目次

第3章

GDP の成長と格差

3.1 概要

あなたは次のような記述をどのように理解するだろうか。

> 「GDP が伸び悩む中で，X 産業の付加価値生産額は年率 10% で成長している。国際競争力を高めつつある X 産業に重点的に補助金を配分することで，GDP 低迷を脱却することができるだろう」

日本経済の救世主とみなすべきか，あるいは業界団体による陳腐なポジショントークと見るか[*1]。おそらく唯一の正しい理解は，「情報が不足しているので判断できない」というものだ。この章で学ぶのはそういう話だ。

マクロ経済学は一国や地域の経済を総体として扱う経済分析である。分析対象の性質上，分析の中心的なターゲットは「合計」や「平均」といった集計量になる。マクロ経済学が分析対象とする集計量の中で，現在のところもっとも重要な概念は GDP であろう。GDP（国内総生産）は企業や政府によって生産された財・サービスを重複なく合算したものである。GDP は経済全体の生産能力を表すとともに居住者の総所得を表して

[*1] 「ポジショントーク」という言葉は否定的に響いてしまうかもしれないが，批判する意図はまったくない。嘘をついてはいけないが，自らのポジションが有利になるような事実を強調することは誰もが日常的に行っている。結局は，情報の受け手のリテラシーが重要だ。

おり，居住者の平均的な厚生（暮らし向き）と強い相関を持っている[2]。

GDP は生産された価値の合計であるから，合計する前の GDP を構成する細部のデータ，例えば産業ごとに計算した付加価値生産額など，も利用することができるということに注意しておこう。年率 10% で成長中の前述の X 産業は，GDP に対するシェアが 0.2% であるとしよう。GDP を 500 兆円とすれば，この産業は付加価値生産額 1 兆円の小さな産業である。補助金の投入によって産業の成長率を 10% → 11% に加速することができたとして，GDP 全体の成長に対する貢献がどの程度になるかを計算してみよう。補助金を投入しなければ 1.1 兆円を次年度に生産していたところ，補助金の投入によって 1.11 兆円に生産額が増える。増分は 0.01 兆円であるから，現在の GDP 全体に占める増分の比率は

$$0.01\text{兆円}/500\text{兆円} = 0.002\%$$

となる。この数字を大きいと見るか，小さいと見るかは意見の分かれるところだろう。しかし，短期的に（例えば，1 年とか 2 年で）GDP を拡大することに大きく貢献してくれるかといえば，答えはノーである。この産業は有望な成長産業であるかもしれないが，今はまだ小さすぎる。

とは言え，もし私が X 産業に従事していて，政府の補助金を獲得するための資料を作るように命じられたら，産業の成長力を強調する前述のようなレポートを書くだろう。ただし，それだけでは経済産業省の審査官や諮問委員は納得させられないだろうとも思う。短期的な GDP への貢献だけを見れば，全体から見て小さすぎるからだ。長期的なスパンで見たときに GDP の構成をどのように塗り替えていくかとか，あるいは関連産業に視野を広げて当該産業の影響の大きさをアピールするといった踏み込んだ分析をしなければ，魅力的な申請書にはならない。

私たちの多くは特定の産業を代弁する立場にもないし，申請書を審査する立場にもないが，このような話を理解しておく必要がある。政府の資源は有限であるし，財源は税金である。明らかに有効でない政策を実施する政府に対しては，投票行動やデモなどを

[2] 生産と所得の関係については，ひとまずのところは，生産された財・サービスを販売することで得た売上から給料が支払われるから，と考えておくとよい。

通じて反対意見を表明しなければならない。しかし，そのためにはまず数字を読めるようになることが大切だ。

理論パートの後半では，近年注目を集めている格差の指標について簡単に紹介する。GDP の成長は国民の平均的な暮らし向きを向上させるが，増えた所得がすべての国民に均等に行き渡る訳ではない。全体の拡大とともに個々の暮らし向きの変化を捉える指標を押さえておこう。

この講義では以下のことを学ぶ。

- 理論
  - 名目 GDP と実質 GDP，GDP デフレータ
  - 寄与度と寄与率
  - 格差の指標: 相対的貧困率，ジニ係数，トップ 1% の所得シェア
- プログラミング
  - 繰り返し計算（`for` ループ，`while` ループ）
  - 条件分岐（`if-elif-else`）

3.2 理論: GDP

3.2.1 名目 GDP

GDP（Gross Domestic Product, 国内総生産）は，ある期間において（通常は，1 年または四半期），各生産者が生産した付加価値をすべて合計したものである。付加価値とは，

$$\text{付加価値} = \text{生産物の価値} - \text{投入中間財の価値}$$

によって定義される。中間財というのは他の企業から購入した財・サービスであって，固定資本（資産）に分類されないものである。原料や材料のことと考えておけばよい。

生産された付加価値がどのように使われたかによって，

- 消費（Consumption）= 民間消費

- 投資（Investment）= 民間投資
- 政府購入（Government Purchases）
- 純輸出（Net Export）

に分類する。これはマクロ経済学や理論分析上の慣例であって，国民経済計算では通常，

- 消費 = 民間消費 + 政府消費
- 投資 = 民間投資 + 政府投資
- 純輸出

のように切り分けている。政府消費と政府投資を合算したものが政府購入である。数字を見るときには定義に注意しよう。

消費というのは，各期の経済厚生を表す概念である。家計部門の消費（民間消費）がその経済の構成員の厚生をもっともよく表すと考える場合には政府消費を切り離して民間消費を消費とみなす。政府から受けたサービス（役所が提供する無料または非常に低廉なサービス）も国民が享受していると考えれば，政府消費を含めて消費と扱う方が望ましい場合もある[3]。GDP 上は民間現実最終消費として記録されている。

投資というのは，生産設備を拡大する経済活動である。民間企業の生産設備を拡充したり，家計部門が持ち家（住宅サービスを供給する）を新築したりすることが含まれる。政府部門の投資には，道路や治水工事，庁舎の整備などが含まれる。いずれも，長期に渡って価値を生産する財・サービスの購入である。経済の生産キャパシティの拡大という意味では，政府投資を投資に含める方が理にかなっている。一方，民間部門の投資は経済の不況を示す重要な指標であり，民間投資の変化は独立した重要性を持っている。

純輸出は

$$\text{純輸出} = \text{輸出（Export）} - \text{輸入（Import）}$$

によって定義される。自国で生産した財やサービスを輸出した場合には，輸出額は GDP に含まれている。生産活動に輸入した価値が投入されている場合は，これを控除しなけ

[3] 例えば，とあるサービスに対して政府が 7 割の補助金を出したとしよう。これは，

ればならない。自国で生産された価値ではないものが含まれないようにするためだ。

マクロ経済学の標準的な教科書では,「生産 = 民間消費+民間投資+政府購入+純輸出」という分解が用いられている。これは政府部門が民間部門と異なる役割を担うからである。不況時には民間消費・民間投資は減少するが，政府は政府購入を拡大することで景気の大幅な後退を避けようとする。逆に好況で民間消費・民間投資が拡大する際には，政府購入を維持するか減少させることで，不況時にできた負債の解消を目指す。民間部門の経済活動と政府部門の経済活動は，通常，反対方向に動くのでこれらを単純に合算してしまうと分析上都合が悪い。

