matrix_util 仕様書

浅川伸一 <asakawa@twcu.ac.jp>

平成 14 年 12 月 3 日

目 次

1 データ形式

データは次の規則に従うテキスト形式である。

- # から行末まではコメントとして扱われる。
- データの区切りは space および tab である。
- 行数と列数がそのままデータ行列のサイズとして扱われる。

このルールに従っていれば、どのような aplication program で作成したものでも、この行列演算に使用できる。
例えば 3 行 4 列のデータであれば

```
0.051531 0.920450 0.082645 0.979234
0.517007 0.144957 0.131360 0.072932
0.078072 0.265250 0.235743 0.990869
```

のようになる。
なお、データの内部表現は `double` である。一行の最大値は 32768 である。
定義は `fileio_util.h` にある。

2 program の説明

すべての program で引数を -h または –help にして起動すると簡単なヘルプが表示される。出力は標準出力に対して行なわれるので、プログラムをいくつか続けてパイプ処理させることができる。たとえばデータを 2 乗して逆行列を求めるためには

```
% cat data | matsq | matinv
```

または

```
% matsq data | matinv
```

とすればよい。
以下のプログラムの説明では以下の表記を用いた。

- `[]` で囲まれた引数は省略可能なことを示す。引数がデータファイル名で省略された場合は標準入力から読み込まれる。
- `...` と表記されている場合は、任意個数の引数を許す。
- `[a|b]` と表記されている場合は、`a` または `b` のいずれか指定することを意味する。

2.1 bstat

データ行列の各列に対して基本統計量を出力する。

書式: `bstat [options] [datafile]`

オプション:

`-mean:` 平均を出力する (デフォルト)

`-sd:` 標準偏差を出力する

`-max:` 最大値を出力する

`-min:` 最小値を出力する

`-range:` レンジを出力する

`-skew:` 歪度を出力する

`-kurtosis:` 尖度を出力する

`-corr:` 相関行列を出力する

`-invcorr:` 相関行列の逆行列を出力する

`-cov:` 分散行列を出力する

`-invcov:` 分散行列の逆行列を出力する

`-ubcov:` 不遍分散行列を出力する

`-invubcov:` 不遍分散行列の逆行列を出力する

`-v :` 冗長な出力 — 平均、合計、標準偏差、分散、2 乗和、最大値、最小値、レンジ、歪度、尖度 — を出力する。

`-h :` ヘルプを表示する

2.2 gram_schmidt

グラム–シュミットの直交化を行なう。

書式: `gram_schmidt [datafile]`

2.3 matadd

2 つの行列の和を出力する。

書式: `matadd A [B]`

2.4 matarith

行列の算術演算をする。行列の全ての要素に対して $y = a\{x\}b$ というような変換をする。b については -plus や -minus の引数で指定し、a については -times や -div の引数で指定する。value は double として扱われる。

書式: matarith -plus|-minus|-times|-div <value> [datafile]

引数:

-plus value : 行列の各要素に対して value を加える。

-minus : 行列の各要素から value を引く。

-times : 行列の各要素に対して value を乗する。

-div : 行列の各要素に対して value を除する。

2.5 matcat

データ行列を整形して出力する。

書式: matcat [-OFMT format] [datafile] ...

引数:

OFMT : 出力形式 printf の書式と同じ。デフォルトは "%8.5f "

2.6 matconvol

合成積 $\int f(x)g(x-s)ds$ を出力する。

書式: matconvol KERNEL [DATA]

引数:

KERNEL : カーネル行列ファイル名

DATA : データ行列ファイル名

2.7 matcorr

各列ベクトルの相関行列を出力する。

書式: matcorr [datafile]

2.8 matgen

全要素が filler である行列を出力する。

書式: matgen N M filler

引数:

N : 行数 (int)

M : 列数 (int)

filler : 各要素の値 (double)

2.9 matI

単位行列を出力する。

書式: matI size

2.10 matinv

逆行列を出力する。

書式: matinv [datafile]

2.11 matnormalize

データ行列の各列を長さ 1 に規格化する。

書式: matnormalize [datafile]

2.12 matprd

2 つの行列の積を出力する。

書式: matprd A [B]

2.13 matresize

行列のサイズ (行数および列数) を変更する。

書式: matresize N M [datafile]

引数:

N : 新しい行数 (int)

M : 新しい列数 (int)

