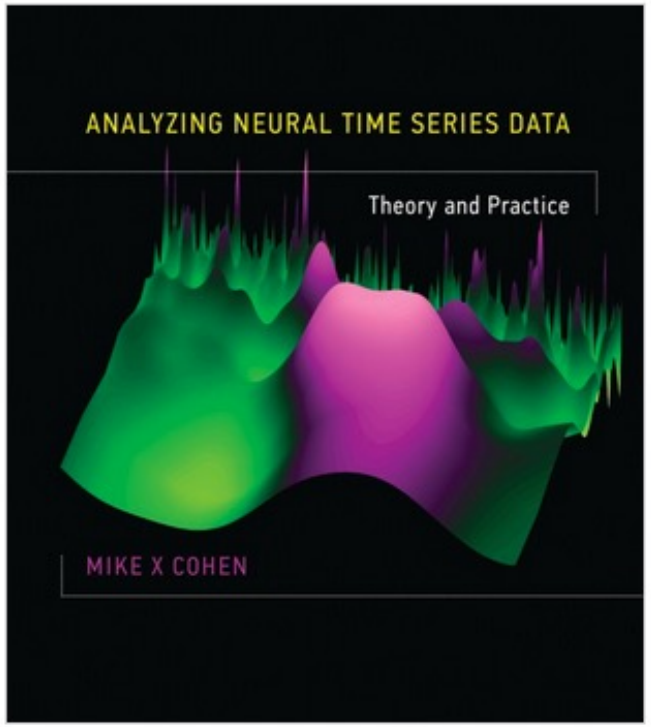
# PART II:

- Working with time series data
- Oscillations and frequency decomposition
- Functional connectivity
- Population coding: working with high dimensional data
- Data mining approaches and best practices




# Analyzing Neural Time Series Data

## Theory and Practice

By Mike X Cohen

A comprehensive guide to the conceptual, mathematical, and implementational aspects of analyzing electrical brain signals, including data from MEG, EEG, and LFP recordings.

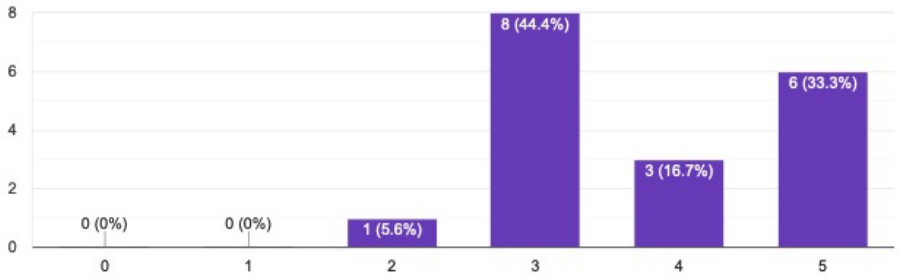
“In my experience teaching this material, most students look at an equation and slowly utter “okay”, as if they hope that by declaring the equation sensible, they will somehow understand what it means. But then they have a bewildered expression when it is time to turn to […insert coding platform of choice…] and implement the equation.”

# Why Matlab?

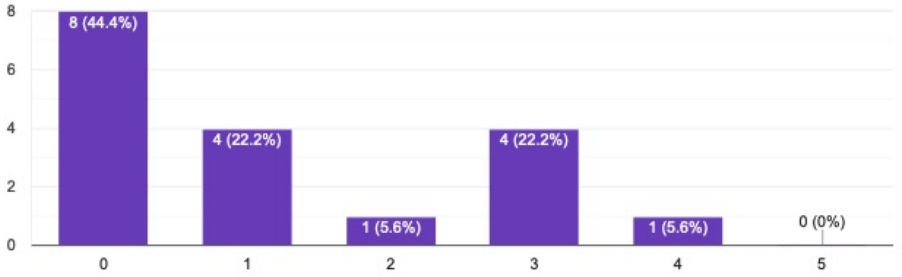
# Why Matlab?