Y を GDP，C を（民間）消費，I を（民間）投資，G を政府購入，NX を純輸出とすると，次の関係式が成り立っている。

$$Y = C + I + G + NX$$

輸出を EX，輸入を IM と書くと，$NX = EX - IM$ であり，

$$Y = C + I + G + EX - IM$$

が成り立つ。これらは教科書的なマクロ経済分析においてもっともよく使われる GDP の加法分解であり，もっとも荒い分解である。

- 消費 C は一国の経済活動において通常もっとも大きな項目である。消費は個人の厚生（幸福度）を決定する重要な要因であるので，マクロ経済理論においても最重要の項目である。
- 投資 I（設備投資，在庫投資，住宅投資）は一国経済における生産のキャパシティを左右する項目である。
- 政府購入 G は政府の規模を表す項目である。政府購入の調整によって短期的な経済変動をコントロールすることができると考えられており，学部標準レベルのマクロ経済学では重要な項目になる。
- 純輸出 NX は外国経済との関わりを表す部分である。輸出 EX から輸入 IM を

控除したものである。

経済の生産規模を測る GDP ではあるが，米 800 万トンと自動車 70 万台を単純に足すことはできないので，貨幣の単位で測った価値を用いて生産規模を計算する。

$$\text{価値} = \text{価格} \times \text{数量}$$

なので，これをすべての最終財（国内生産物であって，民間・政府に消費または投資されるか，外国に輸出される財のこと）について合計し，輸入品の価値を控除したものが GDP になっている。価格として当期の価格を用いる GDP を**名目 GDP**（nominal GDP）という。

3.2.2　実質 GDP

名目 GDP の計算の基礎となる価値の式に，時間のインデックスを付記しておこう。

$$\begin{aligned} \text{価値}_t &= \text{価格}_t \times \text{数量}_t \\ \text{価値}_{t+1} &= \text{価格}_{t+1} \times \text{数量}_{t+1} \end{aligned}$$

特定の財について，生産物の価値の上昇は 2 つの原因によって起こる。つまり，価格が上昇するか，生産数量が拡大するかだ。

$$\begin{aligned} (\text{価値の粗変化率}) &= \frac{\text{価値}_{t+1}}{\text{価値}_t} = \frac{\text{価格}_{t+1}}{\text{価格}_t} \times \frac{\text{数量}_{t+1}}{\text{数量}_t} \\ &= (\text{価格の粗変化率}) \times (\text{数量の粗変化率}) \end{aligned}$$

私たちの豊かさ（より正確には，物質的な豊かさ material wealth）は，財やサービスをどれだけ消費・利用できるかによって決まる。その財をいくらで買ったかとは関係ないはずである[4]。

[4] 個別の財や特別な消費行動を説明する文脈では，価格が上がることで消費の効用が高まるような財（ヴェブレン財）が存在することは否定しない。しかし，そのような財は多くはないだろうし，経済全体で見れば小さな市場規模だろう。マクロ経済学はあくまでも総体的・平均的な行動を分析することが目標なので，ひとまず素朴な消費行動のみを考えておけばよい。

そこで，豊かさを測る指標として，成長率

$$\frac{\text{数量}_{t+1}}{\text{数量}_t} = \frac{\text{価値}_{t+1}/\text{価値}_t}{\text{価格}_{t+1}/\text{価格}_t}$$

を用いようというのが実質 GDP の考え方である。この数値が 1 より大きければ経済活動は拡大していて，1 より小さければ縮小している。1 であれば現状維持となる。多種多様な財・サービスが生産される経済において，数量変化率を計算するためには 第**??**章で扱ったような工夫が必要になる。

各最終財（$i = 1, \ldots, N$）について，t 期における生産量を並べたベクトルを $\boldsymbol{x}_t$ ，価格ベクトルを $\boldsymbol{p}_t$ とする。$\boldsymbol{x}_t$ の第 i 要素 $x_{t,i}$ は財 i の生産量，$\boldsymbol{p}_t$ の対応する要素 $p_{t,i}$ が財 i の当該期の平均価格を表している。t 期の名目 GDP,Y_t, は

$$Y_t = \text{名目 GDP}_t = \boldsymbol{p}_t \cdot \boldsymbol{x}_t$$

と書ける。$t-1$ 期から t 期にかけての経済成長率 g_t は

$$\text{（粗）経済成長率}_t = 1 + g_t = \frac{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_t}{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_{t-1}}$$

と定義する。ベクトルを数値化するベクトルとして過去の情報 $\boldsymbol{p}_{t-1}$ を用いているので，ラスパイレス指数（あるいは**ラスパイレス数量指数**）である。

参照年を 0 期とする。0 期においては

$$\text{実質 GDP}_0 = \text{名目 GDP}_0 \qquad \text{あるいは} \qquad y_0 = Y_0$$

が成り立つとする。比較年 t 期の**実質 GDP**（連鎖方式）は次のように定義される。

$$\begin{aligned} y_t &= \prod_{i=1}^{t} (1 + g_i) \times y_0 \\ &= (1 + g_t)(1 + g_{t-1}) \times \cdots \times (1 + g_1) \times y_0 \\ &= \frac{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_t}{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_{t-1}} \times \frac{\boldsymbol{p}_{t-2} \cdot \boldsymbol{x}_{t-1}}{\boldsymbol{p}_{t-2} \cdot \boldsymbol{x}_{t-2}} \times \cdots \times \frac{\boldsymbol{p}_0 \cdot \boldsymbol{x}_1}{\boldsymbol{p}_0 \cdot \boldsymbol{x}_0} \times y_0 \end{aligned} \tag{3.1}$$

すなわち，連続する 2 期の経済成長率をラスパイレス数量指数で計算し，参照年から累

積的に積算したものが実質 GDP である。

式 (3.1) において，0 期の実質 GDP が $y_0 = \boldsymbol{p}_0 \cdot \boldsymbol{x}_0$ であることに注意すれば，異なる表現が得られる。

$$\begin{aligned} y_t &= \frac{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_t}{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_{t-1}} \times \frac{\boldsymbol{p}_{t-2} \cdot \boldsymbol{x}_{t-1}}{\boldsymbol{p}_{t-2} \cdot \boldsymbol{x}_{t-2}} \times \cdots \times \frac{\boldsymbol{p}_0 \cdot \boldsymbol{x}_1}{\boldsymbol{p}_0 \cdot \boldsymbol{x}_0} \times \boldsymbol{p}_0 \cdot \boldsymbol{x}_0 \\ &= \boldsymbol{p}_t \cdot \boldsymbol{x}_t \times \frac{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_t}{\boldsymbol{p}_t \cdot \boldsymbol{x}_t} \times \frac{\boldsymbol{p}_{t-2} \cdot \boldsymbol{x}_{t-1}}{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_{t-1}} \times \cdots \times \frac{\boldsymbol{p}_0 \cdot \boldsymbol{x}_1}{\boldsymbol{p}_1 \cdot \boldsymbol{x}_1} \end{aligned}$$

