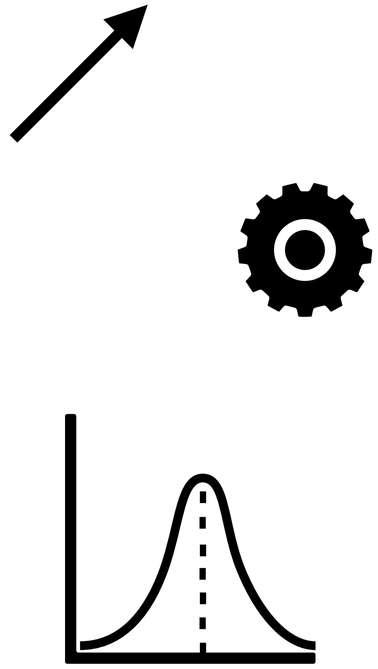
Pythonic Math



Michael Colaresi

What we have been doing

- How to “enhance” your computers imagination
  - Variables — nouns
  - Functions — verbs
  - Classes and methods — coherent phrases
  - programs — narratives/stories

Remember that all of this is “math” to the computer

- Back to binary
  - All of these processes
  - Everything in memory
  - Everything in storage
- A type of math that is defined as computer science
  - How computers can efficiently process, read, write, etc

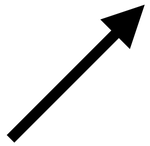
Not quite what we mean by math in empirical research

- Data
  - Vectors, matrices, graphs/networks
- Models
  - Functional forms (likelihood of the data given some unknowns), parameters, etc
  - Distributions of prior
- Inferences and quantities of interest
  - Distributions
  - Communication

But also not so different

- Data
  - Vectors, matrices, graphs/networks
- Models
  - Functional forms (likelihood of the data given some unknowns), parameters, etc
  - Distributions of prior
- Inferences and quantities of interest
  - Distributions
  - Communication

Storage and memory



But also not so different

- Data
  - Vectors, matrices, graphs/networks
- Models
  - Functional forms (likelihood of the data given some unknowns), parameters, etc
  - Distributions of prior
- Inferences and quantities of interest
  - Distributions
  - Communication

Storage and memory

Algorithms/functions/methods

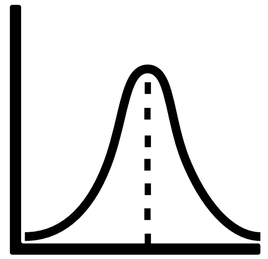
But also not so different

- Data
  - Vectors, matrices, graphs/networks
- Models
  - Functional forms (likelihood of the data given some unknowns), parameters, etc
  - Distributions of prior
- Inferences and quantities of interest
  - Distributions
  - Communication

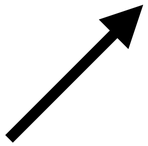
Storage and memory

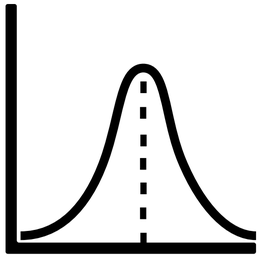
Algorithms/functions/methods

More algorithms/functions/methods



Numerical Computing

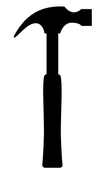




- Use cases:
  - Finding minima and maxima (and stopping rules)
  - Running simulations and calculating integrals
  - Visualizing densities and quantities of interest
  - But many others
    - Shortest path in a network
    - Edit distance between two strings

The *math and fractions* library

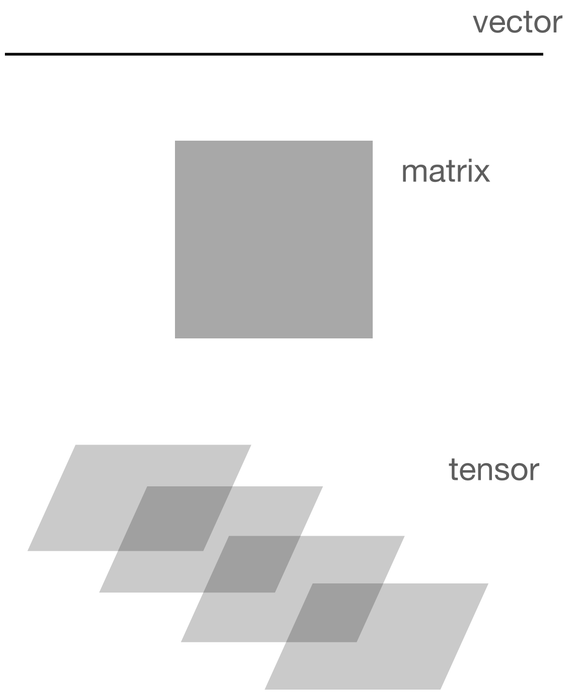
Builtin to Python



- import math
  - math.sin, math.cos, math.sqrt
- import fractions
  - fractions.Fraction(m, n) or fractions.Fraction(‘m/n’)
  - Better precision for rational numbers as ratios of ints

Numpy for matrices and arrays

- Import numpy as np
- np.array()
- np.array([[a, b], [c, d])
- np.array.shape
- indexing
  - x[0:2, 0:3]; rows, then columns

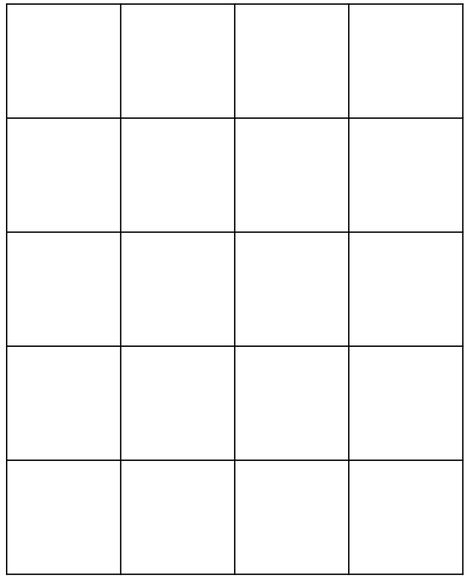


Not only storage, processing

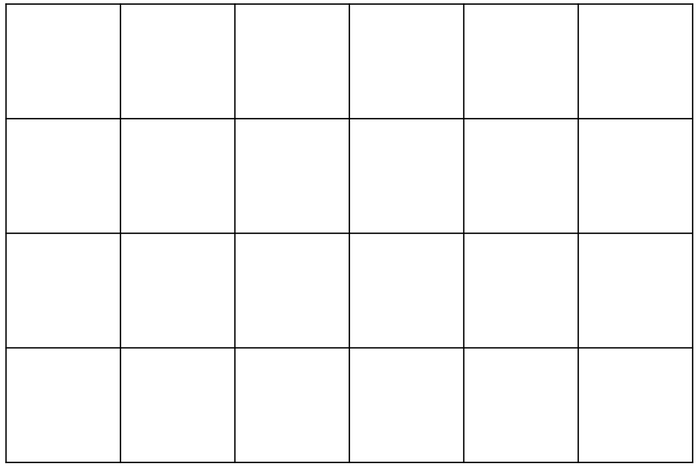
Linear Algebra

- Vectors
  - Points or arrows
- Linear transformations
  - Matrices
    - Act on the vectors that make up a space, transform them

