R による時系列分析の方法 1[†]

以下では統計ソフト R を使って時系列分析をやる方法を簡単に要約する。具体的には

0. R をインスツールする
1. データを読み込む
2. データをプロットする
3. 変化率の計算
4. 標本平均、標本自己相関、標本共分散の計算、コレログラムの作成
5. 自己相関の検定、Ljung-Box 統計量の計算
6. AR, MA, ARMA モデルの推定
7. AR, MA, ARMA モデルの次数の選択(AIC, BIC の計算)
8. AR, MA, ARMA モデルによる予測
9. R による日次データの取り扱いについて
10. AIC と BIC の計算について

について説明する。

0. R をインスツールする。

(学校の PC にはすでにインスツールしてある)自分の PC などに統計ソフト R をインスツールするには

(Windows 版) https://cran.r-project.org/ に行き、「Download R for Windows」→「base」→「Download R x.y.z for Windows」(ここで x, y, z は何か数字が入る。例えば 4.01.など)をクリックして、インスツーラーをダウンロード(どこかに保存して実行)。
(Mac 版)多少複雑。http://aoki2.si.gunma-u.ac.jp/R/begin.html を参照。

1. データを読み込む

以下では `read.table()` 関数によるデータの読み込みについて説明する。

1.1 データの用意

テキスト形式のファイルを用意する(拡張子が.txt のファイル)。以後では tsdata.txt というファイルが用意してあるものとして話を進める(講義のホームページのところにあります)。tsdata.txt ファイルをメモ帳で開くと

```
##  左から日付(月と西暦下 2 桁)、TOPIX、実効為替レート、鉱工業生産指数)
date,      topix,      exrate,   indprod
Jan-75,    276.09,    29.13,     47.33
Feb-75,    299.81,    29.7,      46.86
Mar-75,    313.5,     29.98,     46.24
        …
```

のようになっている。1 行目にコメント、2 行目に変数の名前が入っており、3 行目からデータが始まっている。

1.2 作業ディレクトリの変更

R の画面のメニュー・バーから「ファイル」→「ディレクトリの変更」によってデータ(tsdata.txt)が置いてあるディレクトリを指定する。確認のため

```
> dir()
```

[†]この資料は私のゼミおよび講義で R の使用法を説明するために作成した資料です。ホームページ上で公開しており、自由に参照して頂いて構いません。ただし、内容について、一応検証してありますが、間違いがあるかもしれません。間違いがあった場合でもそれによって生じるいかなる損害、不利益について責任は負いかねますますのでご了承下さい。

と入力すると、現在の作業ディレクトリにあるファイルが全て表示されるので、そこに tsdata.txt があるか確認する。

1.3 read.table( ) 関数による読み込み

次のコマンドを実行する

```
> tsdata = read.table("tsdata.txt", header=TRUE, skip= 1)
```

3 番目の引数は 1 行目のコメントデータを読み込まない(skip する)事を R に知らせている(skip = k で最初の k 行を読み込まないようにする)。2 番目の引数 header=TRUE は実際に読み込むことになる最初の行に、各データの名前(data, topix, exrate,..等の変数名)がヘッダーとして記述されていることを R に知らせるための指示である(もし、各データ系列の名前(変数名)がなく、数字の実データから始まっていれば、 header=FALSE とする)。これにより tsdata という名前のデータのセット(正確には R ではデータフレームという)が作成される。 (tsdata とコマンドすると確認できます)。

```
> head(tsdata, 5)
```

と入力すると(head(変数名,k) というのはその変数の最初の k 行を表示する関数)

```
  date    topix exrate indprod
1 Jan-75 276.09  29.13   47.33
2 Feb-75 299.81  29.70   46.86
3 Mar-75 313.50  29.98   46.24
4 Apr-75 320.57  29.80   47.33
5 May-75 329.65  29.79   47.33
```

のように最初の 5 行目まで出力される。date が上から下に行くにつれて新しくなっている事に注意。以下に述べる方法は全てデータがこのように並べてある事を想定している。

