クレジット:
UTokyo Online Education 統計データ解析 Ⅱ 2018 小池祐太

東京大学 THE UNIVERSITY OF TOKYO

CC BY NC ND

統計データ解析 (II) 第 3 回

小池祐太

2017 年 4 月 26 日

行列とその演算: 積

- ベクトルの場合と同様に, サイズが同じ 2 つの行列 `A` と `B` に対して, `A*B` は成分ごとの積 (Hadamard 積)

$$(A \circ B)_{ij} = a_{ij} b_{ij} \tag{1}$$

  を計算する
- 一方で, A が $n \times m$ 行列, B が $m \times l$ 行列のとき, 通常の意味での行列に対する積 AB

$$(AB)_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj} \tag{2}$$

  が定義されるが (AB は $n \times l$ 行列), こちらは `A%*%B` で計算できる
- 実行例 `matrix-prod2.r`

行列とその演算: 初等関数の適用

- ベクトルの場合と同様, 行列に初等関数 (sin, exp, . . . など) を適用すると, 成分ごとに計算した結果が返される
- 例えば, 行列 A に関数 sin を適用した結果 `sin(A)` は, (i, j) 成分が

$$\sin(a_{ij})$$

で与えられる, 行列 A と同じサイズの行列となる
- 実行例 `matrix-fun.r`

行列とその演算: 行列式とトレース

- 連立 1 次方程式

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

は, n 次正方行列 A, n 次元列ベクトル $\boldsymbol{x}$ および $\boldsymbol{b}$ をそれぞれ

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

で定めると, 行列方程式

$$A\boldsymbol{x} = \boldsymbol{b} \tag{3}$$

の形で書くことができる

行列とその演算: 行列式とトレース

- 方程式 (3) の解の存在や一意性を議論する上で, 行列 A の**行列式 (determinant)** と呼ばれる量が重要な役割を果たす
- 行列 A の行列式は記号 $\det A$ で表される
- 一般の行列に対する行列式の定義は複雑なためここでは割愛するが, A が 2 次正方行列の場合は単純であり,

$$\det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

  となる
- R においては, 行列式は関数 `det()` を用いて計算することができる

行列とその演算: 行列式とトレース

- 正方行列の**トレース (trace)** は対角成分 (対角線上の成分) の総和として定義される
- 正方行列 A のトレースは記号 $\mathrm{tr}\, A$ で表されることが多い:

$$\mathrm{tr}\, A = \sum_{i=1}^{n} a_{ii}.$$

- R においては, トレースを計算するための専用の関数は用意されていないが, 対角成分を取り出す関数 `diag()` とベクトルの成分の総和を計算する関数 `sum()` を組み合わせて

```
sum(diag(A))
```

のように計算できる

- 実行例 `matrix-det3.r`

行列とその演算: 逆行列

- 対角成分以外の成分がすべて 0 であるような正方行列を**対角行列 (diagonal matrix)** と呼ぶ
- 特に, 対角成分がすべて 1 であるような対角行列を**単位行列 (identity matrix)** と呼ぶ
- 以下 n 次単位行列を記号 E_n で表す
- R では, n 次単位行列は

  `diag(n)`

  で生成することができる

行列とその演算: 逆行列

- n 次正方行列 A に対して, n 次正方行列 B が

$$AB = BA = E_n$$

を満たすとき, B を A の**逆行列 (inverse matrix)** と呼び, 記号 A^{-1} で表す (逆行列は存在すれば一意的に定まる)
  - 逆行列をもつような正方行列は**正則 (regular)** もしくは**非特異 (non-singular)** であるという
  - 逆行列は常に存在するとは限らない. 正方行列 A が正則である (つまり, 逆行列をもつ) ための必要十分条件は, A の行列式 $\det A$ が 0 でないことであるという事実が知られている
- 正則行列の逆行列を求めるには関数 `solve()` を用いる (正則でない行列に適用するとエラーとなる)

行列とその演算: 一般化逆行列

- 一般に A が (正則とも正方とも限らない) 行列の場合でも, 逆行列に類する概念として**一般化逆行列 (generalized inverse matrix)** を考えることができ, データ解析においてしばしば必要となる
- A の一般化逆行列とは,

$$AA^{\dagger}A = A \tag{4}$$

を満たす行列 $A^{\dagger}$ のことである

行列とその演算: 一般化逆行列

- 一般に A の一般化逆行列は一意には定まらないが, 以下の条件を満たす一般化逆行列 A^+ は一意的に定まることが知られている

$$AA^+A = A \tag{5}$$
$$A^+AA^+ = A^+ \tag{6}$$
$$(AA^+)^\top = AA^+ \qquad (* \text{ は転置行列の意}) \tag{7}$$
$$(A^+A)^\top = A^+A \tag{8}$$