右辺の最初の項に余分な項 $\boldsymbol{p}_t \cdot \boldsymbol{x}_t / \boldsymbol{p}_t \cdot \boldsymbol{x}_t$ を追加して，$\boldsymbol{p}_0 \cdot \boldsymbol{x}_0 / \boldsymbol{p}_0 \cdot \boldsymbol{x}_0$ を消去した。分母を 1 つずつずらしていることにも注意しよう。これは，さらに

$$y_t = \frac{\boldsymbol{p}_t \cdot \boldsymbol{x}_t}{\frac{\boldsymbol{p}_t \cdot \boldsymbol{x}_t}{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_t} \times \frac{\boldsymbol{p}_{t-1} \cdot \boldsymbol{x}_{t-1}}{\boldsymbol{p}_{t-2} \cdot \boldsymbol{x}_{t-1}} \times \cdots \times \frac{\boldsymbol{p}_1 \cdot \boldsymbol{x}_1}{\boldsymbol{p}_0 \cdot \boldsymbol{x}_1}}$$

と書き直すことができる。分母がパーシェ式連鎖指数となっていることに注意する。すなわち，

$$\begin{aligned} \mathrm{DFL}_0 &= 1 \\ \mathrm{DFL}_i &= \frac{\boldsymbol{p}_i \cdot \boldsymbol{x}_i}{\boldsymbol{p}_{i-1} \cdot \boldsymbol{x}_i} \times \mathrm{DFL}_{i-1}, \quad i = 1, 2, \ldots, t \end{aligned} \tag{3.2}$$

と定義すれば，

$$y_t = \frac{\boldsymbol{p}_t \cdot \boldsymbol{x}_t}{\mathrm{DFL}_t}$$

とできる。もちろん，DFL_t は当期の価格指数として用いることができる。このように定義される価格指数 DFL が連鎖方式の **GDP デフレータ**である。次のおなじみの公式が成り立っている[5]。

$$\text{実質 GDP} = \frac{\text{名目 GDP}}{\text{GDP デフレータ}}$$

[5] 理論上は，(3.2) のようにデフレーターを計算してから実質 GDP を計算しても，(3.1) で実質 GDP を計算してからデフレーターを計算することもできる。実務上は，実質 GDP を先に計算しているようだ。

$$\text{デフレーター} = \frac{\text{名目値}}{\text{実質値}}$$

のように計測されるデフレーターを「インプリシット・デフレーター」という。

問題 3.1. 名目 GDP の変化率が 2%，実質 GDP の変化率（経済成長率）が 1.5% であったとする。このとき，GDP デフレータで測った物価変化率（インフレ率）はいくらか。計算しなさい。

Answer

3.2.3 寄与度と寄与率

政府発表やそれに基づいた新聞記事では，経済成長が加速したり減速したりする現象の要因を説明するための記述として，「個人消費が順調だった」とか「設備投資が伸び悩んだ」といった記述をよく見かける。この手の分析では**寄与度**（contribution）や寄与率という概念が用いられている。

成長率 t 期の GDP を Y_t，その次の期（$t+1$ 期）の GDP を Y_{t+1} とする[6]。経済成長率は GDP の変化率

$$\frac{\Delta Y_{t+1}}{Y_t} = \frac{Y_{t+1} - Y_t}{Y_t} = \frac{Y_{t+1}}{Y_t} - 1$$

によって定義される。Y_t,Y_{t+1} の分解を次のように書こう。

$$Y_t = C_t + I_t + G_t + NX_t$$
$$Y_{t+1} = C_{t+1} + I_{t+1} + G_{t+1} + NX_{t+1}$$

GDP が成長するためには C,I,G,NX のうち少なくとも 1 つは 大きくなっているはずだ。GDP 成長率を項目ごとに分解するのが寄与度である。

[6] ここでは慣例に従って大文字を利用しているが，実質変数（名目変数をデフレータで除した変数）であると考えてよい。

寄与度

項目ごとの寄与に分解するためには，GDP 成長率の定義に加法分解の公式を代入すればよい。

$$\begin{aligned}\frac{\Delta Y_{t+1}}{Y_t} &= \frac{Y_{t+1}-Y_t}{Y_t}\\ &= \frac{(C_{t+1}+I_{t+1}+G_{t+1}+NX_{t+1})-(C_t+I_t+G_t+NX_t)}{Y_t}\\ &= \frac{(C_{t+1}-C_t)+(I_{t+1}-I_t)+(G_{t+1}-G_t)+(NX_{t+1}-NX_t)}{Y_t}\\ &= \frac{\Delta C_{t+1}+\Delta I_{t+1}+\Delta G_{t+1}+\Delta NX_{t+1}}{Y_t}\\ &= \frac{\Delta C_{t+1}}{Y_t}+\frac{\Delta I_{t+1}}{Y_t}+\frac{\Delta G_{t+1}}{Y_t}+\frac{\Delta NX_{t+1}}{Y_t}\end{aligned}$$

ここで，例えば $\Delta I_{t+1}/Y_t$ が，GDP 成長における投資の寄与度である。通常，100 倍して % 表記する。

GDP に限らず加法的に分解できる指標であれば同様の公式で寄与度を計算出来る。例えば，

$$X_t = X_{t,1}+X_{t,2}+\cdots+X_{t,N-1}+X_{t,N} = \sum_{n=1}^{N} X_{t,n}$$

とすれば，

$$\frac{\Delta X_{t+1}}{X_t} = \sum_{n=1}^{N}\frac{\Delta X_{t+1,n}}{X_t} \Longrightarrow \text{項目}n\text{ の寄与度} = \frac{\Delta X_{t+1,n}}{X_t}$$

問題 3.2. $X_t, t = 2000, 2001,$ の加法分解 $X_t = X_{t,1}+X_{t,2}+X_{t,3}$ が下の表で与えられている。2000 年から 2001 年にかける X の成長について，項目 1, 2, 3 の寄与度を計算しなさい。

	$X_{t,1}$	$X_{t,2}$	$X_{t,2}$
2000 年	100	300	150
2001 年	120	330	140

Answer

加法分解に控除項目が入る場合には，控除項目を負数として扱えばよい。

$$\begin{aligned} Y_t &= C_t + I_t + G_t + EX_t - IM_t \\ &= C_t + I_t + G_t + EX_t + (-IM_t) \end{aligned}$$

$$\frac{\Delta Y_{t+1}}{Y_t} = \frac{\Delta C_{t+1}}{Y_t} + \frac{\Delta I_{t+1}}{Y_t} + \frac{\Delta G_{t+1}}{Y_t} + \frac{\Delta EX_{t+1}}{Y_t} + \frac{\Delta(-IM_{t+1})}{Y_t}.$$

問題 3.3. 2017 年度と 2018 年度の実質 GDP（支出側）の構成は下表の通りであった。単位は兆円。このとき，経済成長率と，各項目の寄与度を計算しなさい。

	C	I	G	EX	IM
2017 年度	298.88	101.56	132.33	91.43	92.62
2018 年度	299.05	102.28	139.39	92.87	94.62

