MATRIX

Algebra & Vectors

Vectors

Let’s create one first!

Go to the R console and type in:

c(“Dharu”, “iz”, “best”)

Questions:

- What do you see?
- Try and type in numbers and Booleans instead of strings, what happens?
- What do you think a vector is?

Vectors

Vectors are a central component of R, not just another data structure. A vector can contain either numbers, strings, or logical values but not a mixture.

The `c(...)` operator can construct a vector from simple elements:

```
> c(1,1,2,3,5,8,13,21)
[1] 1 1 2 3 5 8 13 21

> c(1*pi, 2*pi, 3*pi, 4*pi)
[1] 3.141593 6.283185 9.424778 12.566371

> c("Never", "say", "never.")
[1] "You" "are" "stupid."

> c(TRUE,TRUE,FALSE,TRUE)
[1] TRUE TRUE FALSE TRUE
```

How would we put a vectors inside a vector?

Try it out first!

Questions:

- What happens when you put two vectors in a vector?
- Why do you think this happens?

How would we put a vectors inside a vector?

If the arguments to c(...) are themselves vectors, it flattens them and combines them into one single vector:

```
> v1 <- c(1,2,3)
> v2 <- c(4,5,6)
> c(v1,v2)

[1] 1 2 3 4 5 6
```

Questions:

- What happens when you put two vectors in a vector?
- What happens when you put two different data types in the same vector?
- Why do you think this happens?
- How would you check what data type it is? [Hint: mode()]

c(...) is a generic operator.

Which means that it works with many datatypes and not just vectors.

However, it might not do exactly what you expect,

so check its behavior before applying it to other datatypes and objects.

That's great!
What's the point?

Questions:

- Why would you want a vector (aka lists of numbers)?
- What would the applications to finance, economics be with a list of numbers?
- What are the effects of having different data types?
- How are different data types handled? What are their uses?

We can perform operations on vectors!

What kind of operations? Statistics obviously.

Try it out first!

Questions:

- What operations would you want to perform on a vector?
- R is relatively natural language based, how do you think you perform these operations?

Use one of these functions as applies, assuming that x and y are vectors:

- mean(x)
- median(x)
- sd(x)
- var(x)
- cor(x, y)
- cov(x, y)

```
> x <- c(0,1,1,2,3,5,8,13,21,34)

> mean(x)
[1] 8.8

> median(x)
[1] 4

> sd(x)
[1] 11.03328

> var(x)
[1] 121.7333
```

The sd function calculates the sample standard deviation, and var calculates the sample variance.
The cor and cov functions can calculate the correlation and covariance, respectively, between two vectors:

```
> x <- c(0,1,1,2,3,5,8,13,21,34)
> y <- log(x+1)

> cor(x,y)
[1] 0.9068053

> cov(x,y)
[1] 11.49988
```

Size doesn’t matter

```
> print(dframe)

   small     medium    big
1  0.6739635 10.526448 99.83624
2  1.5524619 9.205156  100.70852
3  0.3250562 11.427756 99.73202
4  1.2143595 8.533180  98.53608
5  1.3107692 9.763317  100.74444
6  2.1739663 9.806662  98.58961
7  1.6187899 9.150245  100.46707
8  0.8872657 10.058465 99.88068
9  1.9170283 9.182330  100.46724
10 0.7767406 7.949692 100.49814

> mean(dframe)
   small     medium   big
1.245040 9.560325 99.946003
```

How would we compare two vectors?

Try it out first!

Questions:

- How do you compare regular numbers?
- R is relatively natural language based, how do you think you perform these operations?
- How would you interpret the result?
- How do you think R compares the vectors?
- What would you call the result?

How would we compare two vectors?

The comparison operators (== , != , < , > , <= , >=) can perform an element-by-element comparison of two vectors. They can also compare a vector's element against a scalar. The result is a vector of logical values in which each value is the result of one element-wise comparison.

Vector operations are one of R's great strengths. All the basic arithmetic operators can be applied to pairs of vectors. They operate in an element-wise manner; that is, the operator is applied to corresponding elements from both vectors:

```
> v <- c(11,12,13,14,15)
> w <- c(1,2,3,4,5)
> v + w
[1] 12 14 16 18 20
> v - w
[1] 10 10 10 10 10
> v * w
[1] 11 24 39 56 75
> v / w
[1] 11.000000 6.000000 4.333333 3.500000 3.000000
> w ^ v
[1] 1 4096 1594323 268435456 30517578125
```

Questions:

- How do you compare regular numbers?
- R is relatively natural language based, how do you think you perform these operations?
- How would you interpret the result?
- How do you think R compares the vectors?
- What would you call the result?

Key properties of vectors:

Vectors are homogeneous

All elements of a vector must have the same type or, in R terminology, the same mode.

Vectors can be indexed by position

So v[2] refers to the second element of v .

Vectors can be indexed by multiple positions, returning a subvector

So v[c(2,3)] is a subvector of v that consists of the second and third elements.