- A^+ は A の **Moore-Penrose の一般化逆行列**と呼ばれている
- Moore-Penrose の一般化逆行列は, **MASS** パッケージの関数 `ginv` によって計算できる
- 実行例 `matrix-inv3.r`

ベクトルと行列の計算: 積

- R 言語においては，列ベクトル・行ベクトルという区別はなく，単一のベクトルという概念で扱われている
- このため, 行列とベクトルの積においては, 行列のどちらからベクトルを掛けるかによって自動的に列ベクトルか行ベクトルか適切な方で扱われる
- ただし, ベクトルも行列の一種であるから, 計算結果は行列として表現されることに注意する
- 実行例 `linear-calc2.r`

ベクトルと行列の計算: 連立方程式

- A が n 次正則行列, $\mathbf{b}$ が n 次元列ベクトルのとき, $\mathbf{x}$ に関する連立 1 次方程式

$$A\mathbf{x} = \mathbf{b}$$

  は解 $\mathbf{x} = A^{-1}\mathbf{b}$ をもつが, この解は関数 `solve` によって計算できる
- 書式

```
solve(A, b)
```

- なお, $\mathbf{b}$ はベクトルのかわりに行数 n の行列でも可
- 実行例 `linear-eq2.r`

固有値と固有ベクトル

- 一般に n 次正方行列 A に対して, 複素数 λ と零ベクトルでない n 次元ベクトル $\mathbf{x}$ が

$$A\mathbf{x} = \lambda\mathbf{x}$$

を満たすとき, λ を A の**固有値 (eigenvalue)**, $\mathbf{x}$ を λ に対する**固有ベクトル (eigenvector)** と呼ぶ

- 正方行列 `A` の固有値と固有ベクトルは `eigen(A)` で計算できる. 結果は固有値と固有ベクトルを列ベクトルとする行列からなるリストで返される

固有値と固有ベクトル

- A の転置行列 $A^\top$ が A 自身に一致する場合 ($A^\top = A$), A は**対称 (symmetric)** であるという
- A が対称ならば, A の固有値はすべて実数であり, かつ A の固有ベクトルを n 個並べて得られる正則行列 V で, $V^{-1}AV$ が対角行列となるようなものがとれることが知られている (この場合 $V^{-1} = V^\top$ となるようにとれる)
- この操作を A の**対角化 (diagonarization)** と呼ぶ
- 実行例 `eigen.r`

関数定義

- 多くの計算機言語と同様, R でもユーザー独自の自作関数を定義できる
- 自作関数の定義には関数 `function` を利用する
- 例 半径 r から球の体積と表面積を求める関数 `myfunc`

```
myfunc <- function(r){
    V <- (4/3) * pi * r^3 # 球の体積
    S <- 4 * pi * r^2 # 球の表面積
    out <- c(V,S)
    names(out) <- c("volume", "area")
    # 返り値に名前をつける
    return(out)
}

myfunc(1) # 実行
```

制御文

- 一般に最適化や数値計算などを行うためには, 条件分岐や繰り返しを行うための仕組みが必要となる
- R 言語を含む多くの計算機言語では `if` (条件分岐), `for`・`while` (繰り返し) を用いた構文が用意されているが, これを制御文と言う
- 実行例 `control2.r`

制御文

- `if` 文: 書式

$$\texttt{if(条件 A)\{ プログラム X\}}$$

  - 条件 A が満たされる場合, プログラム X を実行する
  - 「条件 A が満たされない場合, 代わりにプログラム Y を実行する」としたければ, 以下のように書く:

$$\texttt{if(条件 A)\{ プログラム X\}else\{ プログラム Y\}}$$

- `for` 文: 書式

$$\texttt{for(i in M)\{ プログラム X\}}$$

  - `M` をベクトルとし, 変数 `i` が `M` の要素となる値すべてを動きながらプログラム X を繰り返し実行する
  - プログラム X は通常変数 `i` によって実行内容が変わる

制御文

- `while` 文: 書式

$$\texttt{while(条件 A)\{ プログラム X\}}$$

  - 条件 A が満たされる限り, プログラム X を繰り返し実行する
  - プログラム X は通常実行するたびに実行内容が変わり, いつか条件 A が満たされなくなるように書く