Familiarity with:

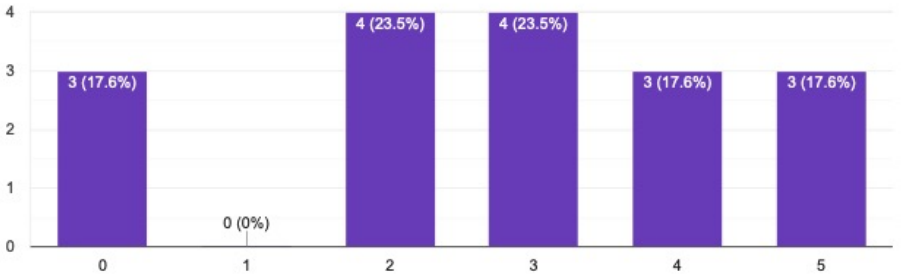
R




Matlab

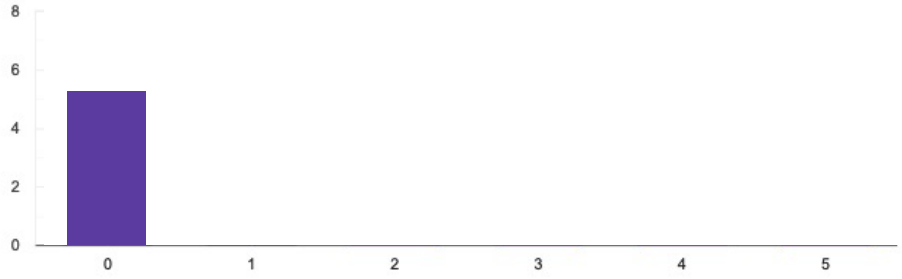



Python

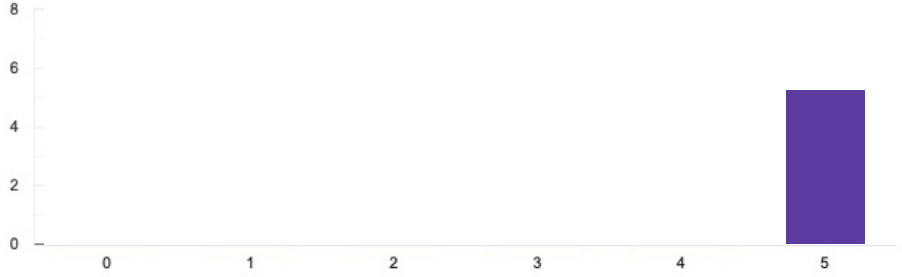



*Never tried it* → *Proficient, I use it regularly*

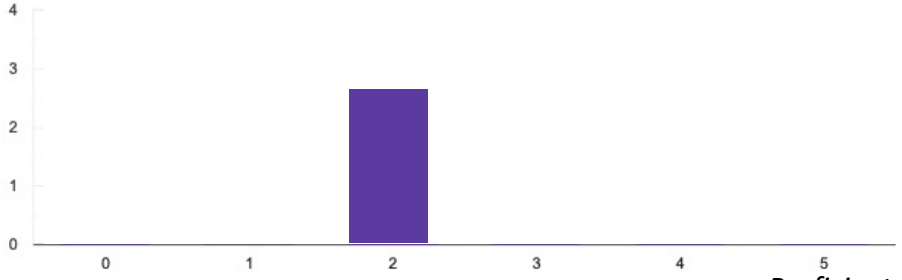
R




Matlab

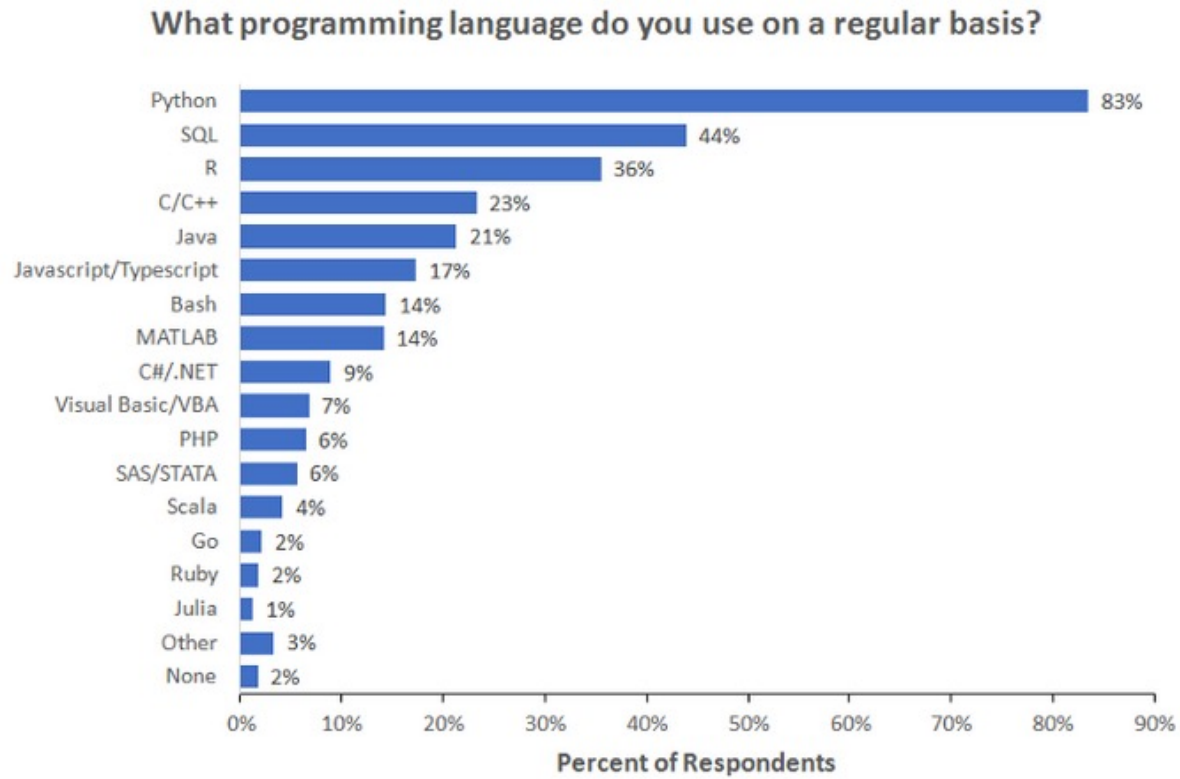



Python




*Never tried it* → *Proficient, I use it regularly*