2.14 matsize

行列のサイズ (行数と列数) を出力する

書式: `matsize [datafile] ...`

2.15 matsq

行列の 2 乗、すなわち $\boldsymbol{X}'\boldsymbol{X}$ を出力する。

書式: `matsq [A]`

2.16 matsub

2 つの行列の差を出力する。

書式: `matsub A [B]`

2.17 matt

転置行列を出力する。

書式: `matt [datafile]`

2.18 mattrim

データ行列から内部の小領域を切り出す。すなわち、

```
(1,1)-----------------------------(1,m)
  |  (top,left)--------+            |
  |     |              |            |
  |     |              |            |
  |     +--------(bottom,right)     |
  |                                 |
(n,1)-----------------------------(n,m)
```

のように n 行 m 列のデータ行列の中から top 行 left 列 目のデータから bottom 行 righit 列目までの小領域の行列を取り出す。

書式: `mattrim sn sm en em [datafile]`

引数:

`sn` : 切り出す上端の行 (int)

sm : 切り出す左端の行 (int)

en : 切り出す下端の行 (int)

em : 切り出す右端の行 (int)

2.19 metricMDS

n 行 m 列のデータ行列から m 行 m 列の距離行列を作って古典的多次元尺度構成法 metric MDS を実行する。あるいは -d option のとき m 行 m 列の距離行列 (対角要素は当然すべて 0) を入力して metricMDS を実行する

書式: metricMDS [options] [datafile]

オプション:

-p [int]: 抽出する次元数 (デフォルトは 2)

-g : 結果のプロットを出力する

-d : 入力データを距離行列として計算を実行する。デフォルトでは生データから計算する。

2.20 pca

主成分分析を行なう。

書式: pca [options] [datafile]

オプション:

-p [int] : 抽出すべき固有値数 (絶対値の大きいものから順に)

-v : 固有値の計算に共分散行列を用いる。デフォルトでは相関行列を用いる

-gw : 主成分負荷量のプロットを出力する

-gs : 主成分得点のプロットを出力する

-notext : テキスト形式の出力を抑制する

2.21 plot2d

テキストベースで散布図を描く。

書式: plot2d [datafile]

2.22 reg

回帰分析を行なう。$\boldsymbol{Y} = \boldsymbol{AX}$ なる回帰式について $\boldsymbol{A}$ と $\boldsymbol{Y}$ とを与えて回帰係数行列 $\boldsymbol{X}$ を求める。

書式: `reg [options] -A datafile -y datafile`

引数:

-A datafile: データ行列ファイル名

-y datafile: ターゲット行列ファイル名

オプション:

`-1` : 定数項を加えた $\boldsymbol{Y} = \boldsymbol{AX} + c\mathbf{1}$ 回帰分析を行なう。

`-v` : 冗長な出力をする

`-h` : ヘルプを表示する

3 ライブラリの仕様

ほとんどすべての行列演算が C の関数として実現されている。

3.1 行列の宣言

行列を宣言したい場合には

```
#include "matrix_util.h"
#include "fileio_util.h"
```

のヘッダを `include` して

```
matrix A;
```

のように宣言する。
宣言した行列に対してメモリを割り当て、初期化するためには `matinit()` 関数を用いる。ただし行列を返す関数を用いる場合には関数内部でメモリの割り当てを行なっているので `matinit()` を使う必要は無い。

3.2 行列へのアクセス

n 行 m 列からなる行列 $\boldsymbol{A}$ の i 行 j 列目の要素にアクセスするためには、`*(A+offs(n,m,i,j))` のようにする。`offs(n,m,i,j)` は関数ではなく次のようなマクロである。

```
#define offs(n,m,i,j) ((m)*((i)-1)+((j)-1))
```

したがって、マクロの副作用に注意しなければならない。

3.3 行列の内部表現

行列データはすべて double へのポインタとして実現した。ある行列を 2 次元の配列と表現したい場合と、1 次元のベクトルとして表現したい場合とがあって煩わしいので、すべて自前で処理することにした。具体的には matrix_util.h の中で

```
#ifndef scalar
  #define scalar double
#endif
typedef scalar *vector, *matrix;
```

のように宣言されている。

3.4 関数の命名規則

matXXXX() という名前の関数については以下の基本ルールに従っている。例えば行列の和を計算する関数 matadd() については

- matAdd() は新しい行列作成し、結果をその作成した行列に入れて行列を返す関数。

  ```
  matrix Ret, A, B;
  int an, am; /* 行列 A の行数、列数 */
  int bn, bm; /* 行列 B の行数、列数 */

  Ret = matAdd(A,an,am, B,bn,bm);
  ```

