MAT 305: Mathematical Computing

Linear algebra

John Perry

University of Southern Mississippi

Fall 2011

Outline

Outline

The `matrix()` command

`matrix(` *ring*, *#rows*, *#cols*, entries`)` where

- *ring* (optional) an algebraic ring (default: $\mathbb{Z}$)
- *#rows*, *#cols* (optional) number of rows and columns (default depends on *entries*; no *entries* $\Longrightarrow 0 \times 0$ matrix)
- *entries* (optional) is one of
  - a list of entries, from northwest corner to southeast (if *#rows*, *#cols* specified)
  - a list of lists of entries, corresponding to rows
  - none specified? all entries 0

Algebraic ring?!?

- Linear algebra: over *fields* ($\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}\ldots$)
  - closed addition, multiplication
  - commutative addition, multiplication
  - associative addition, multiplication
  - distributive multiplication over addition
  - identity, inverses for addition, multiplication

Algebraic ring?!?

- Linear algebra: over *fields* ($\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}\ldots$)
  - closed addition, multiplication
  - commutative addition, multiplication
  - associative addition, multiplication
  - distributive multiplication over addition
  - identity, inverses for addition, multiplication
- Commutative algebra: over *rings* ($\mathbb{Z}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, matrices of fixed dimension…)
  - commutative multiplication, inverses for multiplication ***not required***

Common rings

- Integers: `ZZ` $\mathbb{Z}$
- Rationals: `QQ` $\mathbb{Q}$
- Reals: `RR` $\mathbb{R}$
  (Sage *approximates* w/53 bits precision)
- Complex: `CC` $\mathbb{C}$
  (Sage *approximates* w/53 bits precision)

Advanced rings

- Algebraic reals: `AA` $\overline{\mathbb{Q}}$
  (algebraic closure of $\mathbb{Q}$)
- Finite fields: `GF`(n) $\mathbb{Z}_n$
  (n power of prime; if not first power, specify string as name for generator)
- Finite rings: `ZZ.quotient_ring(ZZ.ideal(`n`))` $\mathbb{Z}_p$
  (n must be an integer)
- Symbolic: `SR`
  (can use expressions with symbols as entries)

Example matrices

```
sage:  MZ = matrix(ZZ,3,3)
sage:  MZ
[0 0 0]
[0 0 0]
[0 0 0]
```

Example matrices

```
sage:  MZ = matrix(ZZ,3,3)
sage:  MZ
[0 0 0]
[0 0 0]
[0 0 0]
sage:  MR = matrix(RR,[[1,2,3],[3,2,1],[1,1,2]])
sage:  MR
[1.00000000000000 2.00000000000000 3.00000000000000]
[3.00000000000000 2.00000000000000 1.00000000000000]
[1.00000000000000 1.00000000000000 2.00000000000000]
```

Example matrices

```
sage:  MZ = matrix(ZZ,3,3)
sage:  MZ
[0 0 0]
[0 0 0]
[0 0 0]
sage:  MR = matrix(RR,[[1,2,3],[3,2,1],[1,1,2]])
sage:  MR
[1.00000000000000 2.00000000000000 3.00000000000000]
[3.00000000000000 2.00000000000000 1.00000000000000]
[1.00000000000000 1.00000000000000 2.00000000000000]
sage:  MS = matrix(SR,[[x**2 + 1, 0, 0],
      [x + I, 1, 0]])
sage:  MS
[x^2 + 1        0        0]
[  x + I        1        0]
```

Help yourself read

Good idea to put rows in different lines

```
sage:  MR = matrix(RR, [
       [1,2,3],
       [3,2,1],
       [1,1,2]
     ])
sage:  MR
[1.00000000000000 2.00000000000000 3.00000000000000]
[3.00000000000000 2.00000000000000 1.00000000000000]
[1.00000000000000 1.00000000000000 2.00000000000000]
```

Accessing matrix entries

Matrix a list of lists $\implies$ $\texttt{M[i,j]} = M_{i,j}$

Accessing matrix entries

Matrix a list of lists $\implies$ `M[i,j]` $= M_{i,j}$

Example

```
sage:  MZ[1,2]
0
sage:  MZ[1,1] = 1
sage:  MZ
[0 0 0]
[0 1 0]
[0 0 0]
sage:  MZ[2,2] == 1
False
```

(counting starts from 0)