データの抽出

- データから必要な部分集合を取り出すためには, 添え字を指定するのが最も基本的な方法
- 添え字の指定の仕方には, 番号を指定する以外に, 論理値で指定する方法がある
  - TRUE は要素の「選択」を, FALSE は要素の「除外」を意味する
- 要素に名前が付けられている場合は, その名前によってアクセス可能
- また, マイナス記号をつけて添え字番号を指定すると, その添え字番号の要素を除外する
- 実行例 `select.r`

データの抽出

- データフレームから必要な部分集合を取り出す際に複雑な条件を指定する場合, 添え字を指定するのではコードが読みにくくなってしまう
- そのような場合にも対応できるように, 関数 `subset()` が用意されている
- 関数 `subset()` の基本書式

```
subset(x, subset, select, drop = FALSE)
```

  - `x`: データフレーム
  - `subset`: 抽出したい行に関する条件
  - `select`: 抽出したい列に関する条件 (未指定の場合はすべての列が抽出される)
  - `drop`: 結果が 1 行もしくは 1 列のデータフレームになる場合に, 結果をベクトルとして返すか否か

- 実行例 `subset.r`

ファイルを用いたデータの読み書き

- 実際の解析の過程においては，収集されたデータを読み込んだり，整理したデータを保存したりする必要が生じる
- R では一般に用いられる CSV 形式 (comma separated values) のテキストファイルと，R の内部表現を用いたバイナリーファイル (ここでは RData 形式と呼ぶ) をサポートしている
- 以下では，データフレームを対象として，それぞれの形式でファイルの読み書きを行うための関数を纏める

作業ディレクトリの確認と変更

- R の実行は特定のフォルダ (ディレクトリ) 上で行われており, そのフォルダを**作業ディレクトリ**と呼ぶ
- R のコード内でファイル名を指定した場合, 特に指定しない限り作業ディレクトリに存在するものとして扱われる
- 現在の作業ディレクトリは, RStudio のコンソールの上部, もしくは

  ```
  getwd()
  ```

  で確認できる

作業ディレクトリの確認と変更

- 作業ディレクトリの変更には関数 `setwd()` を利用するか, RStudio 上部の「Session」という項目から「Set Working Directory」を選び, その中の「Choose Directory...」という項目を選択すれば, 変更後のフォルダを選択できるようになる
- 実行例 `getwd.r`

ファイルを用いたデータの読み書き

- 1 つのデータフレームを CSV 形式のファイルへ書き出すには，関数 `write.csv()` を用いる
- 基本書式

```
write.csv(x, file = "")
```

  - `x`: 書き出したいデータフレーム
  - `file`: 書き出すファイルの名前
  - 他にも細かいオプションあり．ヘルプ参照
- ファイルの保存先は (指定しない限り) 作業ディレクトリとなる
- 実行例 `data-write.csv.r`

ファイルを用いたデータの読み書き

- CSV 形式のファイルから読み込むには，関数 `read.csv()` を用いる
- 基本書式

```
read.csv(file, header = TRUE, row.names)
```

  - `file`: 読み込みたいファイルの名前 (作業ディレクトリ下にある必要あり．もしくはディレクトリも指定)
  - `header`: ファイルの 1 行目をデータフレームの列名として使うか否か?
  - `row.names`: データフレームの行名を指定．(i) 行名を含む列番号/列名を指定, (ii) 行名の直接指定, というオプションがある．デフォルトでは行番号がそのまま行名になる．
  - 他にも細かいオプションあり．ヘルプ参照
- なお, より一般のテキストファイルを読み込むための関数として `read.table()`, `scan()` などがある

ファイルを用いたデータの読み書き

- 以降の講義の実行例で利用するデータ `kikou2016.csv` は, 以下の Web ページ

  `https://elf-c.he.u-tokyo.ac.jp/courses/228`

  からダウンロードできる
- 実行例 `data-read.csv2.r`

ファイルを用いたデータの読み書き

- RData 形式のファイルへの書き出しは，関数 `save()` を用いる
- CSV 形式と異なり，複数のデータフレームを 1 つのファイルに同時に保存することもできる
- 基本書式

```
save(..., file)
```

  - `...`: 保存したいオブジェクト名 (複数可, データフレーム以外も可)
  - `file`: 書き出すファイルの名前

- ファイルの保存先は (指定しない限り) 作業ディレクトリとなる
- 実行例 `data-save.r`

ファイルを用いたデータの読み書き

- RData 形式のファイルからの読み込みは，関数 `load()` を用いる
- 基本書式

```
load(file)
```

  - `file`: 読み込みたいファイルの名前 (作業ディレクトリ下にある必要あり．もしくはディレクトリも指定)