Answer

寄与率

GDP 成長率を 1（100%）に基準化したときの寄与度の大きさが寄与率である。例えば，GDP の支出面の分解であれば，寄与度の計算式

$$\frac{\Delta Y_{t+1}}{Y_t} = \frac{\Delta C_{t+1}}{Y_t} + \frac{\Delta I_{t+1}}{Y_t} + \frac{\Delta G_{t+1}}{Y_t} + \frac{\Delta NX_{t+1}}{Y_t}$$

の両辺を全体の成長率 $\Delta Y_{t+1}/Y_t$ で割ってやればよい。

$$\begin{aligned} 1 &= \frac{(\Delta C_{t+1}/Y_t) + (\Delta I_{t+1}/Y_t) + (\Delta G_{t+1}/Y_t) + (\Delta NX_{t+1}/Y_t)}{(\Delta Y_{t+1}/Y_t)} \\ &= \frac{\Delta C_{t+1}}{\Delta Y_{t+1}} + \frac{\Delta I_{t+1}}{\Delta Y_{t+1}} + \frac{\Delta G_{t+1}}{\Delta Y_{t+1}} + \frac{\Delta NX_{t+1}}{\Delta Y_{t+1}} \end{aligned}$$

問題 3.4. 問題 3.2,3.3 について，各項目の寄与率を計算しなさい。

Answer

3.3 理論: 格差の指標

前節までは GDP の拡大と，その評価方法について説明してきた。「GDP の拡大は善である」という立場を暗黙のうちに取っていた。GDP は一国経済の居住者の平均的な暮らし向きを表す重要な指標であるので，多くの経済分析ではこの前提を置いている。しかし近年では，環境への影響や所得不平等の広がりを背景として，GDP の拡大だけを目標とする政策に対する批判が強くなってきている。ここでは，所得の不平等や格差を測る指標について，その一端を紹介する。

3.3.1 順序統計量

概念の理解に必要な順序統計量について説明する。

順序に意味のない n 個の数値データ

$$x_1, x_2, x_3, \ldots, x_n$$

を小さい順に並べ替える。新しい記号を導入したくないので，下付き添字に括弧を付けて順位を表すものとする。すなわち，

$$\begin{aligned} x_{(1)} &= 1 \text{ 番目に小さいデータ} \\ x_{(2)} &= 2 \text{ 番目に小さいデータ} \\ &\vdots \\ x_{(n)} &= n \text{番目に小さいデータ} \end{aligned}$$

x_i と書いたときの i はデータに付けた単なる ID で特に意味のないものである。$x_{(j)}$ と書いたときの j はデータを小さい順に並べて j 番目のデータである。$x_{(j)}, j = 1, \ldots, n$, の作り方から，次の不等式が成り立つ。

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n-1)} \leq x_{(n)}$$

最小値・最大値

データ $\{x_1, \ldots, x_n\}$ に対して，$x_{(1)}$ を**最小値**，$x_{(n)}$ を**最大値**という。

中央値

データ $\{x_1, \ldots, x_n\}$ に対して，**中央値**（**median**）を次のように定義する。

$$\mathrm{med}(\{x_1, \ldots, x_n\}) = \begin{cases} x_{(\frac{n+1}{2})} & n \text{ が奇数のとき} \\ \frac{x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)}}{2} & n \text{ が偶数のとき} \end{cases}$$

中央値は昇順で並べたデータを半々に分ける数のことである。

分位数

$q \in [0,1]$ に対して[*7]，q**-分位数**と呼ばれる統計量 Q_q は次のように定義される。

$$\frac{x_{(i)} \leq Q_q \text{ を満たす } i \text{ の数}}{n} = q$$

中央値は $Q_{1/2}$ である。厳密な定義は難しいので省略する。詳細は**?**を参照。

3.3.2　相対的貧困率

相対的貧困率（relative poverty rate）の計算には可処分所得（所得から税・社会保険料等を引いた額）を用いる。所得を得ていない個人もいるので，世帯人員数も考慮した**等価可処分所得**を以下のように計算する。

$$\text{個人の等価可処分所得} = \frac{\text{属する世帯の可処分所得の総額}}{\sqrt{\text{属する世帯の人員数}}}$$

属する世帯の人員数の平方根を用いるのは，世帯人員数が多いほうが 1 人あたりの生活コストが下がることを考慮したものである。例えば，3 人世帯の家賃は 1 人暮らしの人

[*7] $a \in [0,1]$ は $0 \leq a \leq 1$,$a \in (0,1)$ は $0 < a < 1$ という意味である。丸括弧と角括弧を組み合わせて $a \in [0,1) \Leftrightarrow 0 \leq a < 1$ などとも書く。

の家賃の 3 倍にはならない。例えば下表左のような可処分所得の分布であれば，等価可処分所得は下表右のように計算できる。

個人	世帯	可処分所得
1	A	400
2	A	300
3	A	0
4	B	1,000
5	B	0
6	C	250
7	D	150

$\Longrightarrow$

個人	世帯人員数	等価可処分所得
1	3	404.1452
2	3	404.1452
3	3	404.1452
4	2	707.17
5	2	707.17
6	1	250
7	1	150

等価可処分所得の中央値（中位点所得）を半分にした数値を**相対的貧困線**と定義する。上の例には 7 人の個人がいるので，大きい順にならべて 4 番目にあたる所得が中央値である。

$$\text{上の例の中央値} = 404.1452$$

さらにこれを半分にしたものが相対的貧困線である。

$$\text{相対的貧困線 RPL} = \frac{404.1452}{2} = 202.0726$$

相対的貧困とは，この相対的貧困線未満の所得しかない状態である。上の例で，相対的貧困の状態にある個人について True，そうでない個人について False を付したものが下の表である。

個人	世帯人員数	等価可処分所得	相対的貧困
1	3	404.1452 ≮ RPL	False
2	3	404.1452 ≮ RPL	False
3	3	404.1452 ≮ RPL	False
4	2	707.17 ≮ RPL	False
5	2	707.17 ≮ RPL	False
6	1	250 ≮ RPL	False
7	1	150 < RPL	True

True となる個人の割合が相対的貧困率である。

$$\text{相対的貧困率} = \frac{\text{等価可処分所得が中位点所得の半分未満である人の数}}{\text{総人口数}}$$

上の例では，$1/7 \fallingdotseq 14\%$ となる。

問題 3.5. 高所得グループと低所得グループとで二極化している場合には，相対的貧困率が高くなる傾向がある。このことを適当な数値例を構築して確認しなさい。

Answer

相対的貧困の状態にあると分類される人たち全員が日々の衣食にも困るような日常的な意味での「貧困」状態にあるとは限らない。しかし，彼らの生活は，多くの世帯が享受できている標準的な生活水準に達していない可能性が高いし，教育を受けることを諦めてしまう可能性も高く，次の世代に貧困を引き継いでしまうおそれがある。したがっ

て，政府は目を配る必要があるし，多くの団体がサポートしようとしている。

相対的貧困の定義上，相対的貧困率は 50% を超えることはない。したがって，国民の多くが「貧困状態」にある経済の分析をする場合には相対的貧困を使うことができない。生活必需品を買うことすらままならないような極度の貧困状態を測るためには**絶対的貧困**という指標を用いる。発展途上国の支援の文脈では絶対的貧困が使われる。