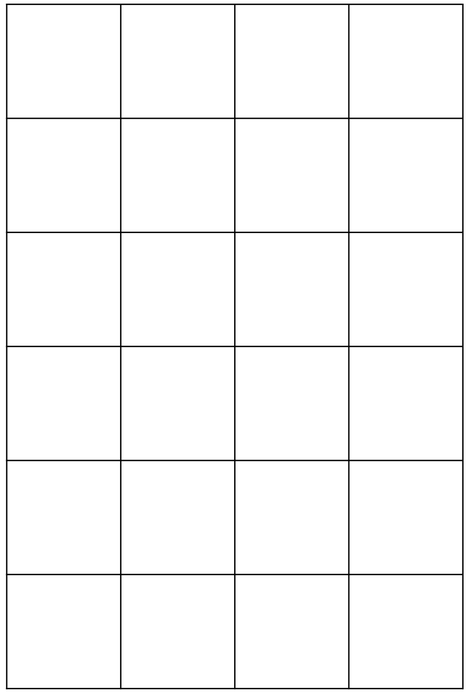
Matrices as transforms



M

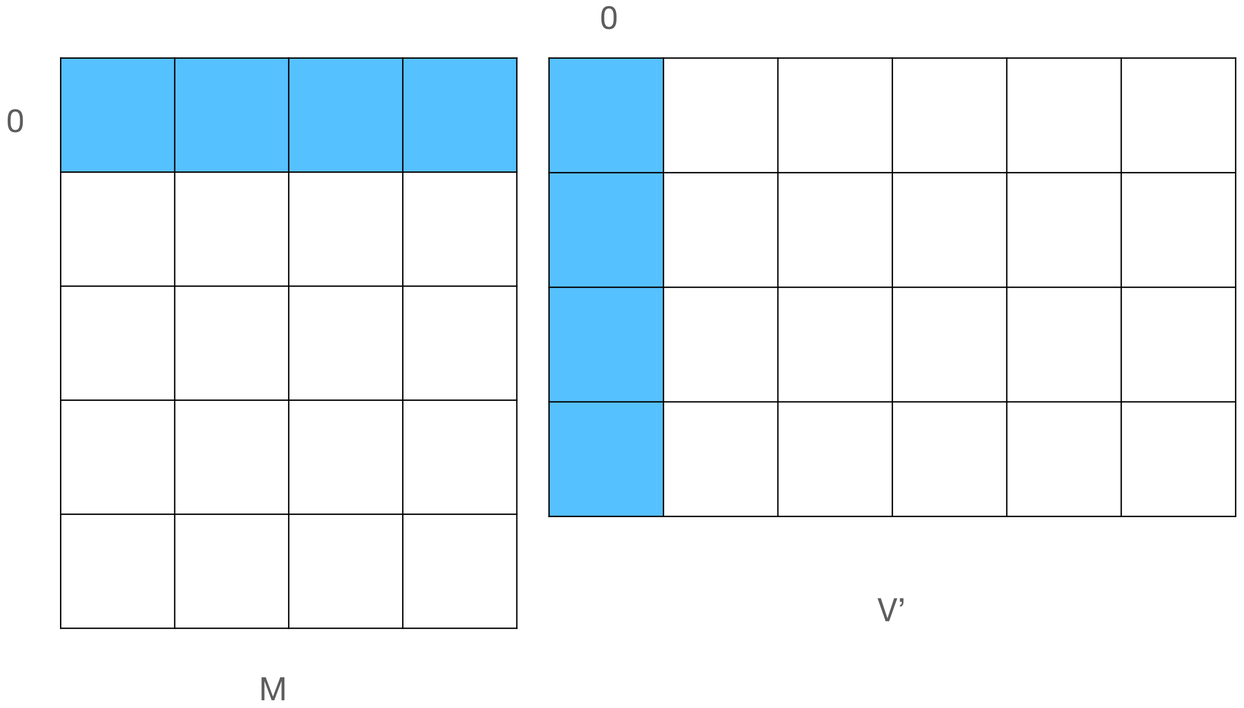


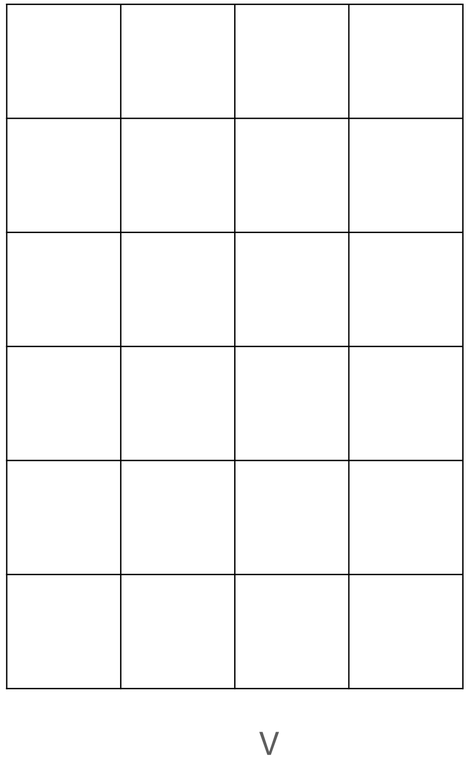
V’