Submatrices

- `M.submatrix(`i`,` j`,` m`,` n`)` gives
  - $m \times n$ submatrix of M
  - whose northwest corner is in row i, column j
- `M.augment(`A`)` gives $(M|A)$

Submatrices

- `M.submatrix(`i`,` j`,` m`,` n`)` gives
  - $m \times n$ submatrix of M
  - whose northwest corner is in row i, column j
- `M.augment(`A`)` gives $(M|A)$

Example

```
sage:  MZ.submatrix(1,1,2,2)
[ 1 0]
[ 0 0]
```

Basic matrix operations

"dot" command	mathematics
`M.det()`	determinant
`M.inverse()`	
`M.transpose()`	
`M.eigenvalues()`	
`M.eigenvectors_right()`	right eigenvectors*
`M.eigenvectors_left()`	left eigenvectors*
`M.echelon_form()`	echelon form of unchanged M
`M.echelonize()`	change M to echelon form
`M.ncols()`	number of columns
`M.nrows()`	number of rows

*"right eigenvectors" are usual "eigenvectors"

Triangularizing matrices

"dot" command	mathematics
`M.set_row_to_multiple_of_row(i,j,a)`	set row i to a times row j*
`M.add_multiple_of_row(i,j,a)`	add a times row j to row i*
`M.swap_rows(i,j)`	swap rows i, j
`M.swap_columns(i,j)`	swap columns i, j

*row i changes; row j remains the same

Example: find inverse of matrix

Sage has a `.inverse()` command, but suppose you want to see steps...?

Example: find inverse of matrix

Sage has a `.inverse()` command, but suppose you want to see steps...?

Algorithm from High School Algebra II!

algorithm Compute inverse
  inputs
    M, an invertible matrix over a field
  outputs
    M^{-1}
  do
    Let $n = \dim(M)$
    Let A be augmented matrix $(M \mid I_n)$
    Triangularize A
    return rightmost $n \times n$ submatrix of A

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Augment MZ *by* I_4

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Augment MZ *by* I_4

To create I_4*, can set diagonal entries of zero matrix to 1...*

```
sage: I4 = matrix(4,4)
sage: for i in range(4):
          I4[i,i] = 1
sage: I4
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Augment MZ *by* I_4

...or use `identity_matrix()` *command*

```
sage: I4 = identity_matrix(4)
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Augment MZ *by* I_4

...or use identity_matrix() *command*

```
sage:  I4 = identity_matrix(4)
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
sage:  A = MZ.augment(I4)
sage:  A
[1 2 3 4 1 0 0 0]
[0 2 2 3 0 1 0 0]
[8 3 1 2 0 0 1 0]
[0 1 2 3 0 0 0 1]
```

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

First column: eliminate non-zero in row 3

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

First column: eliminate non-zero in row 3

```
sage: A.add_multiple_of_row(2,0,-8)
sage: A
[   1   2   3   4   1   0   0   0]
[   0   2   2   3   0   1   0   0]
[   0 -13 -23 -30  -8   0   1   0]
[   0   1   2   3   0   0   0   1]
```

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

First column: eliminate non-zero in row 3

```
sage: A.add_multiple_of_row(2,0,-8)
sage: A
[   1   2   3   4   1   0   0   0]
[   0   2   2   3   0   1   0   0]
[   0 -13 -23 -30  -8   0   1   0]
[   0   1   2   3   0   0   0   1]
```

Second column: swap row w/pivot to row 2, eliminate other non-zeros

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Second column: swap row w/pivot to row 2, eliminate other non-zeros

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Second column: swap row w/pivot to row 2, eliminate other non-zeros

```
sage: A.swap_rows(1,3)
sage: A.add_multiple_of_row(0,1,-2)
sage: A.add_multiple_of_row(2,1,13)
sage: A.add_multiple_of_row(3,1,-2)
sage: A
[   1   0  -1  -2   1   0   0  -2]
[   0   1   2   3   0   0   0   1]
[   0   0   3   9  -8   0   1  13]
[   0   0  -2  -3   0   1   0  -2]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Second column: swap row w/pivot to row 2, eliminate other non-zeros