2. データをプロットする

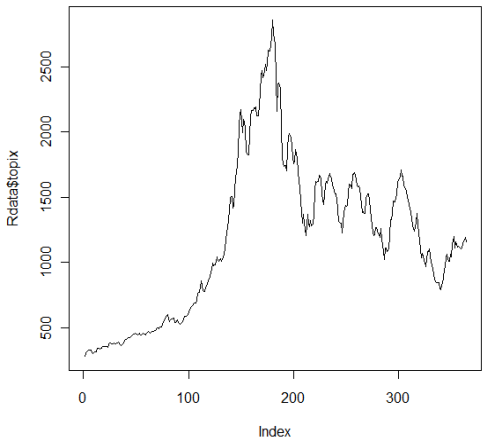
以下では plot( ) 関数によるデータのプロットの仕方について説明する。

2.1 plot( ) 関数によるデータのプロット

ここでは tsdata の topix をプロットしてみよう。

```
> plot(tsdata$topix,type="l")  ("l" は 小文字のエルである事に注意)
```

以下のようなグラフが出力される。ちなみに"l"というのは line の l でグラフを実線で書くことを意味している。このほかには"p"などとすると、(これは point の p) 点によってプロットされたグラフが出力される。どのようなグラフが出力可能かは help(plot) で(オンライン上で)確認できる。またコマンド help(関数名) によってその関数に関するヘルプをオンライン上で見る事もできる。



2.2 ts() 関数による時系列オブジェクトの作成

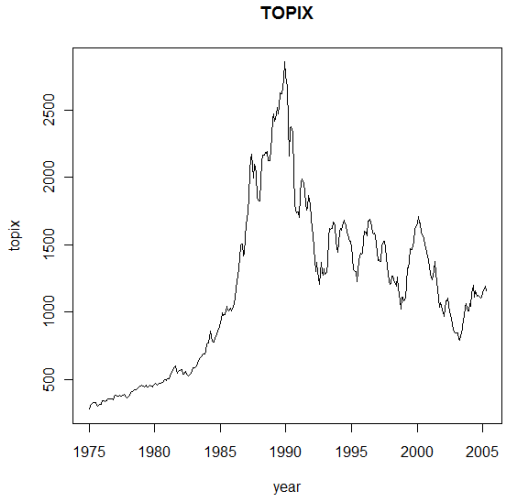
上のグラフの X 軸は単純にデータの番号を表している。ここで topix は時系列データなので、X 軸もそれに対応して年、月、日などで表示したい。このような場合 ts() 関数を用いて、データが時系列データである事を R に認識させる。

```
> topix = ts(tsdata$topix, start=c(1975,1), frequency=12)
```

と入力する。これはこの topix のデータが月次データである事を R に認識させるコマンドである。start=c(1975,1) はデータが 1975 年の 1 月から始まる事を意味している。frequency=12 はデータが月次のデータである事を示している(年次データであれば frequency=1, 4 半期データであれば frequency=4 とする)。再び plot( ) 関数によって

```
> plot(topix,type = "l", xlab="year", main="TOPIX")
```

でグラフを描くと、今回は以下のようなグラフが出力される。xlab="year"という引数は X 軸のラベルを year にするという事(Y 軸ラベルの変更は ylab="…" で行う)。main="TOPIX" はグラフタイトルを TOPIX にするという事)。



3. 変化率の計算

3.1 変化率の定義による計算

ある変数 x_t の時点 $t-1$ から時点 t にかけての変化率 r_t は

$$r_t = \frac{x_t - x_{t-1}}{x_{t-1}} \times 100$$

と定義される(単位を%にするために 100 を掛けている)。R を用いてこれを計算する。時系列オブジェクトの差をとる diif( )関数と期をずらした「ラグ」をとる関数の lag( ) 関数を用いる。

```
> topixrate1 = diff(topix)/lag(topix, k=-1)*100
```

と入力する(plot(topixrate1,type="l")でプロットしてみよう)。

3.2 対数階差による計算

変化率は(自然)対数の階差によって(実用上十分な精度で)近似することができる。ある変数 x_t の時点 $t-1$ から時点 t にかけての変化率 r_t は

$$r_t = (\log x_t - \log x_{t-1}) \times 100$$

で近似できる。R を用いてこれを計算する。対数は log( ) 関数によって計算できる。

```
> topixrate2 = diff(log(topix))*100
```

と入力する(plot(topixrate2,type="l")でプロットしてみよう)。

3.3 図を並べて表示する、図を重ねて表示する

先程の topixrate1 と topixrate2 を並べてプロットするには `par(.)` 関数を用いる。まずプロット画面を上と下に 2 つに分割する:

```
> par(mfcol=c(2,1))
```