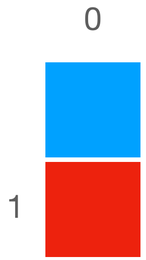
V

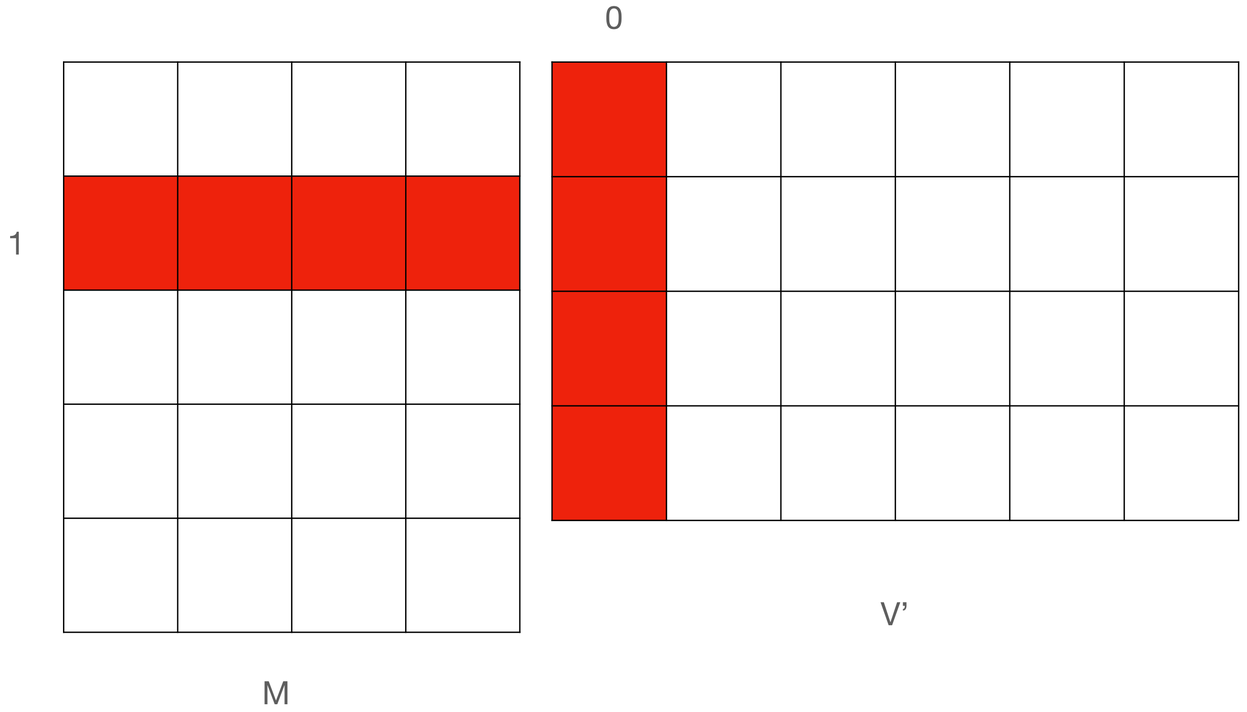
Matrices as transforms

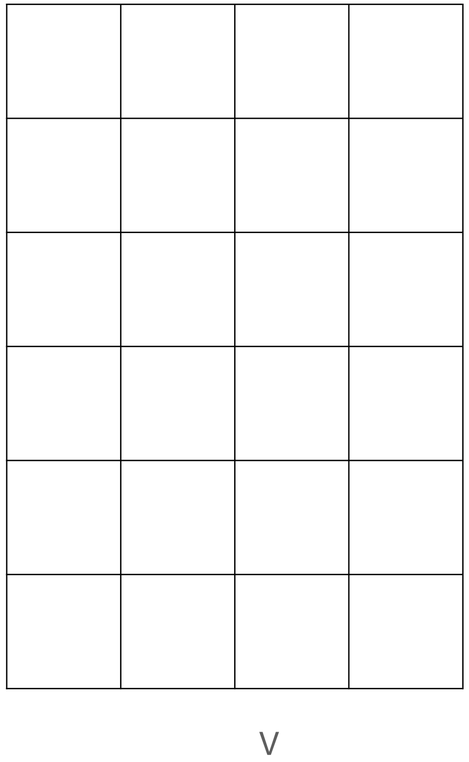




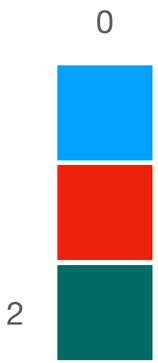
Matrices as transforms

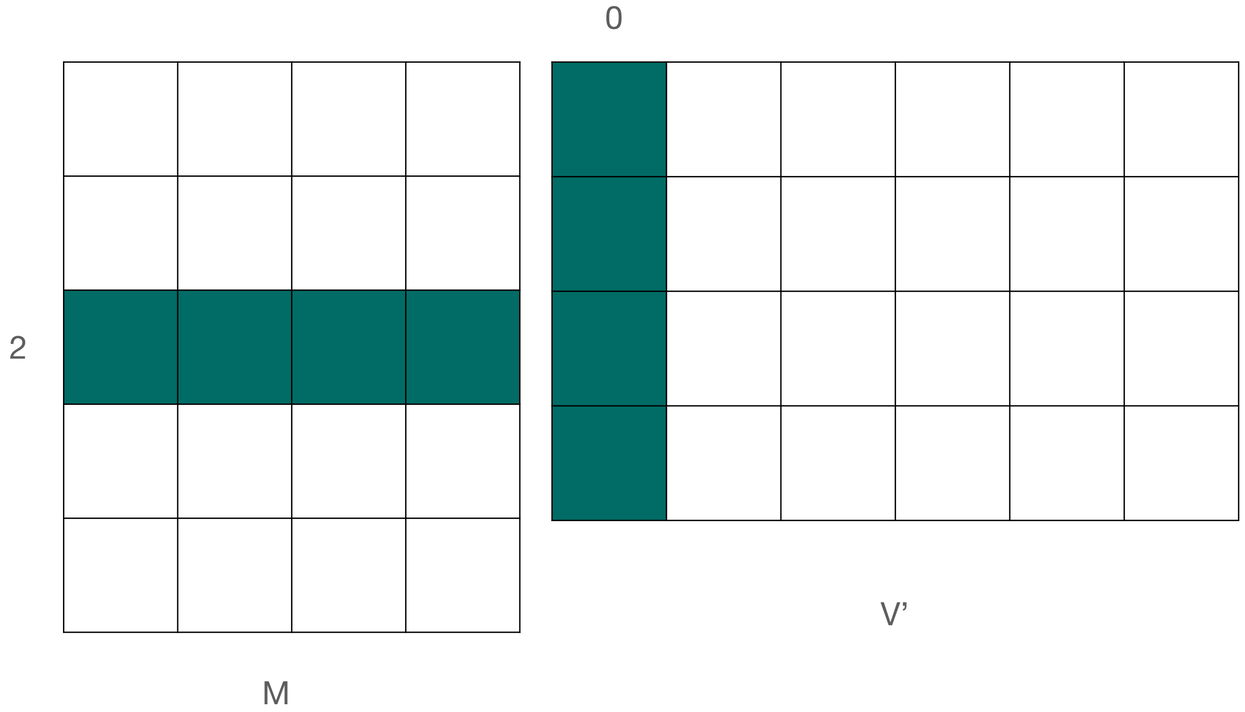


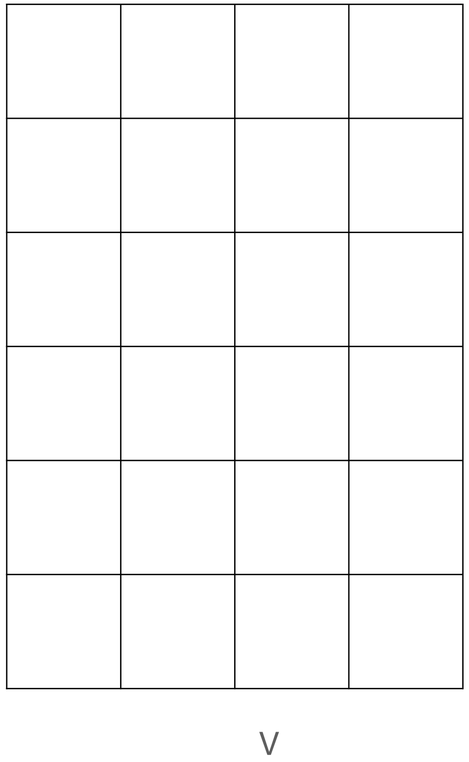