```
sage:  A.swap_rows(1,3)
sage:  A.add_multiple_of_row(0,1,-2)
sage:  A.add_multiple_of_row(2,1,13)
sage:  A.add_multiple_of_row(3,1,-2)
sage:  A
[   1   0  -1  -2   1   0   0  -2]
[   0   1   2   3   0   0   0   1]
[   0   0   3   9  -8   0   1  13]
[   0   0  -2  -3   0   1   0  -2]
```

Third column: need pivot multiply row 3 by 1/3

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

```
sage: A.set_row_to_multiple_of_row(2,2,1/3)
...
TypeError: Multiplying row by Rational Field
element cannot be done over Integer Ring, use
change_ring or with_row_set_to_multiple_of_row
instead.
```

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

```
sage: A.set_row_to_multiple_of_row(2,2,1/3)
...
TypeError: Multiplying row by Rational Field
element cannot be done over Integer Ring, use
change_ring or with_row_set_to_multiple_of_row
instead.
```

Uh-oh! No multiplicative inverses in default ring! ($\mathbb{Z}$)
Change to $\mathbb{Q}$ *and proceed.*

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Uh-oh! No multiplicative inverses in default ring! ($\mathbb{Z}$)
Change to $\mathbb{Q}$ *and proceed.*

```
sage: A = A.change_ring(QQ)
sage: A
[   1    0   -1   -2    1    0    0   -2]
[   0    1    2    3    0    0    0    1]
[   0    0    3    9   -8    0    1   13]
[   0    0   -2   -3    0    1    0   -2]
```

Looks the same, but it's not.
Return to regularly-scheduled programming.

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

```
sage: A.set_row_to_multiple_of_row(2,2,1/3)
sage: A
[     1     0    -1    -2     1     0     0    -2]
[     0     1     2     3     0     0     0     1]
[     0     0     1     3  -8/3     0   1/3  13/3]
[     0     0    -2    -3     0     1     0    -2]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: need pivot
multiply row 3 by 1/3

```
sage:  A.set_row_to_multiple_of_row(2,2,1/3)
sage:  A
[     1     0    -1    -2     1     0     0    -2]
[     0     1     2     3     0     0     0     1]
[     0     0     1     3  -8/3     0   1/3  13/3]
[     0     0    -2    -3     0     1     0    -2]
```

Third column: eliminate other non-zeros

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: eliminate other non-zeros

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: eliminate other non-zeros

```
sage:  A.add_multiple_of_row(0,2,1)
sage:  A.add_multiple_of_row(1,2,-2)
sage:  A.add_multiple_of_row(3,2,2)
sage:  A
[      1      0      0      1  -5/3      0   1/3   7/3]
[      0      1      0     -3   16/3      0  -2/3 -23/3]
[      0      0      1      3   -8/3      0   1/3  13/3]
[      0      0      0      3 -16/3      1   2/3  20/3]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Third column: eliminate other non-zeros

```
sage:  A.add_multiple_of_row(0,2,1)
sage:  A.add_multiple_of_row(1,2,-2)
sage:  A.add_multiple_of_row(3,2,2)
sage:  A
[      1      0      0      1   -5/3      0    1/3    7/3]
[      0      1      0     -3   16/3      0   -2/3 -23/3]
[      0      0      1      3   -8/3      0    1/3   13/3]
[      0      0      0      3  -16/3      1    2/3   20/3]
```

Fourth column: need pivot
multiply row 4 by 1/3

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: need pivot
multiply row 4 by 1/3

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: need pivot
multiply row 4 by 1/3

```
sage:  A.set_row_to_multiple_of_row(3,3,1/3)
sage:  A
[      1      0      0      1  -5/3      0   1/3   7/3]
[      0      1      0     -3  16/3      0  -2/3 -23/3]
[      0      0      1      3  -8/3      0   1/3  13/3]
[      0      0      0      1 -16/9    1/3   2/9  20/9]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: need pivot
multiply row 4 by 1/3

```
sage:  A.set_row_to_multiple_of_row(3,3,1/3)
sage:  A
[      1      0      0      1  -5/3      0   1/3   7/3]
[      0      1      0     -3   16/3      0  -2/3 -23/3]
[      0      0      1      3   -8/3      0   1/3  13/3]
[      0      0      0      1 -16/9    1/3   2/9  20/9]
```

Fourth column: eliminate other non-zeros

Try it!

```
sage: MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                   [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: eliminate other non-zeros

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: eliminate other non-zeros