問題 3.6. 絶対的貧困の定義を調べて，調べたことを簡潔にまとめなさい。

Answer

3.3.3 ジニ係数

所得分布の不平等を測る指標の 1 つとして**ジニ係数**（Gini coefficient）がよく使われている。データ $\{y_1, y_2, \ldots, y_n\}$ を個人の所得（等価可処分所得）であるとしよう。各個人の所得を小さい順に並び替えた

$$y_{(1)}, y_{(2)}, \cdots, y_{(n-1)}, y_{(n)}$$

をもとに次のようなデータを構築する。所得順位が下から i 番目の個人までの所得の合計を

$$\hat{y}_i = y_{(1)} + y_{(2)} + \cdots + y_{(i)}$$

と定義する。便宜上，$\hat{y}_0 = 0$ とする。国民の総所得は

$$\hat{y} = \hat{y}_n = y_{(1)} + y_{(2)} + \cdots + y_{(n)}$$

第 i 番目の個人までの所得 $\hat{y}_i$ が総所得 $\hat{y}$ にしめる割合 $Y_i = \hat{y}_i/\hat{y}$ のベクトル

$$\begin{aligned}\boldsymbol{Y} &= (Y_0, Y_1, \ldots, Y_{n-1}, Y_n) \\ &= \left(\frac{\hat{y}_0}{\hat{y}}, \frac{\hat{y}_1}{\hat{y}}, \ldots, \frac{\hat{y}_{n-1}}{\hat{y}}, \frac{\hat{y}_n}{\hat{y}}\right) \\ &= \left(0, \frac{y_{(1)}}{y_{(1)} + \cdots + y_{(n)}}, \ldots, \frac{y_{(1)} + \cdots + y_{(n-1)}}{y_{(1)} + \cdots + y_{(n)}}, 1\right)\end{aligned}$$

を作る。さらに，i 番目までの個人の全体の総人口 n に占める割合 i/n を並べたベクトルも構築する。

$$\boldsymbol{X} = \left(0, \frac{1}{n}, \frac{2}{n}, \frac{3}{n}, \ldots, \frac{n-1}{n}, 1\right).$$

このようにして構築した $\boldsymbol{X}$,$\boldsymbol{Y}$ をプロットした点を結んだ線を**ローレンツ曲線**という。

問題 3.7. すべての個人が同じ所得を受け取っているとき，すなわち完全な平等が達成されているとき，ローレンツ曲線は $Y = X$ で示される直線になる。この事実を示しなさい。

Answer

完全な平等が達成されているときのローレンツ曲線を**完全平等線**と呼ぶ。図 3.1 の実線がローレンツ曲線の一例である。完全平等線と実際のローレンツ曲線に囲まれる領域（図 3.1 の影をつ付けた部分）の面積を 2 倍にした値がジニ係数である[8]。

$$G = \text{ジニ係数} = 2 \times \left(\text{ローレンツ曲線と完全平等線を囲む領域の面積}\right)$$

[8] 完全平等線について対称にローレンツ曲線をもう 1 つ描くと，ジニ係数はこれら 2 つのローレンツ曲線で囲まれる部分の面積になる。

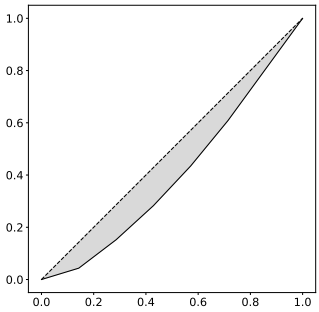

図 3.1: ローレンツ曲線

定義より次の関係が成り立つ。

$$0 \leq G < 1$$

$G = 0$ はすべての個人が同じ所得を得ているときに限って成り立つ。$G = 1$ にもっとも近づくのは一人の個人が全所得を独占している場合である。完全な不平等を保った状態で総人口について $n \to \infty$ の極限を取れば，ジニ係数は 1 に収束する。この「完全な不平等」のケースについて次の問題を考えてみよう。

問題 3.8. $y_{(n)}$ を稼ぐ個人を除いて所得がゼロであるとする。すなわち，

$$0 = y_{(1)} = y_{(2)} = \cdots = y_{(n-1)} < y_{(n)}$$

が成り立つとする。この場合のローレンツ曲線を描きなさい。また，$n \to \infty$ の極限でジニ係数が 1 となることを示しなさい。

Answer

ジニ係数は 0 に近いほど不平等が小さく，1 に近ければ近いほど不平等が大きいものとみなせる。ジニ係数の国際比較や経年変化を見ることによって，各国の不平等を評価することができる。

注意 3.1. 個人の所得データからではなく，集計データ（例えば，所得階級ごと相対度数）からジニ係数を計算することも多い。詳細は**?** を参照。

注意 3.2. 政府には，税や社会保障を通じて，所得の格差を縮小する役割がある。これを所得再分配という。再分配前のジニ係数と再分配後のジニ係数を比較することで，所得再分配政策の効果を測ることができる。

便宜上 $y_{(0)} = 0$ とする。n 個の台形の面積を足し上げればよいので，

$$\text{ローレンツ曲線の下側の面積} = \sum_{i=1}^{n} \left[\sum_{j=1}^{i} \frac{Y_{j-1} + Y_j}{2n} \right]$$

となる。したがって，ジニ係数は

$$\begin{aligned} G &= 1 - \sum_{i=1}^{n} \sum_{j=1}^{i} \frac{Y_{j-1} + Y_j}{n} \\ &= 1 - \frac{\sum_{i=1}^{n} Y_{j-1} + \sum_{j=1} Y_j}{n} \\ &= 1 - \frac{2\left(\sum_{j=1}^{n} Y_j\right) - Y_n}{n} \\ &= 1 - \frac{2\left(\sum_{j=1}^{n} Y_j\right) - 1}{n} \end{aligned} \tag{3.3}$$

とできる。総和記号 $\sum$ の部分は

$$\sum_{j=1}^{n} Y_j = \sum_{j=1}^{n} \left(\frac{y_{(1)} + y_{(2)} + \cdots + y_{(j)}}{y_{(1)} + y_{(2)} + \cdots + y_{(j)} + y_{(j+1)} + \cdots + y_{(n)}} \right)$$

とできることを思い出そう。データを小さい順に並べ替えて，累積和の和を計算している。

3.3.4 トップ 1% の所得シェア

超富裕層に富が集中している傾向について，近年大きな社会問題になっている。（ピケティ『21 世紀の資本』を参照する）

所得上位 1% の人々の所得が全所得に占める割合（top percentile income share）が，近年大幅に上昇しているというのだ。

上位 1% は下位 99% と同じなので，0.99-分位点以上の所得を合計すればよい。すなわち，

$$\text{(Top percentile income share)} = \frac{\sum_{i:y_i \geq Q_{0.99}} y_i}{\sum_i y_i}$$