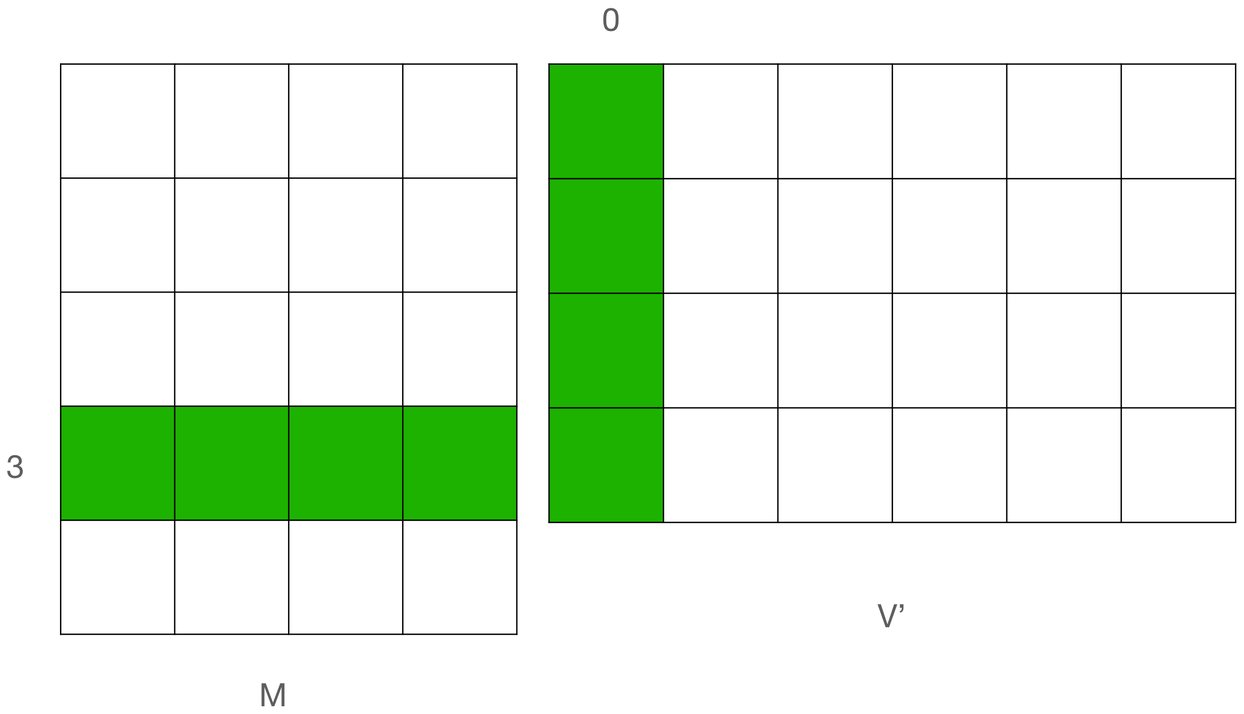
Matrices as transforms

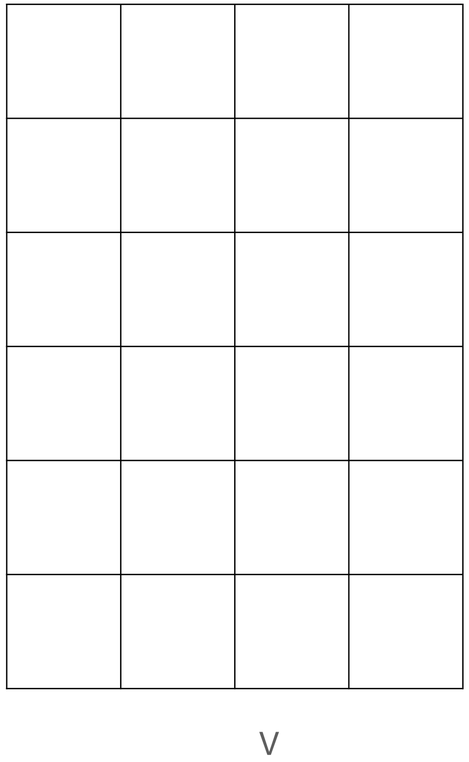




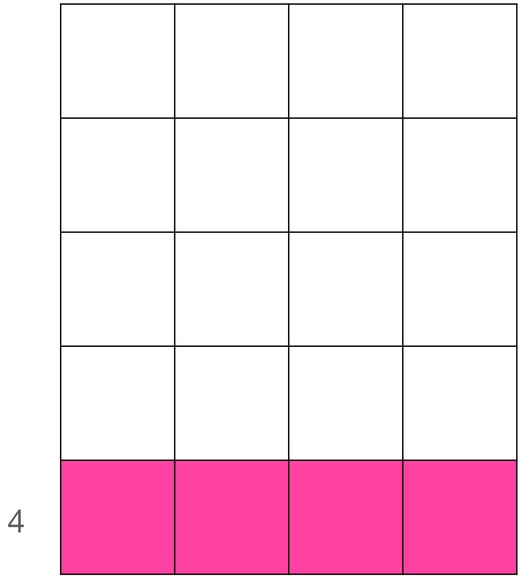


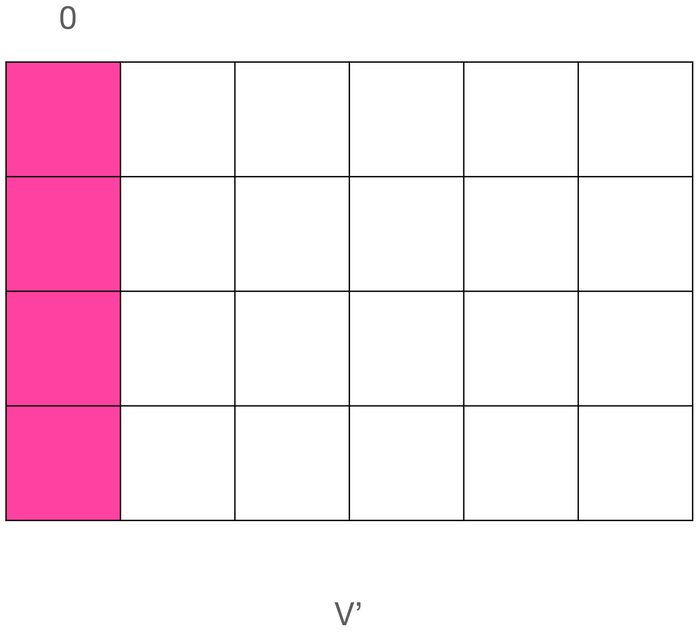
Matrices as transforms

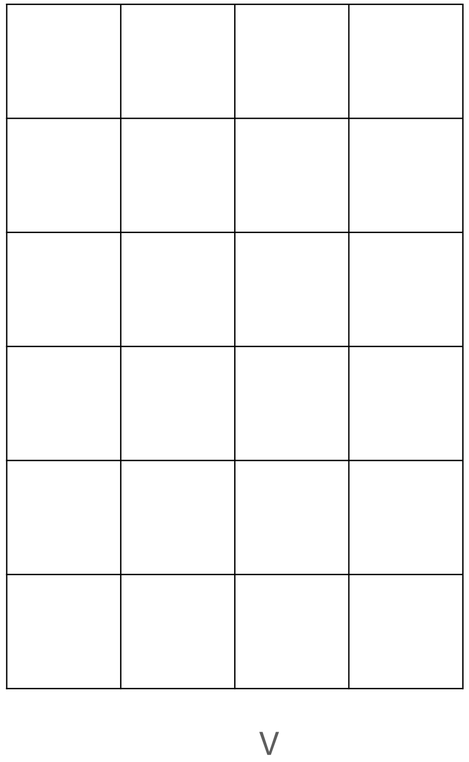




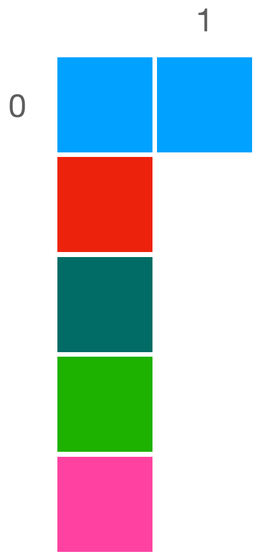
Matrices as transforms