  のように Ret には予めメモリを確保しておく必要は無い。

- matadd() は $A + B$ の結果を最後の引数 Ret に結果を代入する関数。Ret は予めメモリを確保しておく必要がある。サンプルオペレーションは

  ```
  matrix Ret, A, B;
  int an, am; /* 行列 A の行数、列数 */
  int bn, bm; /* 行列 B の行数、列数 */

  Ret = matinit(an,am);
  matAdd(A,an,am, B,bn,bm, Ret);
  ```

- matADD() は $A+=B$ のように A に B を加えたものを新たな A にする関数である。

同様の命名規則に従う関数群に matsub(), matprd() などがある。

4 ライブラリ関数リファレンス

4.1 matinit()

2 つの引数で指定されるサイズである行列のメモリを確保し、そのポインタを返す関数。行列の要素はすべて '\0' で初期化されている。

書式: `matrix matinit(int n, int m)`

引数:

n : 行数

m : 列数

4.2 intmatinit()

matinit() の整数版。各要素が整数型の行列のメモリを確保し、そのポインタを返す。

書式: `int intmatinit(int n, int m)`

引数:

n : 行数

m : 列数

4.3 vectinit()

ベクトル用のメモリを確保しそのポインタを返す関数。内部で matinit(n,1) を使っている。

書式: `vector vectinit(int n);`

引数:

n : ベクトルの次数

4.4 matfill()

第一引数の matrix を全ての要素が filler である行列にする。

書式: `int matfill(matrix A, int n, int m, double filler);`

引数:

A : matrix

n : 行数

m : 列数

filler : 行列の全要素に埋め込まれる値

4.5 matFill()

全ての要素が filler である行列を返す。

書式: `matrix matFill(int n, int m, double filler);`

引数:

n : 行数

m : 列数

filler : 行列の全要素に埋め込まれる値

4.6 vectinnerprd()

2 つのベクトル内積を返す関数。

書式: `double vectinnerprd(vector a, vector b, int n);`

引数:

a : ベクトル

b : ベクトル

n : ベクトルの次数

4.7 vectouterprd()

2 つのベクトルの外積を返す行列に書き込む関数。第 3 引数で指定する行列はあらかじめ領域を確保しておく必要がある。

書式: `int vectouterprd(vector a, vector b, int n, matrix Out);`

引数:

a: ベクトル

b: ベクトル

out: 結果を格納する行列

4.8 vectOuterprd()

2 つのベクトルの外積を返す関数。この関数内部で結果を表す行列のメモリ領域し、そのポインタを返す。

書式: `matrix vectOuterprd(vector a, vector b, int n);`

引数:

a: ベクトル

b: ベクトル

4.9 vectnormal()

引数で与えられたベクトルを $|x| = 1$ に規格化する関数。引数で与えられたベクトルが書き変えられる。

書式: `vectnormal(vector x, int n);`

引数:

x: ベクトル

n: ベクトルの次元数

4.10 matI()

引数で与えられた次元数の単位行列を返す関数。

書式: `matrix matI(int n) ;`

引数:

n: 行列の大きさ

4.11 matdiag()

第１引数で指定された行列 (行列のサイズは第２引数で指定される) の対角要素のみからなる行列を第４引数で指定された行列に代入する関数。第１、第４引数の行列はあらかじめメモリを確保しておかなければならない。

書式: `int matdiag(matrix A, int n, matrix Ret);`

引数:

A: 対角要素を抽出すべき行列

n: 行列のサイズ

Ret: 結果を格納する行列

4.12 matDiag()

第１引数で指定された行列 (行列のサイズは第２引数で指定される) の対角要素のみからなる行列を返す関数。

書式: `vector matDiag(matrix A, int n);`

引数:

A: 対角要素を抽出すべき行列

n: 行列のサイズ

4.13 mattrace()

第一引数で指定された行列のトレース (対角要素の和) を第３引数で指定された変数に格納する関数。

書式: `int mattrace(matrix A, int n, double *trace);`

引数:

A: トレースを計算すべき行列

n: 行列の次数

trace: 結果を格納する変数

4.14 matTrace()

第１引数で指定された行列のトレースを返す関数。

書式: `scalar matTrace(matrix A, int n);`

引数:

A: トレースを計算すべき行列

n: 行列の次数

4.15 matadd()

第１引数と第４引数で指定された二つの行列の和を第７引数で指定された行列に代入する関数。第１、第４、第７引数で指定される行列はあらかじめメモリ領域を確保していなければならない。$Ret = A + B$