問題 3.9. それぞれの定義に基づいて，ジニ係数とトップ 1% 所得シェアの類似性と相違点を説明しなさい。

Answer

3.4　プログラミング: 基礎

テキストファイルに書き込んだプログラムは基本的には上から順に実行される。例えば，次のコードは (1) 関数 `f()` を呼び出す，(2) 関数 `g()` を呼び出す，(3) 関数 `h()` を呼び出す，という順に実行される。

```
f()
g()
h()
```

注意 3.3.　`f()`,`g()`,`h()` を定義していないので，このコードは実行できない。実行できないコードは理解しにくいという方は，次のように打ち込んでおくとよい。

```
def f(): print("f is called.")
def g(): print("g is called.")
def h(): print("h is called.")
```

□

改めて言うまでもない当たり前のことと感じるかもしれないが，これは非常に重要なことなので忘れないでほしい[*9]。

線形の処理の流れを変更するための構文を**制御フロー構文**という。この節では，

- ループ
- 条件分岐

について説明する。

Python ではループの使用はできる限り避けるべきであるとされる。**NumPy** を用いてベクトル化することでループを回避する方法，さらにはベクトル化によって避けることのできないループについても説明する。

[*9] 特に，Jupyter Notebook などの環境でコードを書いているときには，コードの見かけ上の前後関係と実行順が異なる場合があるので混乱が生じやすい。気をつけよう。

3.4.1 ループ

ループは同じような処理を繰り返し実行するための構文である。for ループと while ループがある。

次のコードは同じコマンド f() を 3 回実行するためのコードである。

\«ch03/no-loop/dnr, caption=実行したい処理", fig.lp='code:'»= @

3 回くらいであればこのままでもいいかもしれないが，100 回，1000 回，あるいは 100 万回繰り返す必要がある場合はどうすればよいだろう。

for ループ

for ループは繰り返しの実行回数が事前に分かっているときに使う構文である。次のように書く（ただし，これはまだ最善ではない）。

Code 3.1 for ループ

```
for i in [0, 1, 2]:
    f()

g()
```

図 3.2 に for ループの書き方を解説しているので確認してほしい。いちばん重要なポイントは，**実行される処理の本体にあたるコードは半角スペース 4 つ分のインデントで明示する**，ということである。実際には，コード全体で統一された数でありさえすれば 4 つでなくてもよいが，4 つ使うのが慣例である。したがって，Python でプログラミングをするときは Tab キーを押したときにスペース 4 つが入力されるように設定できるテキストエディタを使うべきである。

上の Code3.1 は for ループ を使わない次のコードと同等である。

```
i = 0
f()

i = 1
f()
```

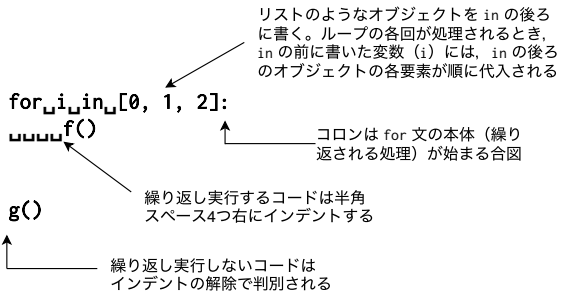

図 3.2: for ループの文法

```
i = 2
f()

g()
```

使用されないカウンター i に代入する処理が増えたせいで，元のコード（Code**??** ）よりも長くなってしまった。しかし，for ループとはこういうものだ。実際には一時的に代入されるオブジェクトも使用できる[*10]。

Code 3.2 カウンターを使う for ループ

```
for i in [0, 1, 2]:
    print(i)
```

```
0
1
2
```

問題 3.10. Code3.2 を for ループを使わないコードに書き換えなさい。

Answer

[*10] print() 関数はオブジェクトの情報を画面に表示する関数である。IPython や Jupyter を使っている場合に print() を使用する必要はないが，いずれ必要になる場面が出てくるので覚えておこう。

range()

上のコードには問題がある。[0, 1, 2] と書くのが面倒くさいということだ[*11]。3 回繰り返しではなく 1000 回繰り返しだったらどうするのか。この問題は，range() 関数を使って解決できる。繰り返し回数が n であれば，in の後ろに range(n) と書けばよい。

```
for i in range(3):
    print(i)
```

```
0
1
2
```

range(n) は [0, 1, ..., n-1] と同じような意味である 。[0, 1, ..., n-1] と書くと整数 n 個分のメモリが消費されるが，range(n) にはそのようなムダがないようにできている[*12]。

問題 3.11. 整数 $i = 0, 1, \ldots, 9, 10$ について i^3 を表示するコードを書きなさい。

Answer

[*11] 面倒なことを繰り返しやらされると人間は必ず間違える。そういう仕事はプログラミングを使ってなくさなければならない。

[*12] list(range(n)) とすれば range オブジェクトが リストに変換される。for ループで使うときはリストに変換してはいけない。

問題 3.12. $m < n$ なる整数 m, n に対して，`range(m,n)` はどのような数列を表しているか。

Answer

3.4.2　真偽値と while ループ

`while` ループは 繰り返しの実行回数が事前に分かっていなくても使える構文である[13]。`while` ループを理解するためには真偽値を理解する必要がある。

真偽値

Python では `True`,`False` という名前に特別な意味がある。次のコードを IPython で実行してみよう。

入力した結果がそのまま表示されるだけだが，それでよい。これらの名前と値が事前に定義されている証拠だ。

問題 3.13. `true`,`false`,`TRUE`,`FALSE` と入力するとどのような結果になるか。また，それは何故か。

[13] `for` と `while` に関するこのような切り分けは プログラミング言語一般に成り立つものではない。Python の `for` ループは range-based for loop と呼ばれる処理である。

Answer

True は「条件が成り立つ（真である）」ことを意味する値，False は「条件が成り立たない（偽である）」ことを意味する値である。and（かつ）,or（または），not（否定）の演算がある。

```
[True and True, True and False, False and True, False and False]
```

```
  [True, False, False, False]
```

```
[True or True, True or False, False or True, False or False]
```

```
  [True, True, True, False]
```

```
[not True, not False]
```

```
  [False, True]
```

条件式

結果が真偽値となる式を条件式という。数値比較のための以下の演算子が基本的である。

表 3.1: 数値比較の演算子

x == y	x と y は等しい
x < y	x は y より小さい
x <= y	x は y より小さいか等しい
x > y	x は y より大きい
x >= y	x は y より大きいか等しい

```
[1 < 2, 1 <= 2, 1 == 2, 1 > 2, 1 >= 2]
```

```
[True, True, False, False, False]
```

問題 3.14. 次の各条件を表す Python の条件式を書きなさい。

1. $x \geq y$ または $x \geq z$
2. $x < y < z$

Answer

真偽値の足し算

真偽値に対して四則演算を実行しようとすると，

- `True` $\longrightarrow 1$
- `False` $\longrightarrow 0$

に勝手に変換される。この性質を使うと，リストや配列の中の `True` の数を数えるのに役立つ。

```
True + True + False
```

```
2
```