# Programming Languages for Data Science




Note: Data are from the 2018 Kaggle Machine Learning and Data Science Survey. You can learn more about the study here: http://www.kaggle.com/kaggle/kaggle-survey-2018. A total of 18827 respondents answered the question.

BUSINESS BROADWAY
DATA SCIENCE | CUSTOMER ANALYTICS | MACHINE LEARNING

# Vectors

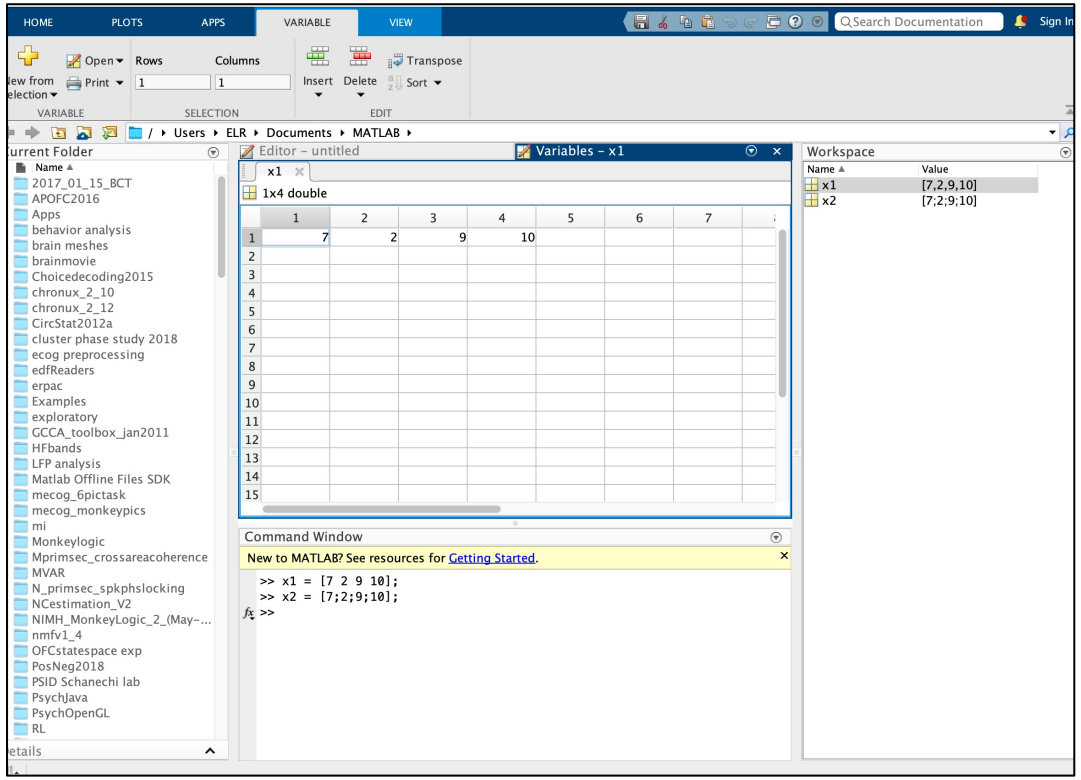
x1 = | 7 | 2 | 9 | 10 |

Shape: 1 x 4
size(x1) = (1,4)

x2 =

| 7 |
|---|
| 2 |
| 9 |
| 10 |

Shape: 4 x 1
Size(x2) = (4,1)



