

Rethinking Arrays in R

July 2019

Davis Vaughan
@dvaughan32
Software Engineer, RStudio

	green	red	blue
peanut	15	8	12
plain	10	6	9

This is a
(2, 3) matrix



	green	red	blue
peanut	15	8	12
plain	10	6	9

This is a
(2, 3) matrix



	green	red	blue
peanut	15	8	12
plain	10	6	9

```
dim_names <- list(  
  c("pb", "plain"),  
  c("green", "red", "blue")  
)
```

```
matrix(  
  c(15, 10, 8, 6, 12, 9),  
  ncol = 3,  
  dimnames = dim_names  
)
```

	green	red	blue
peanut	15	8	12
plain	10	6	9

	green	red	blue
peanut	15	8	12
plain	10	6	9

+

green	red	blue
3	1	2

=

	green	red	blue
peanut	15	8	12
plain	10	6	9

+

green	red	blue
3	1	2

=

Error:
non-conformable
arrays

	green	red	blue
peanut	15	8	12
plain	10	6	9

+

green	red	blue
3	1	2

=

	green	red	blue
peanut	18	9	14
plain	13	7	11

Subsetting

Broadcasting

Manipulation

Subsetting

	green	red	blue
peanut	15	8	12
plain	10	6	9

bag

	green	red
peanut	15	8
plain	10	6

bag[:,1:2]

	green	red	blue
peanut	15	8	12
plain	10	6	9

bag

	green	red
peanut	15	8
plain	10	6

bag[:,1:2]

?

bag[:,1]

	green	red	blue
peanut	15	8	12
plain	10	6	9

bag

	green	red
peanut	15	8
plain	10	6

bag[:,1:2]

peanut plain

[15 10]

bag[:,1]

	green	red	blue
peanut	15	8	12
plain	10	6	9

bag

	green	red
peanut	15	8
plain	10	6

bag[,1:2]

	green
peanut	15
plain	10

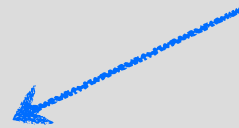
bag[, 1, drop = FALSE]

bag1		green	red	blue
plain	peanut	15	8	12
		10	6	9

bag2		green	red	blue
plain	peanut	13	6	15
		3	5	11

bags

This is a
(2, 3, 2) 3D array



bag1		green	red	blue
plain	peanut	15	8	12
	plain	10	6	9

bag2		green	red	blue
plain	peanut	13	6	15
	plain	3	5	11

bags

bag1		green	red	blue
plain	peanut	15	8	12
	peanut	10	6	9

bag2		green	red	blue
plain	peanut	13	6	15
	peanut	3	5	11

bags

bag1		green
plain	peanut	15
	peanut	10

bag2		green
plain	peanut	13
	peanut	3

bags[, 1, , drop = FALSE]

bag1		green	red	blue
plain	peanut	15	8	12
	plain	10	6	9

bag2		green	red	blue
plain	peanut	13	6	15
	plain	3	5	11

bags

bag1		green
plain	peanut	15
	plain	10

bag2		green
plain	peanut	13
	plain	3

bags[, 1, , drop = FALSE]

bag[, 1, drop = FALSE]

The confusion?

Subsetting is not
dimensionality-stable.

rray

rray is designed to provide a
stricter array class.

Create an rray

```
library(rray)
```

```
bag
```

```
#>           green red blue  
#> peanut      15   8  12  
#> plain       10   6   9
```

```
bag_rray <- as_rray(bag)
```

```
bag_rray
```

```
#> <rray<dbl>[,3][2]>  
#>           green red blue  
#> peanut      15   8  12  
#> plain       10   6   9
```

	green	red	blue
peanut	15	8	12
plain	10	6	9

bag_rray

	green
peanut	15
plain	10

bag_rray[, 1]

	green	red
peanut	15	8
plain	10	6

bag_rray[, 1:2]

bag1			
	green	red	blue
peanut	15	8	12
plain	10	6	9

bag2			
	green	red	blue
peanut	13	6	15
plain	3	5	11

bags_rray

bag1	
	green
peanut	15
plain	10

bag2	
	green
peanut	13
plain	3

bags_rray[, 1]

bag1		
	green	red
peanut	15	8
plain	10	6

bag2		
	green	red
peanut	13	6
plain	3	5

bags_rray[, 1:2]