Vector elements can have names

Vectors have a names property, the same length as the vector itself, that gives names to the elements:

```
> v <- c(10, 20, 30)
> names(v) <- c("Moe", "Larry", "Curly")
> print(v)
Moe Larry Curly
10    20     30
```

If vector elements have names then you can select them by name. Continuing the previous example:

```
> v["Larry"]
Larry
20
```

Matrices

- A Matrix is a 2-D list of numbers, written as a square or rectangular array
- Matrices are characterized by their dimensions always written as *rows* x *columns*

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \mathbf{X} = \begin{bmatrix} x_{00} & \cdots & x_{0j} \\ \vdots & \ddots & \vdots \\ x_{i0} & \cdots & x_{ij} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 5 & 3 & 0 \\ -3 & 2 & 3 \end{bmatrix}$$

- To create a matrix in R, use the **matrix(d,m,n)** command

> data<- c(1.1, 1.2, 2.1, 2.2, 3.1, 3.2)

> matrix(data, 2, 3)

[,1] [,2] [,3]

[1,] 1.1 2.1 3.1

[2,] 1.2 2.2 3.2

[3,] 2.2 3.1 3.2

- Matrix arithmetic is simple to do in R (Assume A,B, C are matrices of the same size)

> A + B

> 2*A + B – 14*C

- Matrix Transpose: denoted $\mathbf{M}^{\mathrm{T}}$, using the **t(…)** command in R swaps row with columns

> t(A)

> t(B-2*A+C)

- Matrix-vector product: vector has to have *same number of columns as the matrix*.

$$\begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix} \begin{bmatrix} y_{00} \\ y_{10} \end{bmatrix} = y_{00} \begin{bmatrix} x_{00} \\ x_{10} \end{bmatrix} + y_{10} \begin{bmatrix} x_{01} \\ x_{11} \end{bmatrix}$$

- To multiply a Matrix and a vector in R:

```
> v = c(1,2)
> A = matrix(c(1,2,3,4),2,2)
> ans = A*v
```

- It is convention to put the Matrix first, i.e. mathematicians get mad at v*A

- For the Matrix-matrix product AB, the B matrix has to have the same number of rows as the number of columns of A

$$\begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix} \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix} = \left[y_{00} \begin{bmatrix} x_{00} \\ x_{10} \end{bmatrix} + y_{10} \begin{bmatrix} x_{01} \\ x_{11} \end{bmatrix} y_{01} \begin{bmatrix} x_{00} \\ x_{10} \end{bmatrix} + y_{11} \begin{bmatrix} x_{01} \\ x_{11} \end{bmatrix} \right]$$

- In R, the Matrix-matrix product operator is **%*%**

> M = A%*%B

> M = A%*%B %*% C

- Note the Matrix-matrix product is *not commutative*, i.e. AB =/= BA!

- There is a special matrix called the identity matrix:

$$\mathbf{I} = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix}$$

It is a square matrix where all the elements along the main diagonal where m = n are 1 and all other elements are 0.

- Any square matrix multiplied by the Identity gives you what you started with. Can you think of an identity matrix for addition?
- In R, type **diag(n)** to get an identity matrix, where n is the size of the matrix

- The Matrix inverse: The matrix inverse is a square matrix **B** such that

$$\mathbf{AB} = \mathbf{I}$$

Therefore $\mathbf{B}$ can be written as $\mathbf{A^{-1}}$. Only square matrices may have an inverse, and only some square matrices are invertible.

- Without going into too much detail you can use the **solve(A)** command to find $\mathbf{A^{-1}}$. If the matrix cannot be inverted, R returns an error saying the system is “exactly singular”.

: Example: Solve the following Matrix equation for A

$$\mathbf{AB} + \mathbf{AC}^{\mathbf{T}} + \mathbf{D} = \mathbf{I}$$

- The solution is

$$\mathbf{A}\left(\mathbf{B} + \mathbf{C}^{\mathbf{T}}\right) = \mathbf{I} - \mathbf{D}$$

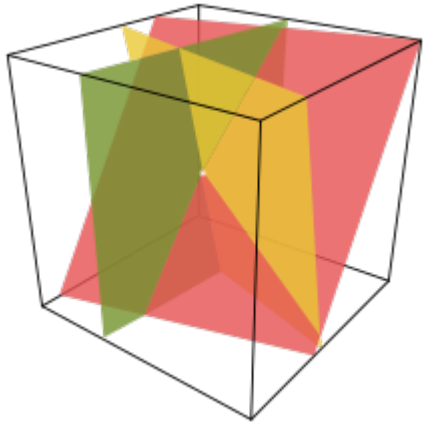
$$\mathbf{A} = (\mathbf{I} - \mathbf{D})\left(\mathbf{B} + \mathbf{C}^{\mathbf{T}}\right)^{-\mathbf{1}}$$

- A numeric example in R:

```
> B = matrix(c(0,2,3,4,5,6,7,8,9),3,3)
> C = matrix(c(1,2,3,1,2,3,1,2,3),3,3)
> D = matrix(c(1,1,1,2,2,2,3,3,3),3,3)
> A = (diag(3)-D)*solve(B+t(C))
```

- Some applications of (square) matrices:
- Can represent coefficients of systems of linear equations

$$x_1 = 3x_1 - x_2 - x_3$$
$$x_2 = x_1 + 2x_2 + x_3$$
$$x_3 = x_1 - x_2 - 4x_3$$



- Can represent linear transformations of Euclidean space
- Shear, mirror, squeeze, rotate, scale, projection
- Used extensively in statistics, Covariance matrix, PCA, least squares regression, etc.