# Vectors

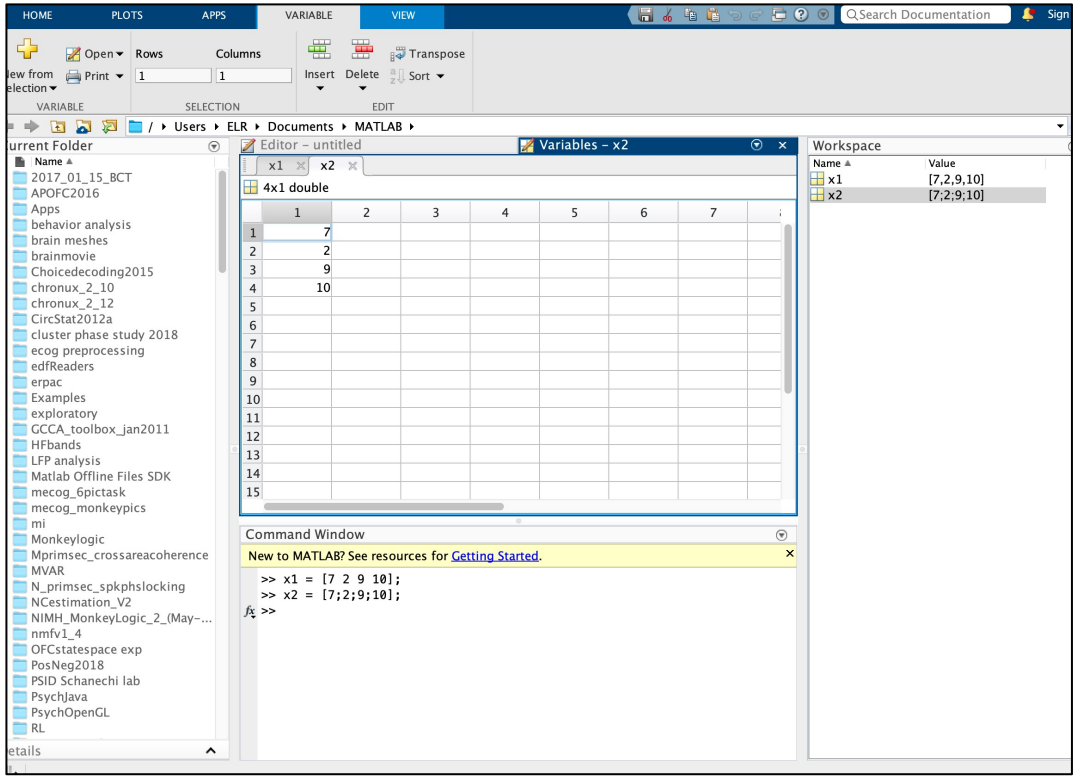
x1 = | 7 | 2 | 9 | 10 |

Shape: 1 x 4
size(x1) = (1,4)

x2 =

| 7 |
|---|
| 2 |
| 9 |
| 10 |

Shape: 4 x 1
Size(x2) = (4,1)



# Vectors

x1 = | 7 | 2 | 9 | 10 |

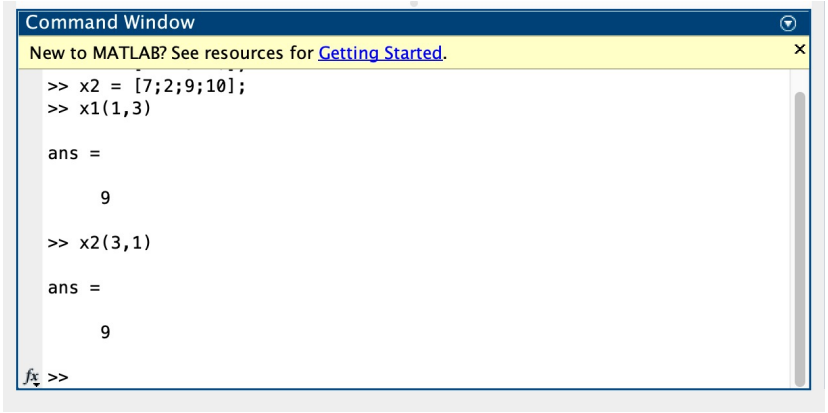
Shape: 1 x 4
size(x1) = (1,4)

Indexing:
x1(1,3) = 9
x2(3,1) = 9

x2 =

| 7 |
|---|
| 2 |
| 9 |
| 10 |

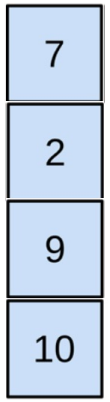
Shape: 4 x 1
Size(x2) = (4,1)