浮動小数点数に関する注意

コンピュータで実数を扱う場合，**浮動小数点数**と呼ばれる近似を用いる。あくまで近似なので，厳密な等式で比較することは避けた方がよい。私たちが有限小数と認識している普通の小数であっても，コンピュータは正確に表現できないものがある。次の 2 つのコードを比較してみよう。

```
0.1 + 0.3 + 0.6 == 1.0
```

```
True
```

```
0.3 + 0.6 + 0.1 == 1.0
```

```
False
```

浮動小数点数が計算に現れる場合，数値が近似的に等しいことを `np.allclose()` といった関数を用いて評価する。浮動小数点数の厳密な等号比較が必要になるような処理は不可能である。この話は行列の標準化を扱う際に重要になる。

```
import numpy as np
np.allclose(0.1 + 0.3 + 0.6, 1.0, rtol=1e-15)
```

```
True
```

`rtol=1e-15` とあるのは，この基準よりも小さい誤差はゼロとみなすということ。`1e-15` は 1×10^{-15} という意味である。

while ループ

繰り返し処理の継続条件が条件式で表されているときには `while` ループを使う。基本は次のような形式になる。処理の本体にあたる部分をスペース 4 つ分のインデントで区別する点などは `for` ループと同じである。

```
while condition:
    f()
```

ここで `condition` の部分に条件式が入る。

- `condition` が `True` であればループの本体を実行し，次のループに進む。
- `condition` が `False` であれば ループを抜ける。

もちろん，`while` ループが正しく終了するためには，各回の処理のたびに条件式が変わっていく必要がある。例えば次のコードは，Code3.1 を `while` ループを使って書き直したものだ。ただし，`i += 1` は `i = i + 1` と同じ意味である。

```
i = 0
```

```
while i < 3:
    f()
    i += 1

g()
```

問題 3.15. 次のコードを IPython で実行すると何が起こるか。現象と理由を説明しなさい。（実行を終了するときはキーボードで Ctrl + C を押す）

```
while True:
    print(i)
    i += 1
```

Answer

`while` ループは非常に複雑なループ処理もできるが，本章では使わないので，紹介はこれくらいにしておく。もっと実用的な `while` ループについては必要になったときに説明する。

3.4.3　条件分岐

`if` 文を使うと，指定した条件が成り立つときにだけ実行される処理を指定することができる。条件が成り立たない場合の処理は `else` ブロックで指定する。実行条件が 2 つ以上ある場合には `elif` ブロックを用いる。`elif` ブロックは複数書くことができる。処理の本体にあたる部分をスペース 4 つ分のインデントで区別する点などは `for`, `while` の構文と同じである。

```
if condition:
    f()
```

```
if condition:
    f()
else:
    g()
```

```
if condition1:
    f()
elif condition2:
    g()
else:
    h()
```

ループと組み合わせると面白いことができる。

例 3.1. 1 から 100 の間に含まれる奇数の合計は次のように計算できる。

Code 3.3 奇数の合計

```
total = 0
for i in range(1, 101):
    if i % 2 == 1:
        total += i

total
```

```
2500
```

`if` の本体ブロック（`total += i`）の先頭に 8 個分のスペースが挿入されている。`for` のために 4 つ，`if` のために 4 つである。 □

問題 3.16. 1 から 1000 の間に含まれる偶数の合計を計算するコードを書きなさい。

Answer

3.4.4　ジェネレータ内包表記

Code3.3 は実際には次のように書かれることが多い。ここでは詳しい説明は省くが，ジェネレータ内包表記と呼ばれる書き方である。英文と思って読めば，なんとなく使い方が分かるはずだ。

```
sum(i for i in range(1, 101) if i % 2 == 1)
```

```
2500
```

もちろん，sum() という関数（np.sum() ではない！）がこのような書き方を許しているというだけであって，あらゆる関数に使える訳ではない。

3.4.5　真偽値の配列

NumPy 配列を比較演算子で比較した結果は真偽値の **NumPy** 配列になる。

```
x = np.array([1, 2, 3])
y = np.array([3, 2, 1])
x < y
```

```
array([ True, False, False])
```

要素取得の角括弧の中に真偽値の配列を入れる。結果は True の位置にあたる要素である。

```
x[x < y]
```

```
array([1])
```

True が 1，False が 0 であることを思い出そう。0 と 1 からなるデータの総和を取ると 1 の個数，平均を取ると 1 の割合が計算できる。したがって，次のコードは 0.05 に近い値になると期待される。

```
z = np.random.random(1000)
np.mean(z > 0.95)
```

```
0.07
```

3.5 プログラミング: 実践

前置きが長くなった。数値例を使って理論パートで学んだ計算方法を確認しよう。

3.5.1 連鎖方式の実質 GDP と GDP デフレータ

前章で用いた簡単な数値例を用いよう。A，B，C の財グループがあるとして，2000 年を基準に価格指数を計算したものが下表左であり，生産数量が下表右であるとする。

実質 GDP の計算例

価格	A	B	C
2000 年	100	100	100
2001 年	101	99	103
2002 年	100	98	104
2003 年	99	99	106

数量	A	B	C
2000 年	1000	2000	500
2001 年	980	1980	510
2002 年	1010	1990	520
2003 年	1005	2005	530

実際の価格ではなく価格指数を使うことで一般性が失われるということはない。実質 GDP にせよ価格指数にせよ，価格上昇率のみに意味があり，水準には意味がないからだ。理由が分からない場合は理論パートの式を丁寧に検証し直すとよい。

名目 GDP，GDP デフレータ，実質 GDP の順に計算しよう。3 群 × 4 年分の人工的なデータを用いたが，適切な形式でデータを格納していれば 3 とか 4 とかいう数字はコードには出てこない。したがって，ここで紹介するコードは，何千もの商品群を考慮している実際の計算でも同じように使えるはずだ。心してかかろう。

まずは表を配列形式で格納する。データの格納形式とその後のコードには密接に関連している。行が伸びる方向に時間が進んでいくものとしよう。配列全体が実数のデータであることを Python に知らせるためには，少なくとも 1 つの数字に小数点をつければよい[*14]。

[*14] NumPy の配列には同じ種類のデータが並ぶことを思い出そう。1 つが浮動小数点数なら全部が浮動小数点数になる。

```
price = np.array([[100, 100, 100],
                  [101, 99, 103],
                  [100, 98, 104],
                  [99, 97, 106.]])
quantity = np.array([[1000, 2000, 500],
                     [980, 1980, 510],
                     [1010, 1990, 520],
                     [1005, 2005, 530.]])
```

名目 GDP を計算するためには，この表を成分ごとにかけてから（演算子は* だ），結果の配列の各行について，全列の和を取る。これは，np.sum(x, axis=1) でできる。

```
nominal_gdp = np.sum(price * quantity, axis=1)
nominal_gdp
```

```
array([350000., 347530., 350100., 350160.])
```

式 (3.2) に基づいてデフレータを計算するのが次のコードだ。前章で学んだコード（連鎖方式のラスパイレス指数を計算したことを思い出そう）とこの章で学んだ for の構文を組み合わせればよい。

```
dfl = np.empty_like(nominal_gdp)
dfl[0] = 1.
for t in range(1, len(dfl)):
    dfl[t] = dfl[t-1] * (np.sum(price[t, :] * quantity[t, :])
                         / np.sum(price[t-1, :] * quantity[t, :]))

dfl * 100
```

```
array([100.         , 100.15273775,  99.44827695,
       98.89752821])
```

