULL) %>%   rename(fdi=BX.KLT.DINV.WD.GD.ZS)
        wd_inb <-   WDI(country = iso2, indicator = “ST.INT.ARVL”, start = 2018, end = 2018,extra
        = F, cache = NULL) %>%
    rename(inbound= ST.INT.ARVL)
# アメリカ表記のみ変更
      top100[1, 1] <- “United States”
# ４つのデータフレームを結合し，worldに格納
        world <- top100 %>%
        arrange(country) %>%
        left_join(wd_fdi) %>%
        left_join(wd_inb,by=“country”) %>%
        left_join(wd_trade,by=“country”)%>%
        select(iso2c.x, country, total, fdi, inbound, trade)
# 回帰分析
        res.lm   <- lm(log(total)〜log(trade)+log(inbound)+log(fdi),world)
        summary(res.lm)
# 感染者数とインバウンド者数の散布図作成
        ggplot(world,aes(log(inbound), log(total)))+
        geom_point()+
        theme_bw()+
        stat_smooth(method=loess,se=T,color=“black”,linetype=4,size=0. 5)+
        geom_text_repel(aes(label=iso2c.x),segment.alpha = 10)+
```

```
labs(
  x="Nnumber of arrivals,international tourism",
  y="Confirmed cases"
)
```

【付録２　COVID-19分析パッケージ】

１．tidycovid19

Joachim Gassen氏によって開発されたtidycovid19は，covid-19データをダウンロードし，整理し，ビジュアル化するために有益なパッケージです(https://github.com/joachim-gassen/tidycovid19)。

⑴　インストール方法

１）パッケージremotes

GitHub等に保存されたパッケージをインストールするためにはremotesをインストールしておく必要があります。一度，組み込むだけですので，コンソール画面で実行しても良いでしょう。コンソール画面に次のように入力し，１行入力ごとにエンターキーを押して下さい。

```
install.pacakages("remotes")
```

２）パッケージtidycovid19のインストール

```
remotes::install_github("joachim-gassen/tidycovid19")
```

３）関数

このパッケージには，現在，５つの関数が利用されています。

１．download_jhu_csse_covid19_data()

JHU CSSEからデータをダウンロードし，整理する関数．データは国レベルで集計される．

２．download_acpas_npi_dataz()

the Assessment Capcities Project (ACAPS)によって提供されている政府の拡散防止政策をダウンロードし，整理する関数.

３．download_google_trends_date()

用語"coronavirus"に関するサーチ量に関するGoogleトレンドデータをダウンロードし，整理する関数．本データはCovid-19に対する社会の関心を評価するために利用可能.

４．download_wbank_data()

wbstatsパッケージを利用した世界銀行のデータをダウンロードし，整理する関数．コロナウィルスとマクロ経済的変数の関連を評価するのに利用可能.

５．download_merged_data()

すべてのデータソースをダウンロードし，データを結合し，国・日のパネルサンプルを作成.

２．nCov2019

Wu, Hu, Tun, Ge, and Yuによって開発されたnCov2019という名前のRパッケージがあります(https://guangchuangyu.github.io/nCov2019)。これも新型コロナ・ウィルスの現状を理解する上で有益なパッケージです。

⑴　パッケージnCovRのインストール

スクリプト画面に次のように入力して下さい。＃以下のコメントは不要です。

```
library(remotes)　#remotes呼び出す
remotes::install_github("GuangchuangYu/nCov2019", dependencies = TRUE)
                                        ＃GitHubからパッケージをインストールする
```

⑵　nCovRパッケージの利用方法

```
library(nCov2019)
```

最新のデータを取得するためには，get_nCov2019()と入力するだけです。次のように入力し，取得したデータをオブジェクトxに格納します。

```
x<-get_nCov2019(lang="en")
```

lang="en"は使用言語を英語にする設定です。オブジェクト名xをコンソール画面に入力すると，中国のデータの感染者データの情報が表示されます。グローバルデータを取得するにはx["global"]と入力するだけです。以下の例では所得したグローバルデータをglobalと名づけたオブジェクトの中に容れています。

```
global<-x["global"]
```

以下の図はhead()を使って取得した新型コロナウィルスデータの冒頭部分を表示させた結果です。

```
> global<-x["global"]
> head(global)
           name confirm suspect  dead deadRate showRate  heal healRate showHeal
1         China   82360     174  3306     4.01    FALSE 75601    91.79     TRUE
2 United States  124676       0  2196     1.76    FALSE  1095     0.88    FALSE
3         Italy   92472       0 10023    10.84    FALSE 12384    13.39    FALSE
4         Spain   73235       0  5982     8.17    FALSE 12285    16.77    FALSE
5       Germany   57695       0   433     0.75    FALSE  8481     14.7    FALSE
6        France   38105       0  2317     6.08    FALSE  5724    15.02    FALSE
>
```

name変数は国名，confirmは確認された感染者数，dead変数は死亡者数，deadRate変数は致死率です。heal変数は治癒者数，healRate変数は治癒率です。