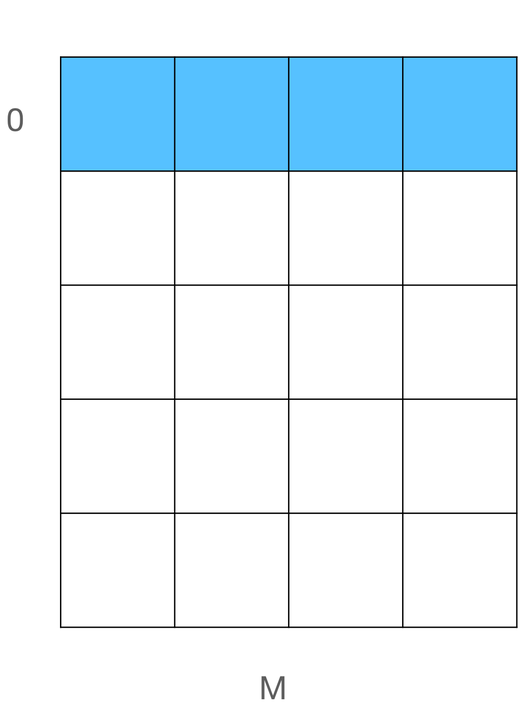


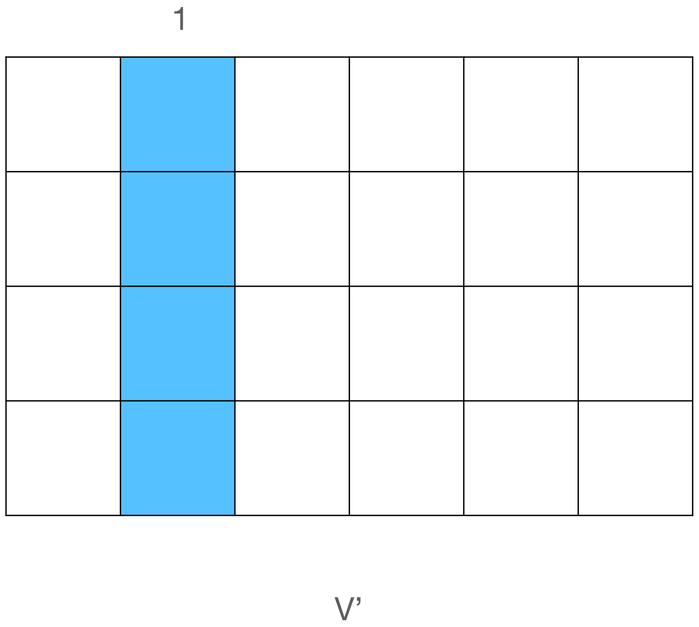


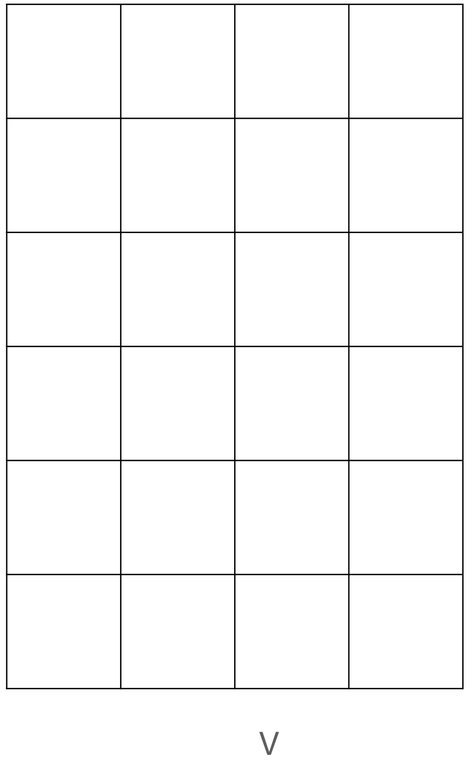


Matrices as transforms

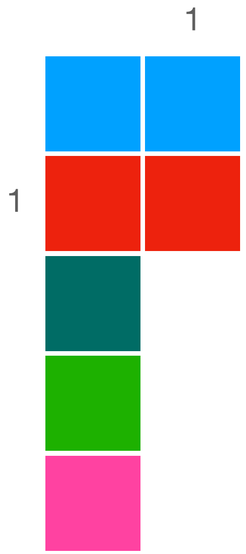


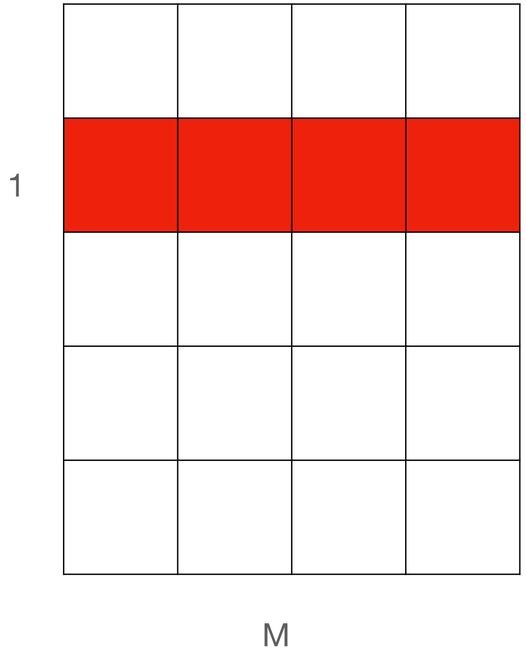


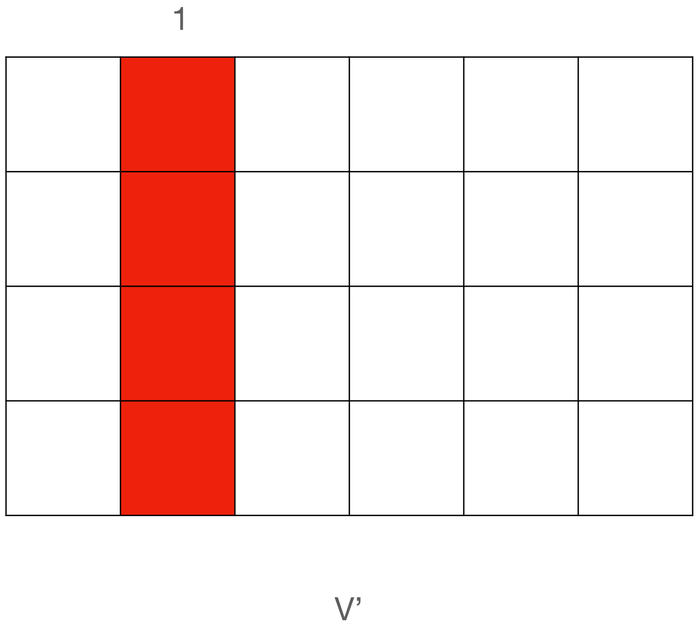


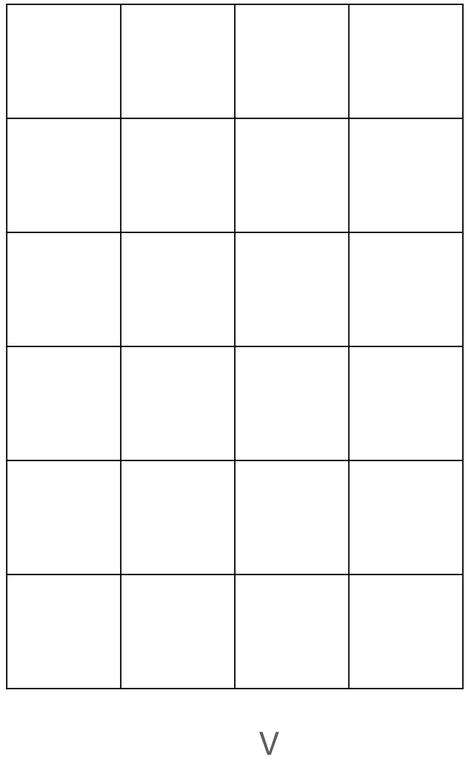