Broadcasting

Broadcasting has to do with

1) increasing dimensionality

2) recycling dimensions

bag1		green	red	blue
plain	peanut	15	8	12
		10	6	9

bag2		green	red	blue
plain	peanut	13	6	15
		3	5	11

bags (2, 3, 2)

+

green	red	blue
3	1	2

extra (1, 3)

=

Error:
non-conformable
arrays

bag1		green	red	blue
peanut	plain	15	8	12
	peanut	10	6	9

bag2		green	red	blue
peanut	plain	13	6	15
	peanut	3	5	11

bags (2, 3, 2)

+

		green	red	blue
	plain	3	1	2
	peanut	3	1	2

extra (2, 3, 2)

=

bag1		green	red	blue
peanut	plain	18	9	14
	peanut	13	7	11

bag2		green	red	blue
peanut	plain	16	7	17
	peanut	6	6	13

(2, 3, 2)

How is extra reshaped
so that this works?

	# row	# col	# frame
bags	2	3	2
extra	1	3	
	?	?	?

	# row	# col	# frame
bags	2	3	2
extra	1	3	
	?	?	?

	# row	# col	# frame
bags	2	3	2
extra	1	3	1
	?	?	?

bag1

	green	red	blue
peanut	15	8	12
plain	10	6	9

bag2

	green	red	blue
peanut	13	6	15
plain	3	5	11

bags (2, 3, 2)

+

green	red	blue
3	1	2

=

extra (1, 3)

bag1

	green	red	blue
peanut	15	8	12
plain	10	6	9

bag2

	green	red	blue
peanut	13	6	15
plain	3	5	11

bags (2, 3, 2)

+

green	red	blue
3	1	2

=

extra (1, 3, 1)

	# row	# col	# frame
bags	2	3	2
extra	1	3	1
	?	?	?

	# row	# col	# frame
bags	2	3	2
extra	1 2	3	1
	? 2	?	?

bag1

	green	red	blue
peanut	15	8	12
plain	10	6	9

+

green	red	blue
3	1	2

=

bag2

	green	red	blue
peanut	13	6	15
plain	3	5	11

bags (2, 3, 2)

extra (1, 3, 1)

bag1

	green	red	blue
peanut	15	8	12
plain	10	6	9

+

green	red	blue
3	1	2
3	1	2

=

bag2

	green	red	blue
peanut	13	6	15
plain	3	5	11

bags (2, 3, 2)

extra (2, 3, 1)

	# row	# col	# frame
bags	2	3	2
extra	1 2	3	1
	? 2	?	?

	# row	# col	# frame
bags	2	3	2
extra	1 2	3	1
	? 2	? 3	?

	# row	# col	# frame
bags	2	3	2
extra	1 2	3	1 2
	2 2	2 3	2 2

bag1				
		green	red	blue
peanut		15	8	12
plain		10	6	9

+

		green	red	blue
peanut		3	1	2
plain		3	1	2

=

bag2				
		green	red	blue
peanut		13	6	15
plain		3	5	11

bags (2, 3, 2)

extra (2, 3, 2)

bag1			
	green	red	blue
peanut	15	8	12
plain	10	6	9

bag2			
	green	red	blue
peanut	13	6	15
plain	3	5	11

bags (2, 3, 2)

+

	green	red	blue
	3	1	2
	3	1	2

	green	red	blue
	3	1	2
	3	1	2

extra (2, 3, 2)

=

bag1			
	green	red	blue
peanut	18	9	14
plain	13	7	11

bag2			
	green	red	blue
peanut	16	7	17
plain	6	6	13

(2, 3, 2)

Broadcasting rules:

Match dimensionality by appending 1's

Match dimensions by recycling them

rarray broadcasts

bag

```
#>      green red blue
#> peanut  15   8  12
#> plain   10   6   9
```

extra

```
#>      green red blue
#> [1,]     3   1   2
```

bag + extra

```
#> Error in bag + extra: non-conformable arrays
```

bag_rarray + extra

```
#> <rarray<dbl>[,3][2]>
#>      green red blue
#> peanut  18   9  14
#> plain   13   7  11
```

Manipulation

rray as a toolkit

`rray_bind()`

`rray_duplicate_any()`

`rray_expand_dims()`

`rray_broadcast()`

`rray_flip()`

`rray_max()`

`rray_sum()`

`rray_mean()`

`rray_reshape()`

`rray_rotate()`

`rray_split()`

`rray_tile()`

`rray_unique()`

`...`

The best part?

The best part?

They all work with base R.

rray as a toolkit

`rray_bind()`

`rray_duplicate_any()`

`rray_expand_dims()`

`rray_broadcast()`

`rray_flip()`

`rray_max()`

`rray_sum()`

`rray_mean()`

`rray_reshape()`

`rray_rotate()`

`rray_split()`

`rray_tile()`

`rray_unique()`

`...`

rray as a toolkit

`rray_bind()`

`rray_duplicate_any()`

`rray_expand_dims()`

`rray_broadcast()`

`rray_flip()`

`rray_max()`

`rray_sum()`

`rray_mean()`

`rray_reshape()`

`rray_rotate()`

`rray_split()`

`rray_tile()`

`rray_unique()`

`...`

Compute proportions

- 1) Overall
- 2) By filling
- 3) By color

	green	red	blue
peanut	15	8	12
plain	10	6	9

	green	red	blue
peanut	15	8	12
plain	10	6	9

Overall

`bag / sum(bag)`

	green	red	blue
peanut	0.25	0.13	0.20
plain	0.17	0.10	0.15

	green	red	blue
peanut	15	8	12
plain	10	6	9

By Filling

`sweep(bag, 1, apply(bag, 1, sum), "/")`

	green	red	blue
peanut	0.43	0.23	0.34
plain	0.40	0.24	0.36

	green	red	blue
peanut	15	8	12
plain	10	6	9

By Color

`sweep(bag, 2, apply(bag, 2, sum), "/")`

	green	red	blue
peanut	0.60	0.57	0.57
plain	0.40	0.43	0.43

	green	red	blue
peanut	15	8	12
plain	10	6	9

Overall

`bag / rray_sum(bag)`

	green	red	blue
peanut	0.25	0.13	0.20
plain	0.17	0.10	0.15

	green	red	blue
peanut	15	8	12
plain	10	6	9

By Filling

`bag / rray_sum(bag, axes = 2)`

	green	red	blue
peanut	0.43	0.23	0.34
plain	0.40	0.24	0.36

	green	red	blue
peanut	15	8	12
plain	10	6	9

By Color

`bag / rray_sum(bag, axes = 1)`

	green	red	blue
peanut	0.60	0.57	0.57
plain	0.40	0.43	0.43

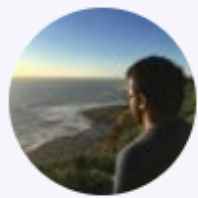
In conclusion...

1) Stricter rray class

2) Broadcasting

3) Toolkit

Questions?



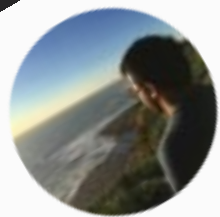
Kanishka Misra 🧠📈📊 @iamasharkskin · Jun 19

Implemented a basic feedforward neural network in R using @dvaughan32's rray and the R6 OOP system. Will start working on a blog post (or two) soon.

GitHub
<https://github.com/r-lib/rray>

Website
<https://rray.r-lib.org>

Powered by: xtensor
<https://github.com/QuantStack/xtensor>



Kanishka Misra 🧠📈📊
Replying to @dvaughan32
rray is <3