実質 GDP は名目 GDP をデフレータで除すことで計算できる。

```
real_gdp = nominal_gdp / dfl
real_gdp
```

```
array([350000.         , 347000.         , 352042.2985066 ,
       354063.44966341])
```

3.5.2 寄与率と寄与度

複数年にわたる計算

表**??**の例を使って計算しよう。ここでは，名目価値の成長要因を分析する。実質変数の場合も，各項目の実質値の推計が手に入れば同じように実行できる。使うデータは `price * value` を計算した次のデータである。

```
value = price * quantity
value
```

```
array([[100000., 200000.,  50000.],
       [ 98980., 196020.,  52530.],
       [101000., 195020.,  54080.],
       [ 99495., 194485.,  56180.]])
```

価格	A	B	C	GDP	成長率
2000 年					
2001 年					
2002 年					
2003 年					

各行の総和はすでに計算した `nominal_gdp` に一致する。これは計算の方法から当たり前である。

```
np.allclose(nominal_gdp, np.sum(value, axis=1), rtol=1e-15)
```

```
True
```

名目 GDP の成長率は次のように計算できる。`np.diff()` は隣接する要素の差を計算する関数である。デフォルトでは元の変数よりも 1 つ長さが短いベクトルが出力される。この振る舞いを `prepend=np.nan` を付けることで変えることができる[15]。これは，

[15] **NumPy** のバージョンが v1.16.0 よりも古い場合，`prepend=np.nan` が使えない。この場合は **NumPy** のアップデートをお勧めする。その権限がない場合には，`np.r_[[np.nan],`

出力の先頭要素に「存在しない値」を意味する NaN (Not a Number) を追加する設定である[16]。NaN を付加すれば，名目 GDP の成長率は次のように簡単に計算できる。

```
np.diff(nominal_gdp, prepend=np.nan) / np.roll(nominal_gdp, shift=1)
```

```
array([         nan, -0.00705714,  0.00739505,  0.00017138])
```

寄与度を計算するためには，まず各項目の増分を計算しないといけない。これにはデータ行列 `value` に対して `np.diff()` を取ればよい。ただし，`axis=0` を付けて行が伸びる方向に差分を取ることを指定する。まとめると，`np.diff(value, prepend=np.nan, axis=0)` が各項目の差分時系列を表している[17]。これを名目 GDP で除したものが寄与度である。単純に `nominal_gdp` で割るとサイズが合わないというエラーになるので，4×1 配列であることを明示するために `nominal_gdp.reshape(4, 1)` で割るとよさそうだ。しかし，これでは $\Delta X_{t+1}/Y_{t+1}$ を計算してしまうので，`np.roll(nominal_gdp, shift=1)` を先に

```
contribution = (np.diff(value, prepend=np.nan, axis=0)
                / np.roll(nominal_gdp, shift=1).reshape(4, 1)) * 100
contribution
```

```
array([[         nan,          nan,          nan],
       [-0.29142857, -1.13714286,  0.72285714],
       [ 0.58124478, -0.28774494,  0.44600466],
       [-0.42987718, -0.15281348,  0.59982862]])
```

最後に，寄与度の合計を計算すると成長率になることを確認しておこう。

```
contribution.sum(axis=1)
```

`np.diff(nominal_gdp)] / nominal_gdp` などとする。

[16] 本来，NaN は `0.0/0.0` のように答えが定められない計算の結果を示す値である。Python には「欠損」(Not Available, missing) を表すための専用の値がなく，**NumPy** や **math** ライブラリがサポートする NaN を欠損に転用している。しかし，R や Julia のように言語レベルで欠損をサポートするプログラミング言語と違って，欠損値として一貫性のある振る舞いは期待できない。Python 用のデータ分析ライブラリである **pandas** には `pandas.NA` という欠損を表すための値が最近になって追加されたので，この状況は少しずつ改善されるだろう。

[17] **NumPy** のバージョンが v1.16.0 よりも古い場合，`prepend=np.nan` が使えない。この場合は **NumPy** のアップデートを改めてお勧めする。どうしても認めてもらえない場合には，`np.diff(..., prepend=np.nan)` の部分を `np.r_[[[np.nan] * value.shape[1]], np.diff(value, axis=0)]` などと置き換えればよい。

```
array([        nan, -0.70571429,  0.7395045 ,  0.01713796])
```

3.5.3 不平等指数

相対的貧困率

相対的貧困線の計算に必要な中央値は `np.median()` を用いる。

```
x = np.array([3, 6, 10, 14, 3, 2, 1, 1])
np.median(x)
```

```
3.0
```

相対的貧困に該当するデータを発見するには次のようにする。

```
x < np.median(x) / 2
```

```
array([False, False, False, False, False, False,  True,
    True])
```

このデータのうち `True` の割合を計算したものが相対的貧困率である。

```
np.mean(x < np.median(x) / 2)
```

```
0.25
```

問題 3.17. 節 3.3.2 の例に対して相対的貧困率を計算するプログラムを書きなさい。

Answer

ジニ係数

次の人工的なデータに対してジニ係数を計算しよう。

```
x = np.random.choice(np.arange(30.), 10)
```

```
x
```

```
array([19., 29., 17., 19.,  5., 18., 10., 26., 13., 12.])
```

ジニ係数を計算するためには，まずデータを小さい順に並べ替える必要がある[18]。

```
y = np.sort(x)
y
```

```
array([ 5., 10., 12., 13., 17., 18., 19., 19., 26., 29.])
```

式 (3.3) に基づいて計算する。Y_i は累積和を総和で割ったものだから，次のように計算すればよい。データの先頭にゼロを付加していることに注意。

```
Y = np.r_[0, y.cumsum()] / y.sum()
Y
```

```
array([0.        , 0.0297619 , 0.08928571, 0.16071429,
       0.23809524, 0.33928571, 0.44642857, 0.55952381,
       0.67261905, 0.82738095, 1.        ])
```

最後に式 (3.3) の公式に当てはめればよい。

```
gini = 1 - (2 * np.sum(Y) - 1) / y.size
gini
```

```
0.22738095238095235
```

トップ 1% 所得シェア

Q_q 統計量は `np.quantile(a, q)` で求められる。ここで，`a` が配列形式のデータで，`q` は下側 $100q$ % を表す $[0, 1]$ の数字である。トップ 1% の所得水準以上の所得というのは，

$$\text{所得} \geq Q_{0.99}$$

と書けるので，次のように書けばトップ 1% のデータを判別できる。

```
x[x >= np.quantile(x, 0.99)]
```

```
array([29.])
```

[18] 中央値の計算のためにも並べ替えは必要だが，`np.median()` がその工程を隠蔽している

この総和を計算して，全体に占める割合を計算しよう。

```
x[x >= np.quantile(x, 0.99)].sum() / x.sum()
```

```
0.17261904761904762
```