Matrices as transforms

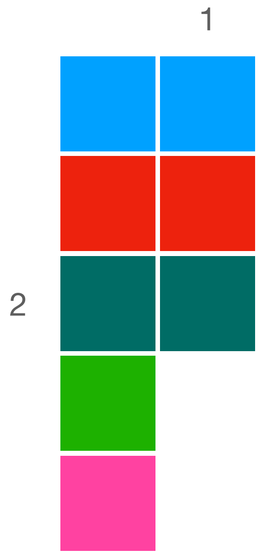


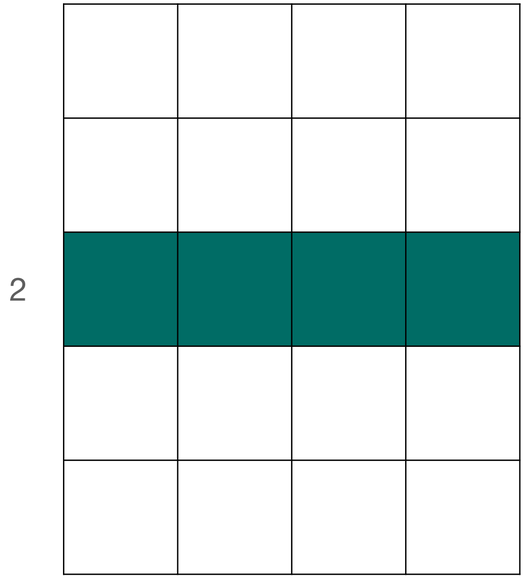




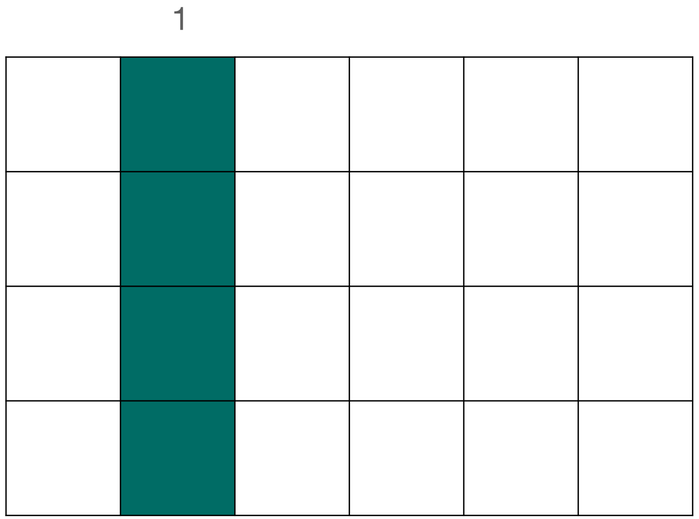


Matrices as transforms

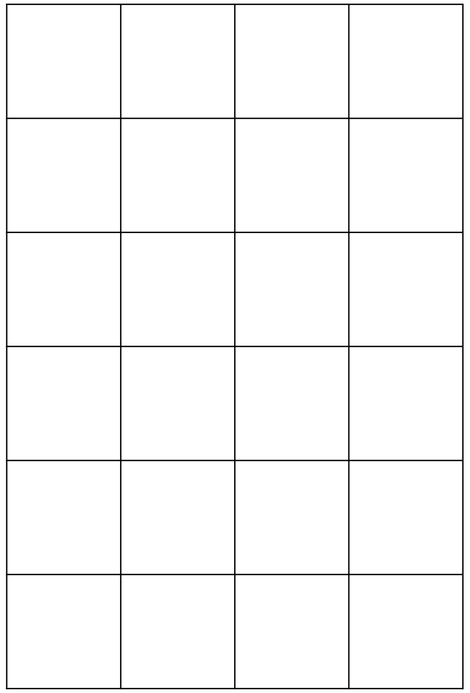




M

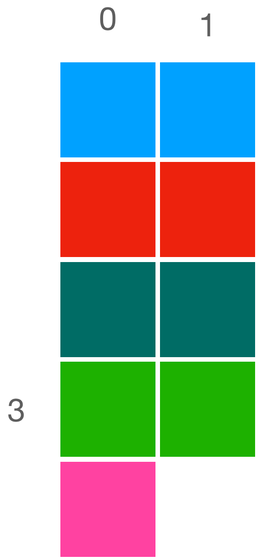


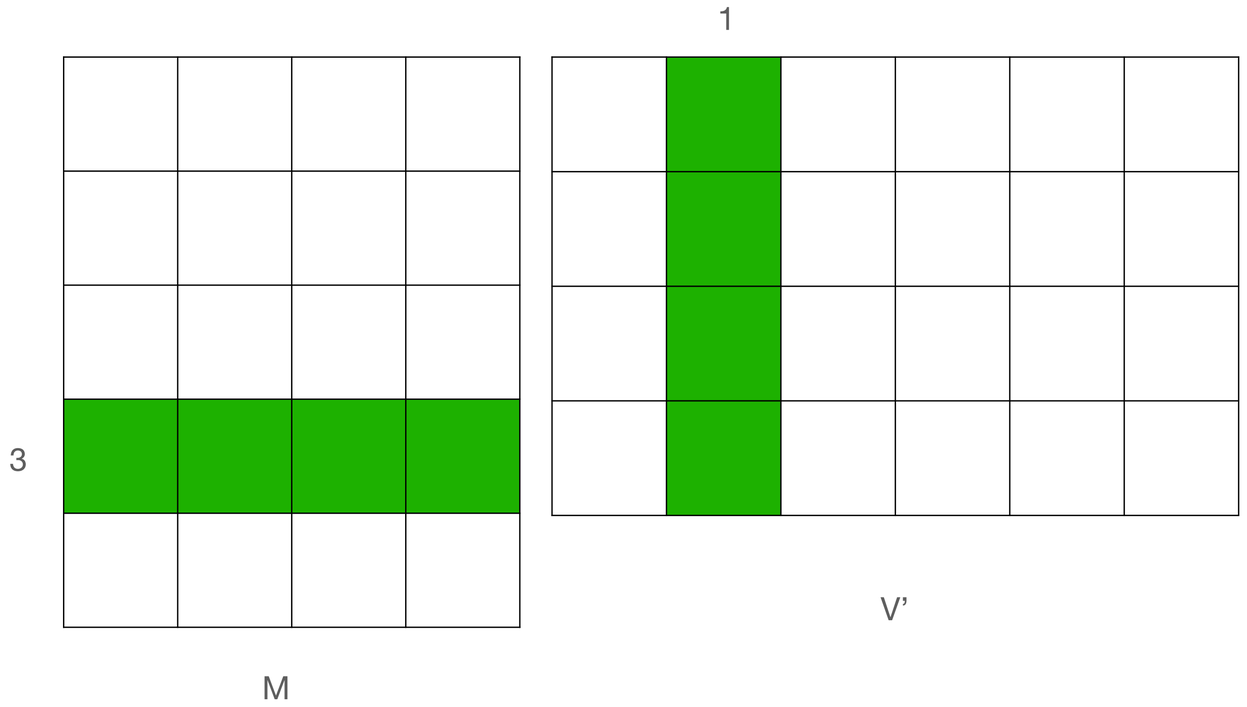
V’