(`par(mfcol=c(n,m))` でプロット画面を n 行 m 列に分割する)。次に topixrate1 と topixrate2 をプロットする

```
> plot(topixrate1, type="l")
> plot(topixrate2, type="l")
```

図が並べて表示される。

次に topixrate1 と topixrate2 を重ねてプロットしてみよう。まず先程 2×1 に分割したものをもとの 1×1 に戻す:

```
> par(mfcol=c(1,1))
```

次に topixrate1 をプロットする。

```
> plot(topixrate1,type="l",ylab="rate",ylim=c(-20,20))
```

ここで `ylim =c(a, b)` は Y 軸の範囲を –20 から 20 に制限することを意味する(X 軸の制限は `xlim=c(a,b)` で行える)。この図に上書きするには

```
> par(new=T)
```

とする。ここで topixrate2 のプロットを上書きするとこの 2 つはほとんど等しいので重なってしまって区別がつかないので、topixrate2 に 5 を足したものをプロットする

```
> plot(topixrate2+5,type="l",col=2,ylab="rate",ylim=c(-20,20))
```

ここで `col=2` はプロットした線の色を赤にするためのものである(数値を変えると色が変わる)。

4. 標本平均、標本自己相関、標本共分散の計算

4.1 標本平均の計算

R で標本平均を計算するには `mean()` 関数を使う。`topixrate2` の平均を計算するには

```
>mean(topixrate2)
```

と入力する。結果は

```
> mean(topixrate2)
 [1] 0.3950173
```

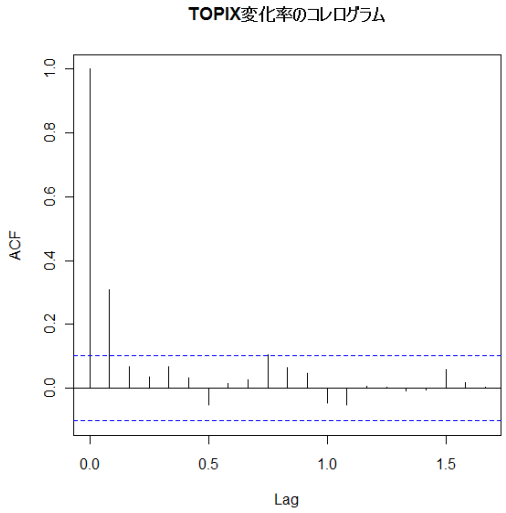
となる。

4.2 標本自己相関の計算、コレログラムの作成

R で標本自己相関を計算するには `acf()` 関数を使う。`topixrate2` の標本自己相関を計算するには

```
> acf(topixrate2,lag.max=20,main="TOPIX 変化率のコレログラム")
```

と入力する。コレログラムが自動的に出力される(点線は 95%信頼区間である)。



lag.max=20 は 20 次までの自己相関を計算するという事である。計算結果を変数として保存するには

```
> acftopixrate2=acf(topixrate2,lag.max=20)
```

と入力する。結果を見るには

```
> acftopixrate2= acftopixrate2$acf
```

と入力する(これは同時に計算結果に acftopixrate2 という名前を付けたという事)。結果は

```
> head(acftopixrate2, 6)
[1] 1.00000000 0.30779994 0.06824984 0.03529433 0.06801588 0.03130508
```

となる。

4.3 標本自己共分散を計算する

再び acf() 関数を用いて

```
> acf(topixrate2,type="covariance",lag.max=20,main="TOPIX 変化率の共分散")
```

と入力する。計算結果を変数として保存するには

```
> acvftopixrate2=acf(topixrate2,type= "covariance",lag.max=20)
```

とすればよい。さらに

```
> acvftopixrate2=acvftopixrate2$acf
```

とし、標本自己共分散に acvftopixrate2 という名前を付けておく。結果は

```
> head(acvftopixrate2,6)
[1] 16.9843193  5.2277724  1.1591771  0.5994501  1.1552034  0.5316956
```

のようになる。

(以下は同じことを違うやり方でやったものです。以下は単に練習のためですので、普段は上記の方法でやって下さい。)
標本自己共分散を計算するには標本自己相関に 0 次の自己共分散(つまり定常分散)の推定値を掛ければよい。R で標本分散を計算するには var( )関数を使う。topixrate2 の標本分散を計算するには

```
> var(topixrate2)
```