書式: `int matadd(matrix A, int an, int am, matrix B, int bn, int bm, matrix Ret)`

引数:

A: 一つめの行列

an: 一つめの行列の行数

am: 一つめの行列の列数

B: 二つめの行列

bn: 二つめの行列の行数

bm: 二つめの行列の列数

Ret: 結果を格納する行列

4.16 matADD()

第１引数で指定された行列に第４引数で指定された行列を足し合わせる関数。この関数が呼び出されることで第一引数で指定された行列の値が変わることに注意。$A+ = B$

書式: `int matADD(matrix A, int an, int am, matrix B, int bn, int bm)`

引数:

A: 足し合わされた結果を格納する行列

an: 行列の行数

am: 行列の列数

B: 足し合わせるべき行列

bn: 行列の行数

bm: 行列の列数

4.17 matAdd()

第１引数で指定された行列に第４引数で指定された行列を足し合わせた行列を返す関数。$A + B$

書式: `matrix matAdd(matrix A, int an, int am, matrix B, int bn, int bm)`

引数:

A: 一つめの行列

an: 一つめの行列の行数

am: 一つめの行列の列数

B: 二つめの行列

bn: 二つめの行列の行数

bm: 二つめの行列の列数

4.18 matsub()

第１引数で指定された行列から第４引数で指定された行列を引いた値を第７引数で指定された行列に格納する関数。$Ret = A - B$ である。

書式: `int matsub(matrix A, int an, int am, matrix B, int bn, int bm, matrix Ret)`

引数:

A: 値を引かれる行列

an: 値を引かれる行列の行数

am: 値を引かれる行列の列数

B: 値を引く行列

bn: 値を引く行列の行数

bm: 値を引く行列の列数

Ret: 結果を格納する行列

4.19 matSUB()

第１引数で指定された行列から第４引数で指定された行列を引いた値を第１引数で指定された行列に格納する関数。$A- = B$

書式: `int matSUB(matrix A, int an, int am, matrix B, int bn, int bm)`

引数:

A: 値を引かれる行列

an: 値を引かれる行列の行数

am: 値を引かれる行列の列数

B: 値を引く行列

bn: 値を引く行列の行数

bm: 値を引く行列の列数

4.20 matSub()

第１引数で指定された行列から第４引数で指定された行列を引いた値を返す関数。$A - B$

書式: `matrix matSub(matrix A, int an, int am, matrix B, int bn, int bm)`

引数:

A: 値を引かれる行列

an: 値を引かれる行列の行数

am: 値を引かれる行列の列数

B: 値を引く行列

bn: 値を引く行列の行数

bm: 値を引く行列の列数

4.21 matprd()

第１引数で指定された行列と第４引数で指定された行列の積を第７引数でしてされた行列に格納する関数。第１、第４、第７引数で指定する行列はあらかじめメモリが確保されていなければならない。

書式: `int matprd(matrix A, int an, int am , matrix B, int bn, int bm, matrix Ret)`

引数:

A: 行列 A

an: 行列 A の行数

am: 行列 A の列数

B: 行列 B

bn: 行列 B の行数

bm: 行列 B の列数

Ret: 結果を格納する行列

4.22 matPRD()

第１引数で指定された行列と第４引数で指定された行列の積を第１引数で指定された行列に代入する関数。第１引数で指定された行列の値が変わることに注意。$A* = B$

書式: `int matPRD(matrix A, int an, int am, matrix B, int bn, int bm)`

引数:

A: 行列 A

an: 行列 A の行数

am: 行列 A の列数

B: 行列 B

bn: 行列 B の行数

bm: 行列 B の列数

4.23 matPrd()

第１引数で指定された行列と第４引数で指定された行列の積を返す関数。

書式: `matrix matPrd(matrix A, int an, int am, matrix B, int bn, int bm)`

引数:

A: 行列 A

an: 行列 A の行数

am: 行列 A の列数

B: 行列 B

bn: 行列 B の行数

bm: 行列 B の列数

4.24 matcpy()

第１引数で指定された行列を第４引数で指定された行列にコピーする関数。第１、第４引数で指定する行列はあらかじめメモリを確保しておく必要がある。

書式: `int matcpy(matrix From, int an, int am, matrix To);`

引数:

From: コピーされるべき元行列

an: 元行列の行数

am: 元行列の列数

To: コピー先の行列

4.25 matCpy()

第１引数で指定された行列をコピーした行列を返す関数。

書式: `matrix matCpy(matrix A, int an, int am);`

引数:

From: コピーされるべき元行列

an: 元行列の行数

am: 元行列の列数

4.26 matarith()

第１引数で指定された行列の算術演算をした結果を第６引数で指定された行列に代入する関数。算術演算は $Ret = aX + b$ の形をとり a と b はそれぞれ第４、第５引数で指定する。

書式: `int matarith(matrix A, int n, int m, double a, double b, matrix Ret);`

引数:

A: 元行列

an: 元行列の行数

am: 元行列の列数

a: 行列の各要素を a 倍する

b: 行列の各要素に b を加える

Ret: 結果を格納する行列

4.27 matARITH()

第１引数で指定された行列を a 倍し b を加えた値を第一引数で指定された行列に代入する関数。第一引数で指定された行列の値が変わることに注意。

書式: `int matARITH(matrix A, int n, int m, double a, double b);`

引数:

A: 元行列

an: 元行列の行数

am: 元行列の列数

a: 行列の各要素を a 倍する

b: 行列の各要素に b を加える

4.28 matArith()

第１引数で指定された行列を a 倍し b を加えた値を返す関数。

書式: `matrix matMag(matrix A, int n, int m, double a, double b);`

引数:

A: 元行列

an: 元行列の行数

am: 元行列の列数

a: 行列の各要素を a 倍する

b: 行列の各要素に b を加える

4.29 matt()

第１引数で指定された行列の転置行列を第４引数に代入する関数。第１、第４引数で指定される行列はあらかじめメモリが確保されていなければならない。

書式: `int matt(matrix A, int an, int am, matrix Ret);`

引数:

A: 元行列

an: 元行列の行数

am: 元行列の列数

Ret: 結果を格納する行列

4.30 matT()

第１引数で指定された行列の転置行列を返す関数。

書式: `matrix matT(matrix A, int an, int am);`

引数:

A: 元行列

an: 元行列の行数

am: 元行列の列数

4.31 matsq()

第１引数で指定された行列の２乗 A^tA を表す行列を第４引数で指定された行列に代入する関数。第１、第４引数で指定する行列はあらかじめメモリを確保しておかなければならない。

書式: `int matsq(matrix A, int n, int m, matrix Ret);`

引数:

A: ２乗する元行列

n: 元行列の行数

m: 元行列の列数

Ret: 結果を格納する行列

4.32 matSQ()

第１引数で指定された行列の２乗 A^tA を新たに第１引数で指定された行列に代入する関数。$A = A^tA$

書式: `int matSQ(matrix A, int n, int m);`

引数:

A: ２乗する元行列

n: 元行列の行数

m: 元行列の列数

4.33 matSq()

第１引数で指定された行列の２乗 A^tA を返す関数

書式: `matrix matSq (matrix A, int n, int m);`

引数:

A: ２乗する元行列

n: 元行列の行数

m: 元行列の列数

4.34 matsqinv()

第１引数で指定された行列の２乗の逆行列を第４引数で指定された行列に代入する関数。第１、第４引数で指定する行列はあらかじめメモリ上に領域を確保されていなければならない。

書式: `int matsqinv(matrix A, int n, int m, matrix Ret);`

引数:

A: ２乗の逆行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

Ret: 結果を格納する行列

4.35 matSQINV()

第１引数で指定された行列の２乗の逆行列を新たに第１引数で指定された行列に代入する関数。

書式: `int matSQINV(matrix A, int n, int m);`

引数:

A: ２乗の逆行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

4.36 matSqinv()

第１引数で指定された行列の２乗の逆行列を返す関数。

書式: `matrix matSqinv(matrix A, int n, int m);`

引数:

A: ２乗の逆行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

4.37 matsqinvmatt()

第１引数で指定した行列 A に対して $A^tA^{-1}A^{-t}$ を計算して第４引数で指定された行列に代入する関数。

書式: `int matsqinvmatt(matrix A, int n, int m, matrix Ret);`

引数:

A: 元行列

n: 元行列の行数

m: 元行列の列数

Ret: $A^tA^{-1}A^{-t}$ の計算結果を格納する行列

4.38 matSQINVMATT()

第１引数で指定した行列 A に対して $A^tA^{-1}A^{-t}$ を計算し結果を第１引数で指定された行列に格納する関数。元行列の値が書き変わることに注意。$A = A^tA^{-1}A^{-t}$

書式: `int matSQINVMATT(matrix A, int n, int m);`

引数:

A: 元行列

n: 元行列の行数

m: 元行列の列数

4.39 matSqinvmatt()