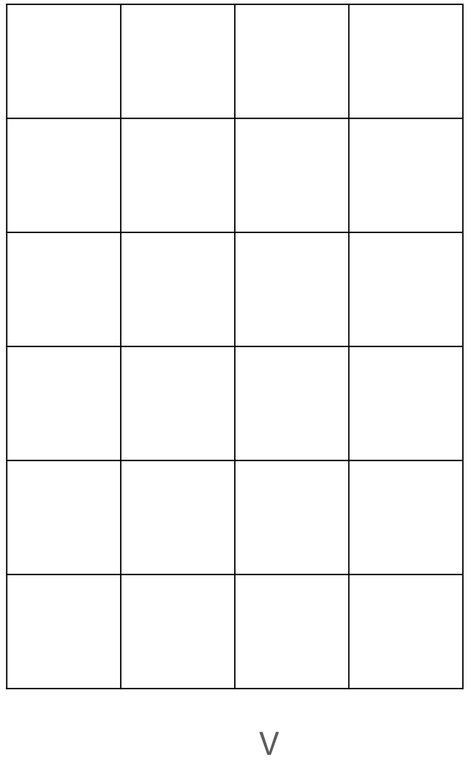


V

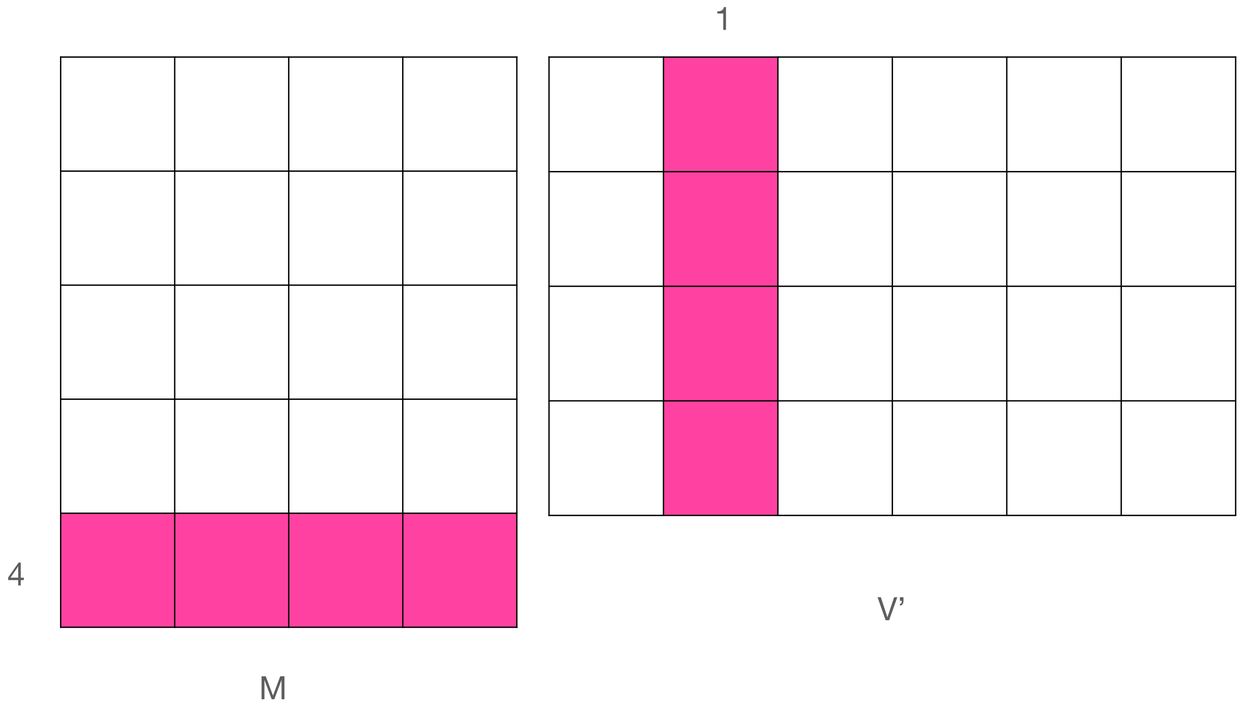
Matrices as transforms

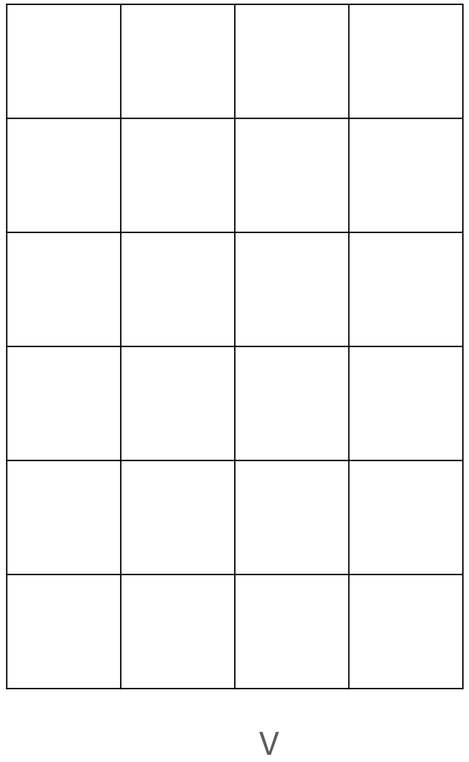




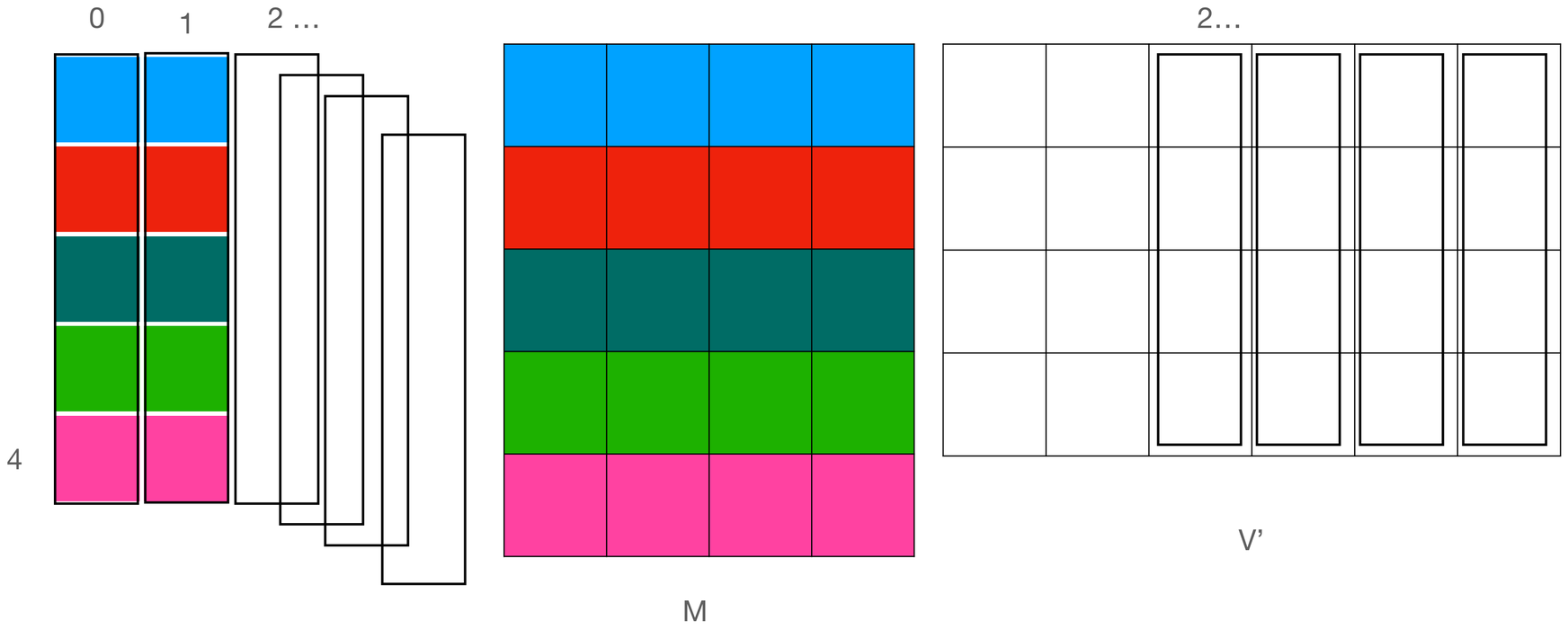


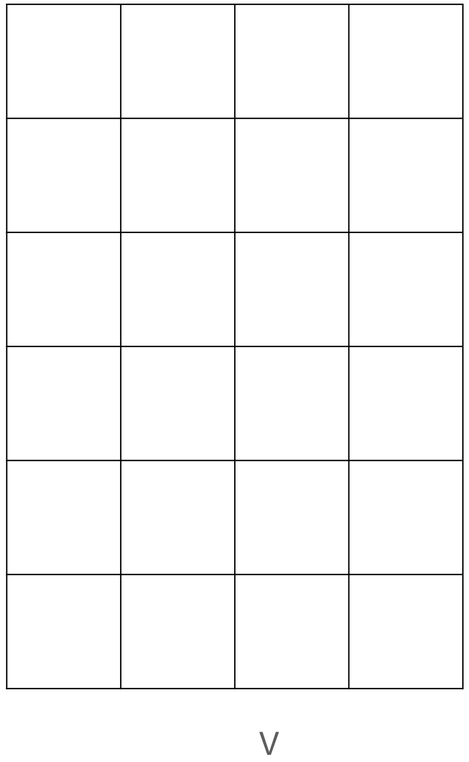
Matrices as transforms



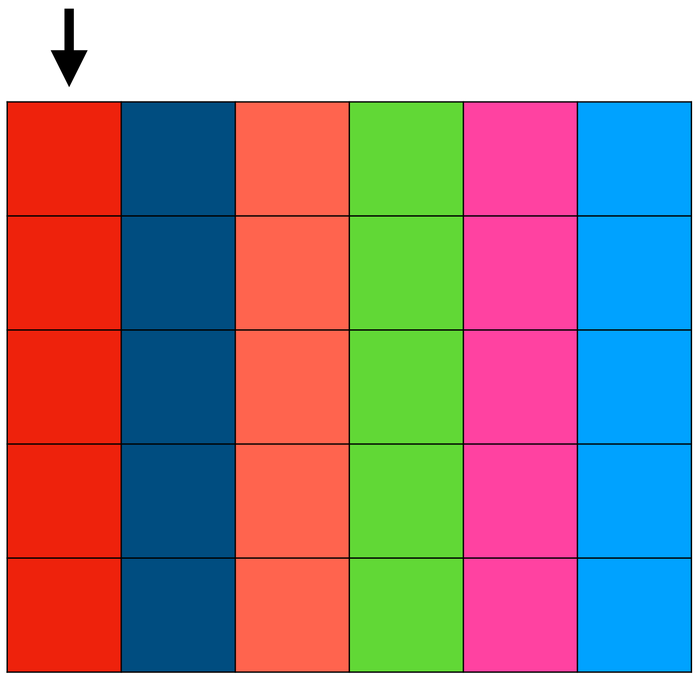


Matrices as transforms





Transformed vectors



G

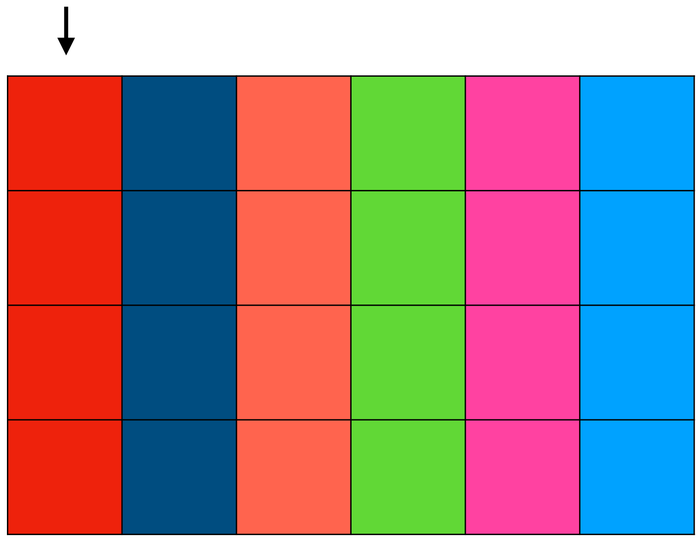
5x6

=

	1	2	3	4
1				
2				
3				
4				
5				

M

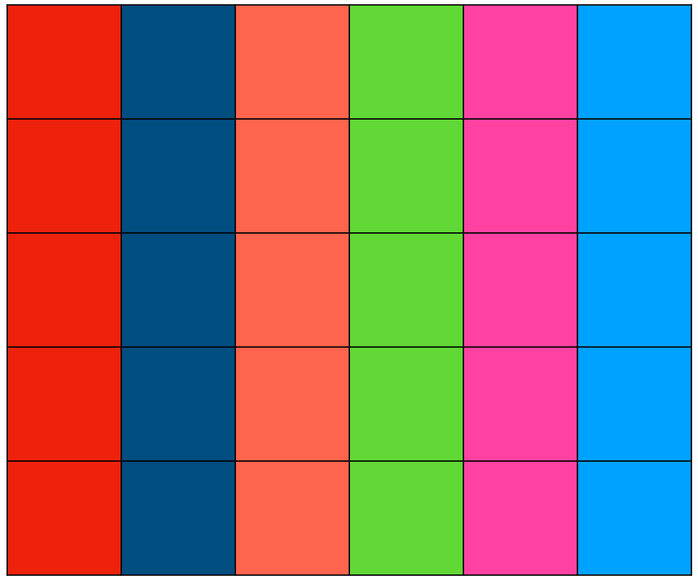
5x4



V’