# Vectors

x1 = 


Shape: 1 x 4
size(x1) = (1,4)

x2 = 


Shape: 4 x 1
Size(x2) = (4,1)

Indexing:
x1(1,3) = 9
x2(3,1) = 9

**Note to python users**
Python starts indices at 0, not 1!

x1[1,3] does not exist!
x2[3,1] does not exist!

x1[0,2] = 9
x2[2,0] = 9

# Vectors

x1 = | 7 | 2 | 9 | 10 |

Shape: 1 x 4
size(x1) = (1,4)

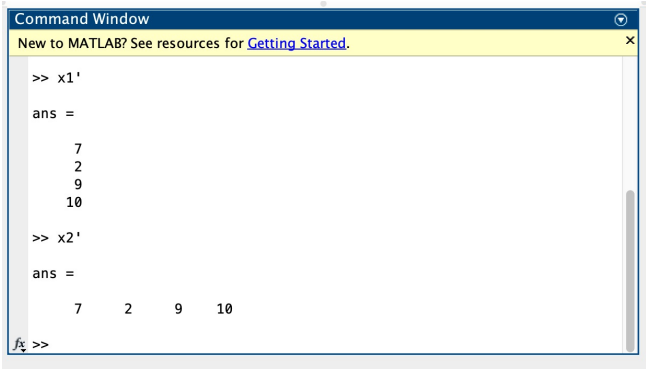
x2 =

| 7 |
|---|
| 2 |
| 9 |
| 10 |

Shape: 4 x 1
Size(x2) = (4,1)

Indexing:
x1(1,3) = 9
x2(3,1) = 9

Transposing:
x1 = x2’
x2 = x1’