と入力する。結果は

```
> var(topixrate2)
```

```
[1] 17.03124
```

となる。標本分散は $T-1$ で割っているので(T は観測数)、$(T-1)/T$ を掛けなくてはならない。T を調べるには `length( )` 関数を用いる。`topixrate2` の観測数を調べるには

```
> length(topixrate2)
```

と入力する。363 となるはずである。これらより標本自己共分散は

```
acvftopixrate2 = (362/363)*var(topixrate2)*acftopixrate2
```

により計算される (“ * ”は掛け算を意味するコマンド) (これは同時に計算結果に `acvftopixrate2` という名前も付けている)。結果は

```
> head(acvftopixrate2, 6)
[1] 16.9843193  5.2277724  1.1591771  0.5994501  1.1552034  0.5316956
```

となる。

5 自己相関の検定、Ljung-Box 統計量の計算

自己相関の検定は先ほどのコレログラムの図において 95%信頼区間の外に出ていれば自己相関 0 の帰無仮説を(5%有意水準で)棄却できる。

R で Ljung-Box 検定を行うには `Box.text()` 関数を用いる。例えば `topixrate2` の Ljung-Box 統計量 $Q(20)$を計算するには

```
> Box.test(topixrate2, lag=20, type="Ljung-Box")
```

と入力する。ここで `lag= 20` とは Ljung-Box 統計量 $Q(m)$において $m = 20$ とするという事である。すると出力結果として

```
        Box-Ljung test

data:  topixrate2
X-squared = 50.1781, df = 20, p-value = 0.0002088
```

のように出力される。`X-squared` が Ljung-Box 統計量の値である。$Q(m)$は帰無仮説のもとで自由度 m のカイ二乗分布に従う。`p-value` とは `X-squared` の値の帰無仮説のもとでのパーセント点でありこれが 0.05 以下であれば帰無仮説を棄却できる。

6. AR, MA, ARMA モデルの推定

R で ARMA モデルを最尤推定すには `arima()` 関数を用いる。`topixrate2` に ARMA(2, 1) モデルを当てはめた最尤推定は

```
> tr2arma21= arima(topixrate2, order=c(2,0,1))
```

で行う事ができる(ちなみに `tr2arma21` の `tr` は topix rate の略。この名前は好きなように決められる)。ここで `order=c(2,0,1)` の 2 と 1 はそれぞれ AR 部分の次数と MA 部分の次数を表している(ARMA モデルの推定では 2 番目の引数は常に 0 にする)。R の `arima()` 関数は正確な尤度関数に基づいて推定している(条件付き最尤推定ではない)。推定結果として

```
>tr2arma21
  Call:
  arima(x = topixrate2, order = c(2, 0, 1))

  Coefficients:
         ar1     ar2     ma1  intercept
      0.1530  0.0207  0.1674     0.3996
s.e.  0.8186  0.2620  0.8168     0.2902

  sigma^2 estimated as 15.34:  log likelihood = -1010.72,  aic = 2031.43
```

と出力される。2 列目がそれぞれのパラメータの出力結果(intercept とは切片の事)、3 列目が標準誤差である。siguma^2 とは攪乱項(ε_t)の推定値、log likelihood は対数尤度の値 aic は AIC の値である。AR、MA モデルを推定するにはそれぞれ MA の次数や AR の次数を 0 にすればよい。以下は topixrate2 の AR(1)モデルの推定結果である。

```
> tr2ar1 = arima(topixrate2,order=c(1,0,0))
> tr2ar1

  Call:
  arima(x = topixrate2, order = c(1, 0, 0))

  Coefficients:
          ar1  intercept
       0.3106     0.401
  s.e. 0.0501     0.298

  sigma^2 estimated as 15.36:  log likelihood = -1010.9,  aic = 2027.79
```