4x6

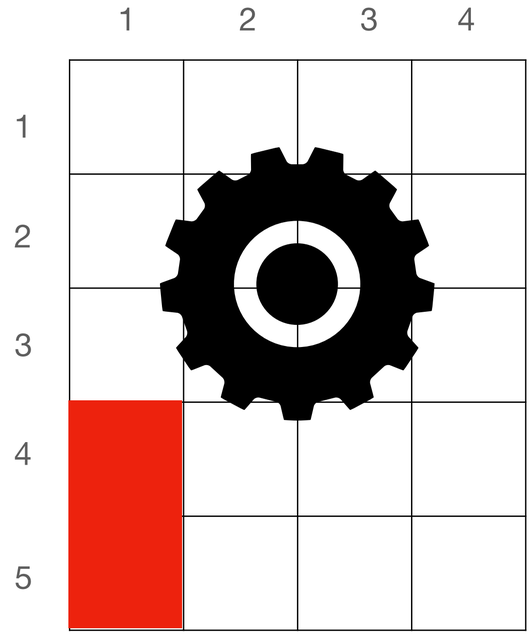
Matrix as transformer



G

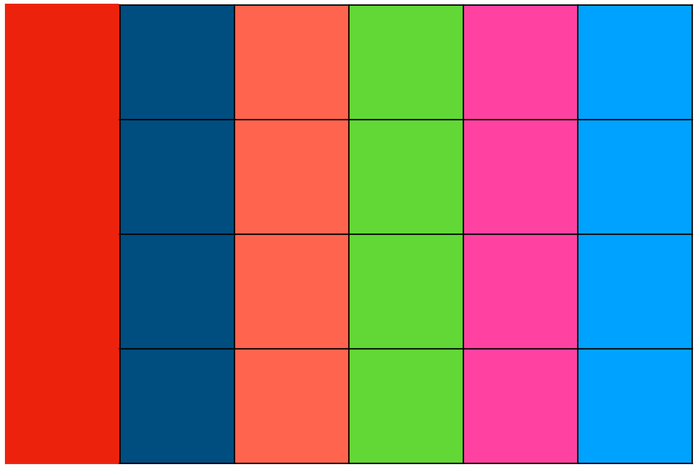
5x6

=



M

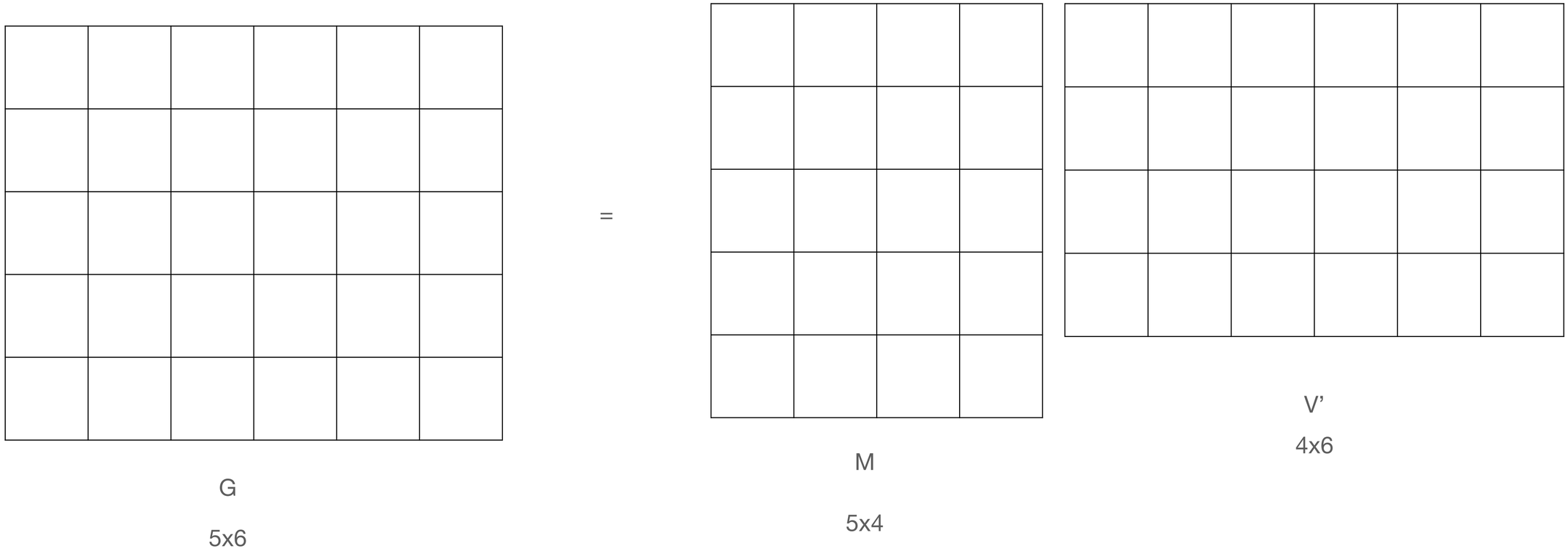
5x4



V’

4x6

Output = Transformer x Input



G

5x6

=

M

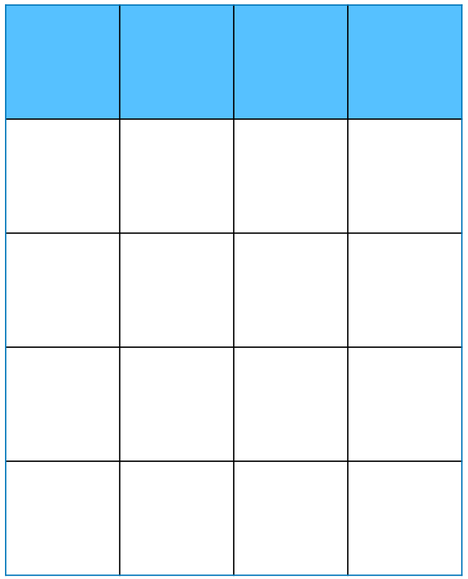
5x4

V’

4x6

Matrices as transforms

Input space is 4-d

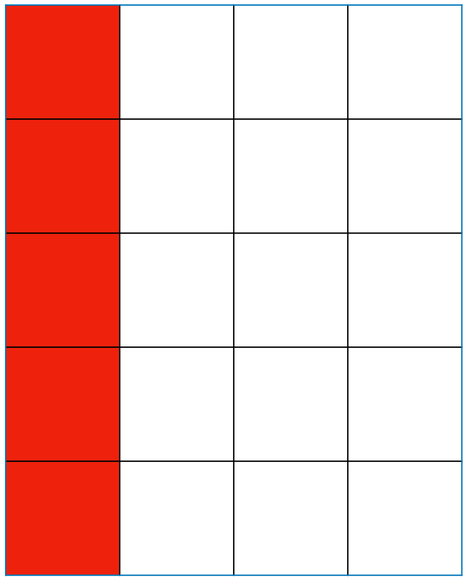


M

5x4

Matrices as transforms

Input space is 4-d



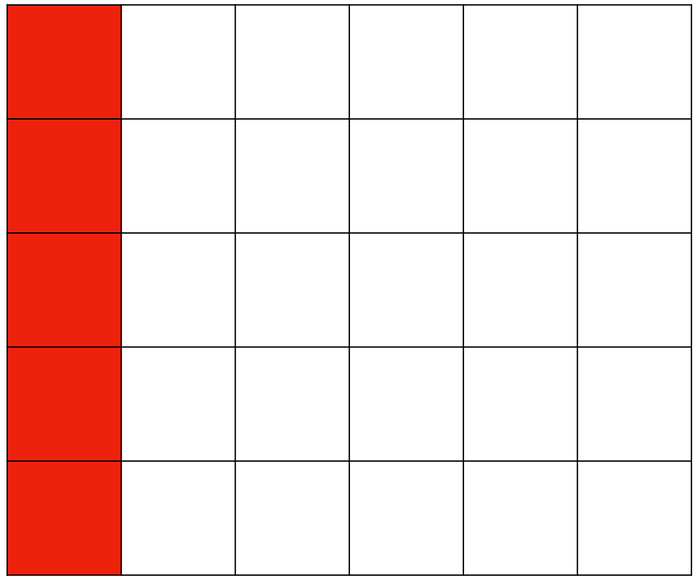
Output space is 5-d

M

5x4

Matrices as transforms

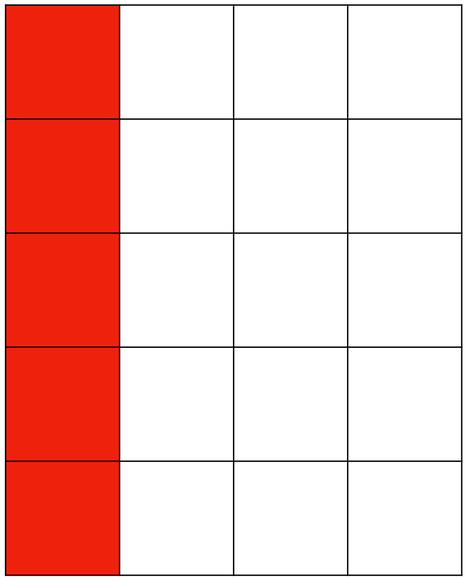
Output vector is 4-D!!



G

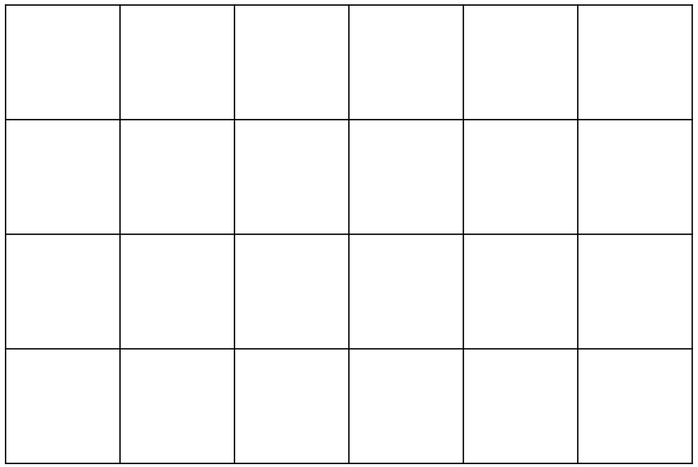
5x6

=



M

5x4



V’

4x6

Matrices as transforms

New 5-d vectors

0	1	2	3	4	5

G

5x6

There are 6 transformed vectors!

=

M

5x4

0	1	2	3	4	5

V’

4x6

There are 6 vectors to transform!

Matrices as transforms

```
Import numpy as np
SEED = 412412
my_rng = np.random.default_rng(412412)
```

New 5-d vectors

0	1	2	3	4	5

G

5x6

=

M

5x4

M = my.rng.random(5, 4)

0	1	2	3	4	5

V’

4x6

Vt = my.rng.random(4, 6)

Matrices as transforms

Import numpy as np
SEED = 412412
my_rng = np.random.default_rng(412412)

New 5-d vectors

0	1	2	3	4	5

G

5x6

G = M@Vt
G.shape

=

M

5x4

M = my.rng.random(5, 4)

0	1	2	3	4	5

V’

4x6

Vt = my.rng.random(4, 6)

See notebook