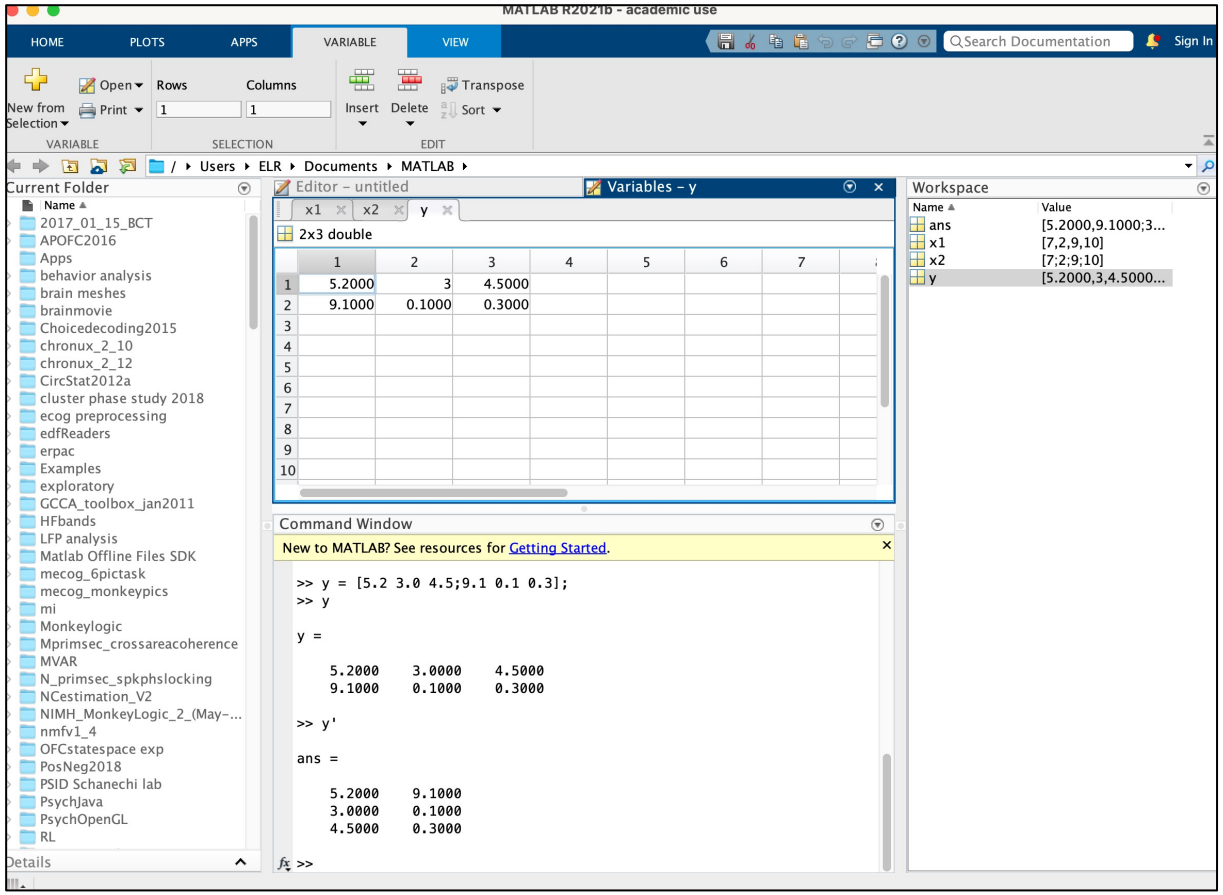
# 2D matrices

y =

| | | |
|---|---|---|
| 5.2 | 3.0 | 4.5 |
| 9.1 | 0.1 | 0.3 |

Shape: 2 x 3

size(y) = (2,3)



# 2D matrices

y =

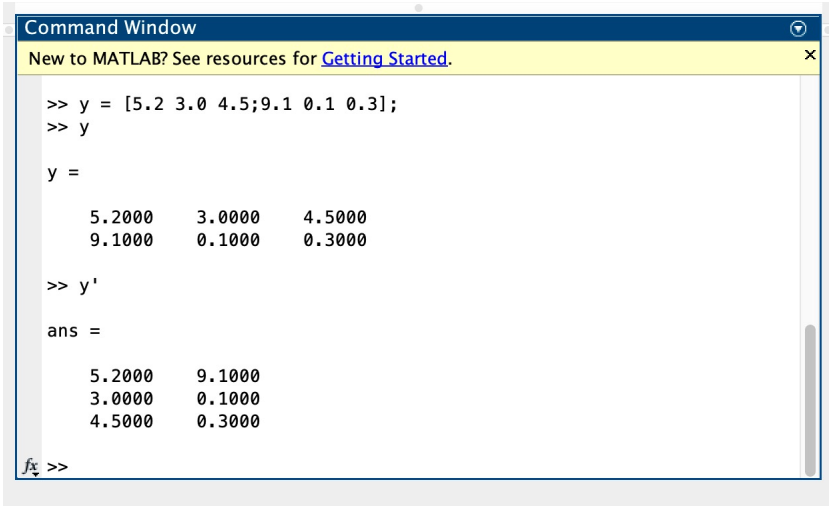
| | | |
|---|---|---|
| 5.2 | 3.0 | 4.5 |
| 9.1 | 0.1 | 0.3 |

Shape: 2 x 3
size(y) = (2,3)

Indexing:
y(1,2) = 3.0
y(2,3) = 0.3

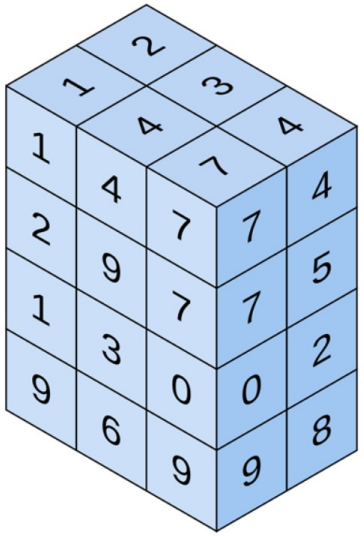
In python:
y(1,2) = 0.3
y(2,3) does not exist

Transposing:



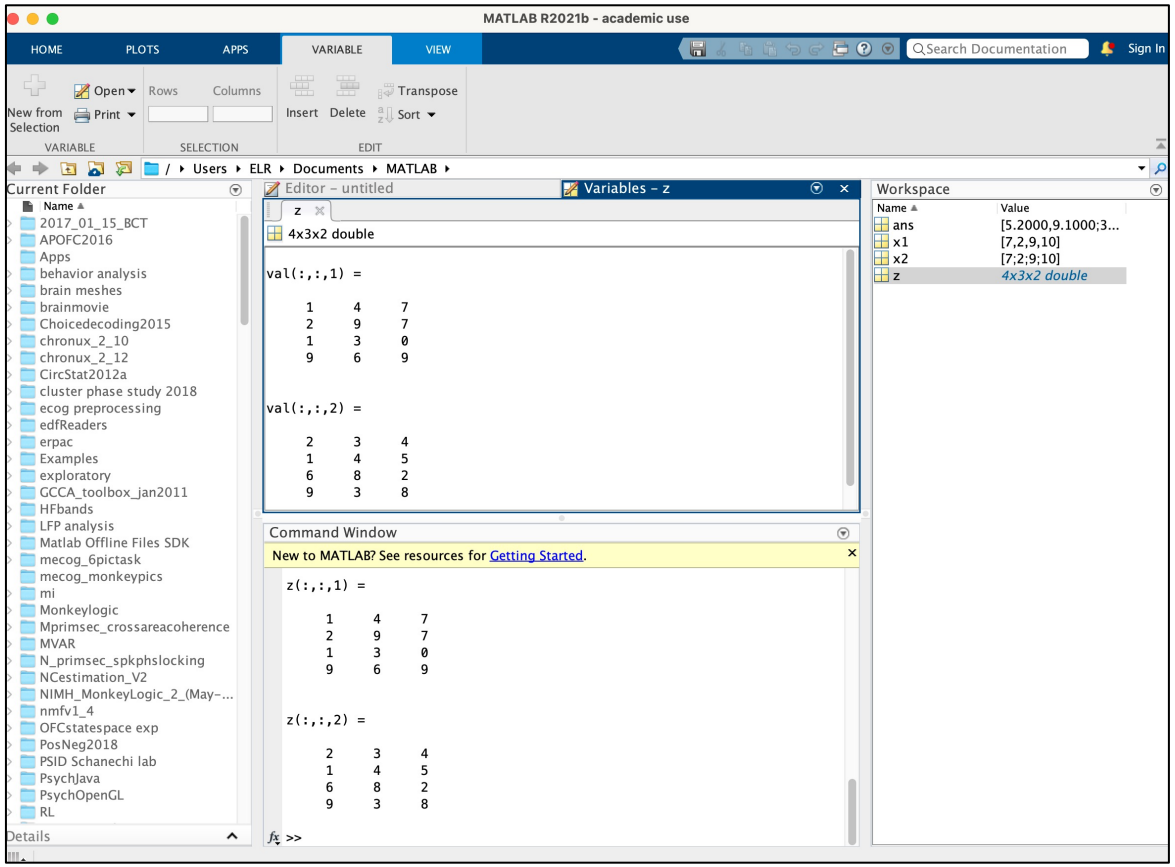
# 3D matrices

z =



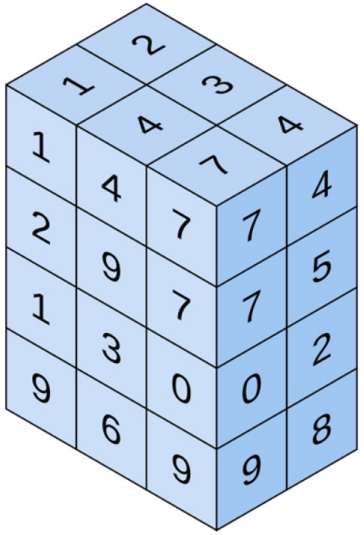

Shape: 4 x 3 x 2

size(z) = (4,3,2)



# 3D matrices

z =




Shape: 4 x 3 x 2
size(z) = (4,3,2)

Indexing:
z(2,2,1) = 9
z(1,3,2) = 4

Transposing: Not defined

# Matrix Algebra

*just a few things to keep in mind…*

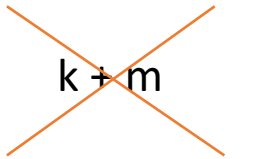
1. For element-by-element addition/subtraction your dimensions must match

k =

| |
|---|
| 4 |
| 12 |
| 1 |
| 23 |
| 9 |
| 18 |

m =

| |
|---|
| 34 |
| 8 |
| 1 |
| 13 |




# Matrix Algebra

*just a few things to keep in mind…*

1. For element-by-element addition/subtraction your dimensions must match

k =

| 4 |
|---|
| 12 |
| 1 |
| 23 |
| 9 |
| 18 |

m =

| 34 |
|---|
| 8 |
| 1 |
| 13 |

~~k + m~~

k + 3 ✓

k =

| 4 |
|---|
| 12 |
| 1 |
| 23 |
| 9 |
| 18 |

n =

| 34 | 8 | 1 | 13 | 10 | 2 |
|---|---|---|---|---|---|

~~k + n~~

k + n’ ✓

# Matrix Algebra

*just a few things to keep in mind…*

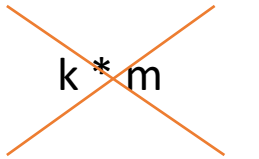
1. For element-by-element addition/subtraction your dimensions must match
2. Multiplication/division are assumed to be matrix operations

k =

| 4 |
|---|
| 12 |
| 1 |
| 23 |
| 9 |
| 18 |

m =

| 34 |
|---|
| 8 |
| 1 |
| 13 |




# Matrix Algebra

*just a few things to keep in mind…*

1. For element-by-element addition/subtraction your dimensions must match
2. Multiplication/division are assumed to be matrix operations