第1引数で指定した行列 A に対して $A^tA^{-1}A^{-t}$ を計算した行列を返す関数。$Ret = A^tA^{-1}A^{-t}$

書式: `matrix matSqinvmatt(matrix A, int n, int m);`

引数:

A: 元行列

n: 元行列の行数

m: 元行列の列数

4.40 gram_schmidt()

第一引数で指定した行列に対して Gram Schmidt の直交化をして第4引数で指定された行列に代入する関数。

書式: `int gram_schmidt (matrix A, int n, int m, matrix Ret);`

引数:

A: 元行列

n: 元行列の行数

m: 元行列の列数

Ret: 結果を格納する行列

4.41 Gram_Schmidt()

第一引数で指定した行列に対して Gram Schmidt の直交化をした行列を返す関数。

書式: `matrix Gram_Schmidt (matrix A, int n, int m);`

引数:

A: 元行列

n: 元行列の行数

m: 元行列の列数

4.42 matNormalize()

第一引数で指定した行列の各列に対して長さが 1 に規格化するした行列を返す関数。

書式: `matrix matNormalize (matrix A, int n, int m);`

引数:

A: 元行列

n: 元行列の行数

m: 元行列の列数

4.43 matNORMALIZE()

第一引数で指定した行列の各列に対して長さが 1 に規格化する関数。元行列の値が書き変わることに注意。

書式: `int matNORMALIZE (matrix A, int n, int m);`

引数:

A: 元行列

n: 元行列の行数

m: 元行列の列数

4.44 matnormalize()

第一引数で指定した行列の各列に対して長さが 1 に規格化した行列を第４引数で指定した行列に代入する関数。

書式: `int matnormalize (matrix A, int n, int m, matrix Ret);`

引数:

A: 元行列

n: 元行列の行数

m: 元行列の列数

Ret: 結果を格納する行列

4.45 writemat()

第１引数で指定したファイルディスクリプタで指定されたファイルに、第２引数で指定した行列を書き込む関数。第６引数で各要素の書式が指定できる。例えば"8.3f" と指定すれば総桁数 8 桁、小数点以下 3 桁で出力される。その際、第５引数で指定された文字列をファイルディスクリプタで指定されたファイルの先頭に印字する。

書式: `writemat(FILE *fp, matrix w, int row, int col, unsigned char *mes, unsigned char *fmt)`

引数:

fp: 行列を書き出すファイル型変数へのポインタ

w: 結果を書き出す行列

row: 結果を書き出す行列の行数

col: 結果を書き出す行列の列数

mes: 行列の前に印字されるメッセージ

fmt: 結果を書き出す行列のフォーマット

4.46 fwritemat()

書式: `fwritematf(FILE *fp, matrix w, int row, int col, unsigned char *mes, unsigned char *fmt)` writmat に同じ。

4.47 freadmat()

第１引数で指定されたファイルディスクリプタから行列を読み込んで第２引数で指定された行列に読み込む関数。第２引数で指定する行列は、予めメモリを確保していなくてはならない。

書式: `int freadmat(FILE *fp, matrix A, int n, int m);`

引数:

fp: 行列を読み込むファイル型変数へのポインタ

A: 結果を格納する行列

n: 行列の行数

m: 行列の列数

4.48 matreadfile()

第１引数で指定されたファイルディスクリプタから行列を読み込んでその行列を返す関数。

書式: `matrix matreadfile(char *filename, int n, int m);`

引数:

filename: 行列を読み込むファイル型変数へのポインタ

n: 行列の行数

m: 行列の列数

4.49 matinfo_read()

引数で指定されたファイルディスクリプタから行列を読み込んで以下の構造体 matinfo に情報を格納する関数。

```
typedef struct MAT_info {
    int n;
    int m;
    matrix mat;
} *matinfo;
```

構造体 `matinfo` は行列の行数と列数および行列本体からなる構造体である。

書式: `matinfo matinfo_read(char *filename);`

引数:

filename: 行列を読み込むファイル型変数へのポインタ

4.50 mattrim()

データ行列から内部の小領域を切り出して第８引数で指定された行列に格納する関数。すなわち、

```
(1,1)----------------------------(1,m)
  |  (top,left)--------+           |
  |     |              |           |
  |     |              |           |
  |     +--------(bottom,right)    |
  |                                |
(n,1)----------------------------(n,m)
```

のような n 行 m 列のデータ行列の中から top 行 left 列 目のデータから bottom 行 righit 列目までの小領域の行列を取り出す。

書式: `int mattrim(matrix O, int n, int m, int top, int left, int bottom, int rigtht, matrix Ret);`