7. AR, MA, ARMA モデルの次数の選択(AIC, BIC の計算)

AIC は ARMA モデルの推定結果として自動的に出力される。BIC は

$$AIC = -2k + \log(T)k$$

で計算することができる(AIC や BIC の定義は講義スライドで確認)。ここで T は標本数、k は推定したパラメーターの数である(これは AR と MA の次数の和+2 で計算される。AR(1)モデルであれば AR(1)の係数と切片と攪乱項の分散の 3 つ)。先ほどの ARMA(2,1)モデルと AR(1)モデルについて AIC は推定結果の tr2arma21 と tr2ar1 に既に含まれている(推定結果の出力のところにも出ているが)。これらは

```
> tr2arma21aic = tr2arma21$aic   (ARMA(2, 1) モデルの AIC)
> tr2ar1aic = tr2ar1$aic         (AR(1) モデルの AIC)
```

によって別個に取り出すことができる。答えはそれぞれ

```
> tr2arma21aic
[1] 2031.43
> tr2ar1aic
[1] 2027.791
```

となる。BIC を計算するには

```
> tr2arma21bic = tr2arma21$aic-2*5+log(363)*5   (ARMA(2,1)モデルの BIC)
> tr2ar1bic = tr2ar1$aic-2*3+log(363)*3         (AR(1)モデルの BIC)
```

と入力すればいい(公式参照: ARMA(p, q) モデルの場合は aic から 2*(p+q+2) を引いて log(T)*(p+q+2) を足せばよい。ここで T はサンプル数)。答えはそれぞれ

```
> tr2arma21bic
[1] 2050.902
> tr2ar1bic
[1] 2039.474
```

となっている。

8. AR, MA, ARMA モデルによる予測

R で AR, MA, ARMA モデルによる h 期先予測を行うには predict( )関数を用いる。topixrate2 に ARMA(2, 1)モデルの推定結果を用いて(データの終点から)20 期先予測を行うには

```
> tr2arma21.pred = predict(tr2arma21,n.ahead=20)
```

と入力する。予測値に tr2arma21hat と名前を付けるには

```
> tr2arma21hat = tr2arma21.pred$pred
```

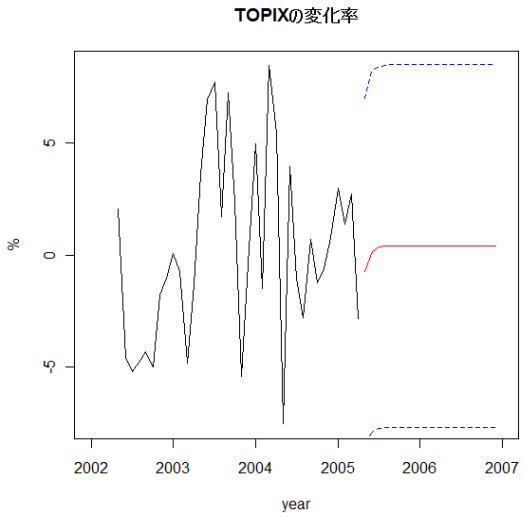
と入力する。topixrate2 の直近 3 年(36 個)の観測値とそれ以降の予測値をプロットするには
(データの終点が 2005 年の 4 月なので)

```
> topixrate2.3y = ts(topixrate2[328:363],start=c(2002,5),frequency=12)
> plot(topixrate2.3y,type="l",xlim=c(2002,2007),ylab="%", xlab="year",
main="TOPIX の変化率")
> lines(tr2arma21hat,lty=1,col=2)
```

と入力する(lines() は図に線を書き込んでいくコマンド。lty は線の種類、col は線の色を指定する。ここの番号を変えるといろいろな線の種類と色が使用できる。上記はそれぞれ 1(実線) と 2(赤)を 使用)。ここで topixrate2.3y が直近 3 年間のデータである。信頼区間をプロットするには

```
> sig =tr2arma21.pred$se
> tr2arma21hatL = tr2arma21hat-1.96*sig
> lines(tr2arma21hatL, lty=1, col=4)
> tr2arma21hatU = tr2arma21hat+1.96*sig
> lines(tr2arma21hatU, lty=1, col=4)
```

と入力する。ここで sig は予測の標準誤差を計算している。下のような図が出力される。



9. R による日次データの取り扱いについて

日次データを R で取り扱うのは、プロット以外は特にやっかいな事はない。例えば nikkei2008.txt にあるデータであれば

```
> nikkei2008 = read.table("nikkei2008.txt",header=T,skip=1)
```

と読み込み、head() 関数で確認すると

```
> head(nikkei2008,3)
      日付      終値
1  2008/1/4 14691.41
2  2008/1/7 14500.55
3  2008/1/8 14528.67
```

