Broadcasting and `np.newaxis` in NumPy — A Visual, Head-First Guide

Why do we need `np.newaxis`?

Imagine you have:

- An array of data (say IQ samples) shaped `(2, 3)` — 2 rows, 3 columns.
- A **gain** array shaped `(2,)` — just 2 elements, one gain value per row.

You want to multiply each row of IQ data by its gain. But shapes don't match!

The Problem: Shape Mismatch

Array	Shape	What it means
IQ data	(2, 3)	2 rows, 3 columns
Gain vector	(2,)	1 dimension, 2 elements

Question: How do we multiply row-wise when the gain vector has no "columns"?

Enter `np.newaxis`

Using `np.newaxis` adds a **new dimension**:

```
gain[:, np.newaxis]
```

Changes shape from `(2,)` to `(2, 1)` — 2 rows, 1 column.

Visualizing the Shapes

Gain vector original: shape (2,)	After `np.newaxis`: shape (2,1)
`[1, 0.5]`	`[[1], [0.5]]`
(just two numbers in a flat list)	(two rows, one column)

How does multiplication work now?

You want to multiply:

- IQ data: `(2, 3)`
- Gain vector with new axis: `(2, 1)`

Can they multiply elementwise?

NumPy Broadcasting Rules Simplified:

- Compare shapes **from the right** (end of shape tuple).
- Dimensions must either be equal or one must be 1.
- If one dimension is 1, it **stretches** to match the other.

Compare shapes side by side:

Dimension (from right)	IQ data size	Gain vector size	Action
-1 (last dimension)	3	1	Gain is stretched to 3
-2	2	2	Matches exactly

What does “stretching” mean?

Logical stretching, not actual copying!

Gain vector `(2,1)`	Broadcasted to `(2,3)` logically:
`[[1], [0.5]]`	`[[1, 1, 1], [0.5, 0.5, 0.5]]`

Now you can multiply elementwise:

IQ data	Gain broadcasted	Res
`[[a, b, c], [d, e, f]]`	`[[1, 1, 1], [0.5, 0.5, 0.5]]`	`[[a*1, b*1, c*1], [d`

Why is this useful?

Without adding the new axis:

- NumPy can’t multiply `(2,3)` and `(2,)` directly.
- You get an error or wrong results.

Adding `np.newaxis` lets you:

- Keep your `gain` as a vector of length 2.
- Multiply it row-wise with your 2D IQ data easily.

Summary Table

Concept	What it means	Example
`gain.shape = (2,)`	1D array with 2 elements	`[1, 0.5]`
`gain[:, np.newaxis]`	Add a new dimension,	`[[1], [0.5]]`

Concept	What it means	Example
	shape becomes (2,1)	
Broadcasting	Dimensions with 1 stretch to match other	`[[1,1,1], [0.5,0.5,0.5]]`
Final multiplication	Multiply each IQ row by corresponding gain	`iq * gain[:, np.newaxis]`

Visual Diagram

```
Before:

iq shape: (2,3)              gain shape: (2,)

[[ a,  b,  c],             [g0, g1]
 [ d,  e,  f]]

After gain[:, np.newaxis]:

gain shape: (2,1)

[[g0],
 [g1]]

Broadcasted for multiplication:

[[g0, g0, g0],
 [g1, g1, g1]]

Multiply elementwise:

[[a*g0,  b*g0,  c*g0],
 [d*g1,  e*g1,  f*g1]]
```

Quick code snippet you can run:

```
import numpy as np

iq = np.array([[1, 2, 3], [4, 5, 6]])
gain = np.array([1, 0.5])

print("iq shape:", iq.shape)
print("gain shape:", gain.shape)

gain_expanded = gain[:, np.newaxis]
print("gain expanded shape:", gain_expanded.shape)

result = iq * gain_expanded
print("result:\n", result)
```