引数:

O: 元行列

n: 元行列の行数

m: 元行列の列数

top: 切り出す左上端の行数

left: 切り出す左上端の列数

bottom: 切り出す右下端の行数

right: 切り出す右下端の列数

Ret: 結果を格納する行列

4.51 matTrim()

データ行列から内部の小領域を切り出した行列を返す関数。

書式: `matrix matTrim(matrix O, int n, int m, int top, int left, int bottom, int right);`

引数:

O: 元行列

n: 元行列の行数

m: 元行列の列数

top: 切り出す左上端の行数

left: 切り出す左上端の列数

bottom: 切り出す右下端の行数

right: 切り出す右下端の列数

4.52 plot2d()

第 1 引数で指定されたファイルディスクリプタに、第 2 引数で指定された行列の散布図を描く関数。

書式: `int plot2d(FILE *fp, matrix x, int n, int m, char *header);`

引数:

fp: 結果を書き出すファイル型変数へのポインタ

x: 結果を書き出す行列

n: 行列の行数

m: 行列の列数

header: 結果を標示する際に用いられる文字列

4.53 matinv()

第 1 引数で指定された行列の逆行列を第 4 引数で指定された行列に書き込む関数。

書式: `int matinv(matrix A, int n, scalar *det, matrix ret);`

引数:

A: 逆行列を計算する元行列

n: 行列の次数

det: 元行列の行列式の値

ret: 計算された逆行列

4.54 matInv()

第 1 引数で指定された行列の逆行列を返す関数。

書式: `matrix matInv(matrix A, int n, scalar *det);`

引数:

A: 逆行列を計算する元行列

n: 行列の次数

det: 元行列の行列式の値

4.55 determinant()

第 1 引数で指定された行列の行列式を返す関数。

書式: `scalar determinant(matrix A, int m);`

引数:

A: 逆行列を計算する元行列

n: 行列の次数

4.56 bstat_min()

第 1 引数で指定された行列の各列の最小値で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_min (matrix A, int n, int m);`

引数:

A: 最小値を計算する元行列

n: 行列の行数

m: 行列の列数

4.57 bstat_max()

第 1 引数で指定された行列の各列の最大値で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_max (matrix A, int n, int m);`

引数:

A: 最大値を計算する元行列

n: 行列の行数

m: 行列の列数

4.58 bstat_range()

第 1 引数で指定された行列の各列の範囲 (最大値 - 最小値) で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_range (matrix A, int n, int m);`

引数:

A: 範囲を計算する元行列

n: 行列の行数

m: 行列の列数

4.59 bstat_square()

第 1 引数で指定された行列の各列の 2 乗和で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_square(matrix A, int n, int m);`

引数:

A: 2 乗和を計算する元行列

n: 行列の行数

m: 行列の列数

4.60 bstat_sum()

第 1 引数で指定された行列の各列の総和で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_sum (matrix A, int n, int m);`

引数:

A: 総和を計算する元行列

n: 行列の行数

m: 行列の列数

4.61 bstat_mean()

第 1 引数で指定された行列の各列の平均値で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_mean (matrix A, int n, int m);`

引数:

A: 平均値を計算する元行列

n: 行列の行数

m: 行列の列数

4.62 bstat_variance()

第 1 引数で指定された行列の各列の標本分散で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_variance (matrix A, int n, int m);`

引数:

A: 標本分散を計算する元行列

n: 行列の行数

m: 行列の列数

4.63 bstat_sd()

第 1 引数で指定された行列の各列の標本標準偏差で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_sd (matrix A, int n, int m);`

引数:

A: 標本標準偏差を計算する元行列

n: 行列の行数

m: 行列の列数

4.64 bstat_skew()

第 1 引数で指定された行列の各列の歪み度で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_skew (matrix A, int n, int m, vector mean, vector sd);`

引数:

A: 歪み度を計算する元行列

n: 行列の行数

m: 行列の列数

4.65 bstat_kurtosis()

第１引数で指定された行列の各列の尖度で構成された行列 (1 行 m 列) を返す関数。

書式: `matrix bstat_kurtosis (matrix A, int n, int m, vector mean, vector sd);`

引数:

A: 尖り度を計算する元行列

n: 行列の行数

m: 行列の列数

4.66 matcorr()

第１引数で指定された行列の標本相関行列を第４引数で指定された行列に書き込む関数。第４引数で指定する行列は予めメモリが確保されていなければならない。

書式: `int matcorr (matrix A, int n, int m, matrix Corr);`

引数:

A: 相関係数を計算する元行列

n: 元行列の行数

m: 元行列の列数

4.67 matCorr ()

第１引数で指定された行列の標本相関行列を返す関数。

書式: `matrix matCorr (matrix A, int n, int m);`

引数:

A: 相関係数を計算する元行列

n: 元行列の行数

m: 元行列の列数

4.68 matcorrinv()

第１引数で指定された行列の標本相関行列の逆行列を第５引数で指定された行列に書き込む関数。第５引数で指定する行列は予めメモリが確保されていなければならない。

書式: `int matcorrinv(matrix X, int n, int m, scalar *det, matrix corrinv);`

引数:

A: 相関係数の逆行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

det: 逆行列の行列式の値を格納する変数

corrinv: 計算された相関係数の逆行列を格納する行列

4.69 matCorrinv()

第１引数で指定された行列の標本相関行列の逆行列を返す関数。

書式: `matrix matCorrinv(matrix X, int n, int m, scalar *det);`

引数:

A: 相関係数の逆行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

det: 逆行列の行列式の値を格納する変数

4.70 matubcov()

第１引数で指定された行列の不偏分散行列を第４引数で指定された行列に書き込む関数。第４引数で指定する行列は予めメモリが確保されていなければならない。

書式: `int matubcov(matrix X, int n, int m, matrix Ret);`

引数:

X: 不偏分散を計算する元行列

n: 元行列の行数

m: 元行列の列数

4.71 matUbcov()

第１引数で指定された行列の不偏分散行列を返す関数。

書式: `matrix matUbcov(matrix X, int n, int m);`

引数:

X: 不偏分散を計算する元行列

n: 元行列の行数

m: 元行列の列数

4.72 matcov()

第１引数で指定された行列の標本分散共分散行列を第４引数で指定された行列に格納する関数。第４引数で指定する行列は予めメモリが確保されていなければならない。

書式: `int matcov (matrix X, int n, int m, matrix Cov);`

引数:

X: 分散共分散行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

Cov: 計算された分散共分散行列を格納する行列

4.73 matCov()

第１引数で指定された行列の標本分散共分散行列を返す関数。

書式: `matrix matCov (matrix X, int n, int m);`

引数:

X: 分散共分散行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

4.74 matubcovinv()

第１引数で指定された行列の不偏分散共分散行列を第５引数で指定された行列に格納する関数。第５引数で指定する行列は予めメモリが確保されていなければならない。

書式: `int matubcovinv (matrix X, int n, int m, scalar *det, matrix Covinv);`

引数:

X: 不偏分散共分散行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

det: 不偏分散共分散行列の行列式を格納する変数

Covinv: 計算された不偏分散共分散行列を格納する行列

4.75 matUbcovinv()

第１引数で指定された行列の不偏分散共分散行列を返す関数。

書式: `matrix matUbcovinv (matrix X, int n, int m, scalar *det);`

引数:

X: 不偏分散共分散行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

det: 不偏分散共分散行列の行列式を格納する変数

4.76 matcovinv()

第１引数で指定された行列の分散共分散行列の逆行列を第５引数で指定された行列に格納する関数。第５引数で指定する行列は予めメモリが確保されていなければならない。

書式: `int matcovinv(matrix X, int n, int m, scalar *det, matrix Covinv);`

引数:

X: 分散共分散行列の逆行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

det: 分散共分散行列の逆行列の行列式を格納する変数

Covinv: 分散共分散行列の逆行列を格納する変数

4.77 matCovinv()

第１引数で指定された行列の分散共分散行列の逆行列を返す関数。

書式: `matrix matCovinv(matrix X, int n, int m, scalar *det);`

引数:

X: 分散共分散行列の逆行列を計算する元行列

n: 元行列の行数

m: 元行列の列数

det: 分散共分散行列の逆行列の行列式を格納する変数

4.78 eigen_pm()

第１引数で指定された行列の固有値を累乗法により求める関数。第２引数には固有値が大きい順に格納される。第３引数には対応する固有ベクトルが格納される。第４引数は元行列の次数。第５引数は求める固有値の数。第２、第３引数のベクトル、行列は予めメモリを確保しておく必要はない。

書式: `int eigen_pm(matrix a0, vector evv, matrix vv, int n, int p);`

引数:

a0: 固有値を求める元行列。

evv: 固有値を格納するベクトル。

vv: 対応する固有ベクトルを格納するベクトル

n: 元行列の次数

p: 求める固有値の数。