```
sage:  A.add_multiple_of_row(0,3,-1)
sage:  A.add_multiple_of_row(1,3,3)
sage:  A.add_multiple_of_row(2,3,-3)
sage:  A
[      1      0      0      0    1/9   -1/3    1/9    1/9]
[      0      1      0      0   16/3      1      0     -1]
[      0      0      1      0   -8/3     -1   -1/3   -7/3]
[      0      0      0      1 -16/9    1/3    2/9   20/9]
```

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Fourth column: eliminate other non-zeros

```
sage:  A.add_multiple_of_row(0,3,-1)
sage:  A.add_multiple_of_row(1,3,3)
sage:  A.add_multiple_of_row(2,3,-3)
sage:  A
[      1      0      0      0    1/9   -1/3    1/9    1/9]
[      0      1      0      0   16/3      1      0     -1]
[      0      0      1      0   -8/3     -1   -1/3   -7/3]
[      0      0      0      1 -16/9    1/3    2/9   20/9]
```

Have inverse! extract, test

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Have inverse! extract, test

Try it!

```
sage:  MZ = matrix([[1, 2, 3, 4], [0, 2, 2, 3],
                    [8, 3, 1, 2], [0, 1, 2, 3]])
```

Have inverse! extract, test

```
sage:  Minv = A.submatrix(0,4,4,4)
sage:  Minv * M
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

Other tools

Need another computation w/M? Remember:

- `M.<tab>` states all tools for M
- `M.<command>?` states help for `command`
- `M.<command>??` lists source code for `command`

Outline

Vectors

`vector(`*ring*`,` *entries*`)` where

- *ring* is base ring of *entries* (a list)
- default ring: $\mathbb{Z}$

Vectors

`vector(`*ring*`,` *entries*`)` where

- *ring* is base ring of *entries* (a list)
- default ring: $\mathbb{Z}$

Example

```
sage:  v = vector([1, 3, -1, 2])
sage:  u = vector([0, 2, 2, 0])
sage:  u*v
4
sage:  u.norm()
2*sqrt(2)
```

Dot product!

Vector spaces

`VectorSpace(`*field*`,` *dimension*`)` where

- *field* is a field (rings won't cut it here)
- *dimension* is dimension of the space

Example

1. Is $(-7,2,5)$ or $(-1,2,8)$ in space generated by $\{(1,2,0),(0,1,2)\}$?

1. If so, how can we write it in terms of a "good" basis?

```
sage:  V = VectorSpace(QQ,3)
sage:  S = V.subspace([[1,2,0],[0,1,2]])
```

Example

1. Is $(-7,2,5)$ or $(-1,2,8)$ in space generated by $\{(1,2,0),(0,1,2)\}$?

1. If so, how can we write it in terms of a "good" basis?

```
sage:  V = VectorSpace(QQ,3)
sage:  S = V.subspace([[1,2,0],[0,1,2]])
sage:  u = vector([-7,2,5])
sage:  S.echelon_coordinates(u)
...
ArithmeticError:  vector is not in free module
```

Example

1. Is $(-7,2,5)$ or $(-1,2,8)$ in space generated by $\{(1,2,0),(0,1,2)\}$?

1. If so, how can we write it in terms of a "good" basis?

```
sage:  V = VectorSpace(QQ,3)
sage:  S = V.subspace([[1,2,0],[0,1,2]])
sage:  u = vector([-7,2,5])
sage:  S.echelon_coordinates(u)
...
ArithmeticError:  vector is not in free module
sage:  v = vector([-1,2,8])
sage:  S.echelon_coordinates(v)
[-1, 2]
```

...that is, $\mathbf{v} = -\mathbf{b}_1 + 2\mathbf{b}_2$ where $\{\mathbf{b}_1, \mathbf{b}_2\}$ is an echelon basis of S

Vector spaces from matrices

```
sage:  M = matrix([[1,3,2],[1,0,2],[0,1,0]])
sage:  V = M.right_kernel()
```

`right_kernel` is the kernel you were taught

Vector spaces from matrices

```
sage:  M = matrix([[1,3,2],[1,0,2],[0,1,0]])
sage:  V = M.right_kernel()
```

right_kernel is the kernel you were taught

```
sage:  V
Free module of degree 3 and rank 1 over Integer Ring
Echelon basis matrix:
[1 0 -1/2]
```

Outline

Summary

- Sage does matrices
  - over fields *and* rings
  - symbolic ring! explore!
  - can change base ring
- Sage does vector spaces