となっている。このうち使うのは終値のデータだけなので

```
> nikkei = nikkei2008$終値
```

とすれば 終値だけを取り出せて

```
> head(nikkei,10)
 [1] 14691.41 14500.55 14528.67 14599.16 14388.11 14110.79 13972.63 13504.51
 [9] 13783.45 13861.29
```

となるので、あとはこの nikkei (の対数階差による収益率)に対して自己相関、相関を計算していけばよい。例えば、対数階差の収益率は)

```
> nikkeirate=diff(log(nikkei))*100
> head(nikkeirate,5)
[1] -1.3076390  0.1937359  0.4840054 -1.4561822 -1.9462418
```

自己相関は

```
> acfnikkeirate=acf(nikkeirate,lag.max=20)
```

などで計算できる。

上でみたようにデータの分析自体は日次データでも同じであるが、株価のような日次データをプロットするのはなかなかにやっかいである。というのは株価のような日次データは土日や祝日が観測されないので(土日がないだけなら、わりと簡単に対処できるが、祝日の存在が厄介) ts() 関数で時系列オブジェクトにするときに frequency がうまく設定できず、プロットすると微妙に日にちと観測データがずれてしまうのである。プロット自体は

```
> plot(nikkei2008$終値,type="l")
```

とすればできるが、*x* 軸のメモリが日次にならない。

10. AIC と BIC の計算について

AIC と BIC の計算をする際、例えば AR と MA の時数を最大 5 までとして計算するとしよう。しかしながら AR と MA の次数を最大 5 とした場合、可能な組み合わせとして、それぞれの次数が 0, 1, …, 5 と 6 つあるので、6 × 6 = 36 と合計 36 回も異なった次数の ARMA モデルを推定しなければならず、大変である。これに対処する方法として R の「for 関数」を使うと推定をいっぺんにやる事ができ、便利である。以下 for 関数について説明する。

TOPIX の収益率のデータが rate という変数として既に R に認識されているとしよう(以下では topixrate2 に rate という名前を付けたとする(> rate = topxirate2 とすればよい)。この時このデータに対して ARMA(*p*, *q*) モデルを推定するには(結果を armapq という名前を付けて保存するとして)

```
> armapq = arima(rate, order=c(p, 0, q))
```

とすればよい(p と q は具体的な数字を入れて下さい)。またこの場合の AIC は

```
> (aic=armapq$aic)
```

とすれば出力される(括弧をつけると計算結果がすぐに出力される)。これを 36 回繰り返すのは大変である。よって例えば AR の次数をとりあえず $p = 0$ に固定して、MA の次数だけを $q = 0$ から 5 まで変化させて推定し、その AIC を表示したいときには(コマンドが少し長くなるが)

```
 > p=0; for (q in 0:5){armapq=arima(rate,order=c(p,0,q));
aic=armapq$aic; cat("p =",p, "q =",q, "aic =", aic, "\n")}
```

と打ち込んでエンターキーを押せばよい(ここでの¥は実際に打ち込むとバックスラッシュで表示されるだろう)。すると

```
 p = 0 q = 0 aic = 2062.271
 p = 0 q = 1 aic = 2029.069
 ...
 p = 0 q = 5 aic = 2033.102
```

のように出力される。ここで $p = 0$ を $p = 1$ にすれば p を 1 に固定して、q を変化させて計算する。また $p = 0$ から 5 まで変化させたものもいっぺんに出力したい場合は

```
 > for (p in 0:5) for (q in 0:5){armapq=arima(rate,order=c(p,0,q));
aic=armapq$aic; cat("p =",p,"q =",q,"aic =",aic, "\n")}
```

とする事もできる(これは結果が出力されるまで少し時間がかかるかもしれない。また警告メッセージが出ると思うが、これは気にしないでよい)。さらに BIC も同時に計算して出力したい場合は

```
 > T=length(rate)
 > for (p in 0:5) for (q in 0:5){armapq=arima(rate,order=c(p,0,q));
aic=armapq$aic; bic=aic-2*(p+q+2)+log(T)*(p+q+2); cat("p =",p, "q =",q, "aic
=", aic, "bic =",bic, "\n")}
```

とすればよい(これもおそらく出力まで時間がかかる)。ここで `length(x)` は `x` のデータの個数を返す R の関数である。