- 実行例 `data-load.r`

記述統計量によるデータの要約

- データ解析の出発点は, 与えられたデータ全体の特徴や傾向を把握することである
- そのための基本的な方法の 1 つは, データの特徴を適切に表す統計値を計算することである
- そのような統計値を記述統計量, 要約統計量もしくは基本統計量と呼ぶ

記述統計量によるデータの要約

- N 個のデータ $x_1, x_2, \ldots, x_N$ が与えられたとき, それらを代表する値として, **平均 (mean)**

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i = \frac{x_1 + x_2 + \cdots + x_N}{N}$$

  が頻繁に利用される
- 平均は R では関数 `mean()` で計算できる

記述統計量によるデータの要約

- また, データのばらつき具合の指標として, **分散 (variance)**

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_N - \bar{x})^2}{N-1}$$

およびその平方根である**標準偏差 (standard deviation)**

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

が広く利用されており, それぞれ関数 `var()` および関数 `sd()` で計算できる

記述統計量によるデータの要約

- データの順位にもとづく記述統計量もよく利用される
- 例えば, $x_1, \ldots, x_N$ の**最大値 (maximum)** は関数 `max()` で, **最小値 (minimum)** は関数 `min()` でそれぞれ計算できる

記述統計量によるデータの要約

- データを
$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}$$
のように昇順に並べ替えた際に中央の位置にくる値を**中央値**または**メディアン (median)** と呼ぶ
- N が奇数の場合, 中央値は $x_{((N+1)/2)}$ であり, N が偶数の場合は $(x_{(N/2)} + x_{(N/2+1)})/2$ である
- 中央値は関数 `median()` で計算できる
- 中央値は平均と同様データを代表する値だと考えられるが, 平均と比較して, 計算結果がデータに含まれる異常な値 (**外れ値**と呼ばれる) の影響を受けにくい

記述統計量によるデータの要約

- 中央値の一般化として, $\alpha \in [0, 1]$ に対して, その点以下のデータの個数が全体の**約** $100\alpha\%$になるような点を $100\alpha\%$**分位点 (quantile)** と呼ぶ
- 特に 25%分位点および 75%分位点をそれぞれ**第 1 四分位点**, **第 3 四分位点**と呼ぶ
  - **第 2 四分位点**は 50%分位点となるが, これは中央値のことである
- ベクトル `x` の 100`alpha`%分位点は `quantile(x, alpha)` で計算できる
- 分位点は一意的には定まらず, いくつかの計算方式がある: `help(quantile)` 参照
- 実行例 `descriptive.r`

記述統計量によるデータの要約

- データフレームが与えられた際には, 列 (あるいは行) ごとに記述統計量を計算したい状況が頻繁にある
- そのような計算に便利な関数として関数 `apply()` がある. 関数 `apply()` は基本的に以下のような書式で利用する:

```
apply(X, MARGIN, FUN)
```

  - `X`: データフレーム
  - `MARGIN`: 行ごとの計算には 1 を, 列ごとの計算には 2 を指定
  - `FUN`: 求めたい統計量を計算するための関数
- なお, データフレーム `x` に対して `summary(x)` を実行することで, 列ごとの最小値, 第 1 四分位点, 中央値, 平均, 第 3 四分位点, 最大値がそれぞれ計算される
- 実行例 `apply.r`

記述統計量によるデータの要約

- 複数のデータが与えられた場合, それらのデータの間の関係性を知りたい場合が頻繁に生じる
- そのような目的のための最も基本的な記述統計量に**相関 (correlation)** がある
  - 2 種類のデータ間の比例関係の大きさを計測
- 2 種類のデータ $x_1, x_2, \ldots, x_N$ および $y_1, y_2, \ldots, y_N$ に対して, それらの相関は

$$\rho = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{9}$$

で定義される ($\bar{x}$ および $\bar{y}$ はそれぞれ $x_1, x_2, \ldots, x_N$ および $y_1, y_2, \ldots, y_N$ の平均)

記述統計量によるデータの要約

- 相関は -1 以上 1 以下の値をとり, 1 に近いほど正の比例関係が強く, -1 に近いほど負の比例関係が強いことになる
- なお, (9) の分子の統計量を $n-1$ で割ったものは**共分散 (covariance)** と呼ばれる
- 相関および共分散はそれぞれ関数 `cor()` および関数 `cov()` で計算できる
  - 2 種類のデータ `x` および `y` が与えられたとき, それらの相関は `cor(x,y)` で計算できる
  - `x` がデータフレームのとき, `cor(x)` は (i, j) 成分が `x` の i 列と j 列の間の相関であるような行列 (**相関行列 (correlation matrix)**) を計算
  - 共分散についても同様
- 実行例 `cor.r`