k =

| 4 |
|---|
| 12 |
| 1 |
| 23 |
| 9 |
| 18 |

m =

| 34 |
|---|
| 8 |
| 1 |
| 13 |

~~k * m~~

k * 3 ✓

k =

| 4 |
|---|
| 12 |
| 1 |
| 23 |
| 9 |
| 18 |

n =

| 34 | 8 | 1 | 13 | 10 | 2 |
|---|---|---|---|---|---|

k * n ⟶ matrix multiplication

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

# Matrix Algebra

*just a few things to keep in mind…*

1. For element-by-element addition/subtraction your dimensions must match
2. Multiplication/division are assumed to be matrix operations
   -> use .* or ./ for element-by-element operations

k =

| 4 |
|---|
| 12 |
| 1 |
| 23 |
| 9 |
| 18 |

m =

| 34 |
|---|
| 8 |
| 1 |
| 13 |

~~k * m~~

k * 3 ✓

~~k .* m~~

k =

| 4 |
|---|
| 12 |
| 1 |
| 23 |
| 9 |
| 18 |

n =

| 34 | 8 | 1 | 13 | 10 | 2 |
|---|---|---|---|---|---|

k * n → matrix multiplication

~~k * n'~~

k .* n' ✓

# Matrix Algebra

*just a few things to keep in mind…*

1. For element-by-element addition/subtraction your dimensions must match
2. Multiplication/division are assumed to be matrix operations
   -> use .* or ./ for element-by-element operations

k =

| | |
|---|---|
| 4 | 7 |
| 12 | 3 |
| 1 | 10 |
| 23 | 11 |
| 9 | 0 |
| 18 | 2 |

m =

| |
|---|
| 34 |
| 8 |
| 1 |
| 13 |
| 10 |
| 2 |

n =

| | | | | | |
|---|---|---|---|---|---|
| 34 | 8 | 1 | 13 | 10 | 2 |
| 5 | 9 | 22 | 6 | 14 | 1 |

k - m ✓

~~k + n~~

~~k * m~~

k * n → matrix multiplication

~~k * n'~~

k .* n' ✓

k ./ n' ✓

~~k .* m'~~

# “for” loops

k =

| | |
|---|---|
| 4 | 7 |
| 12 | 3 |
| 1 | 10 |
| 23 | 11 |
| 9 | 0 |
| 18 | 2 |

n =

| | | | | | |
|---|---|---|---|---|---|
| 34 | 8 | 1 | 13 | 10 | 2 |
| 5 | 9 | 22 | 6 | 14 | 1 |

for each row of k: find the row mean, multiply by the corresponding column of the first row of n, then subtract the corresponding column of the second row of n

*make a new variable (e.g. ‘i’)*

*define how ‘i’ will change with each loop*

```
for i = 1 : length(k(:,1))
    Y(i,1) =(mean(k(i,:)) * n(1,i) ) - n(2,i))
end
```

*starting with ‘for’ indicates you’re opening a for loop*

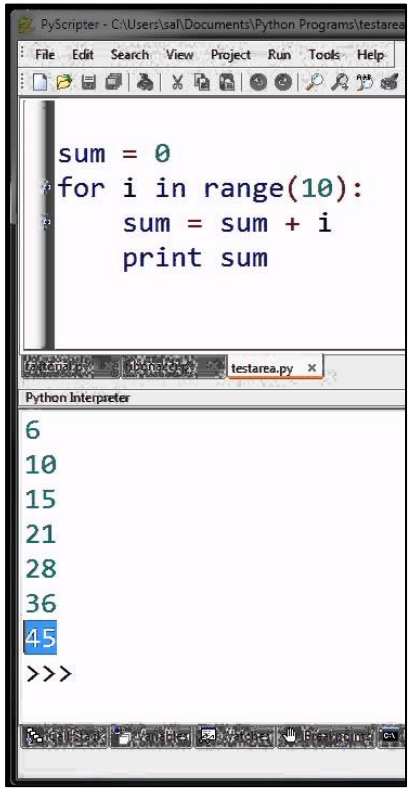
*Same answer with matrix operations*

```
Y = (mean(k,2).*n(1,:)’) - n(2,:)’
```

for loop in R

```
> for(i in 1:5) {                          # Basic for-loop
+   print(paste("This is step", i))
+ }
[1] "This is step 1"
[1] "This is step 2"
[1] "This is step 3"
[1] "This is step 4"
[1] "This is step 5"
>
```

for loop in python



# Self-guided exercises on matrix operations and for loops