Data Structures in R: Arrays and Factors

Stat 133 by Gaston Sanchez

Data Types & Vectors recap

Data Types (primitives)

```
1L        # integer
2.5       # double (real)
TRUE      # logical
"hello"   # character
1 + 3i    # complex
```

Fundamental concepts

Atomic structures

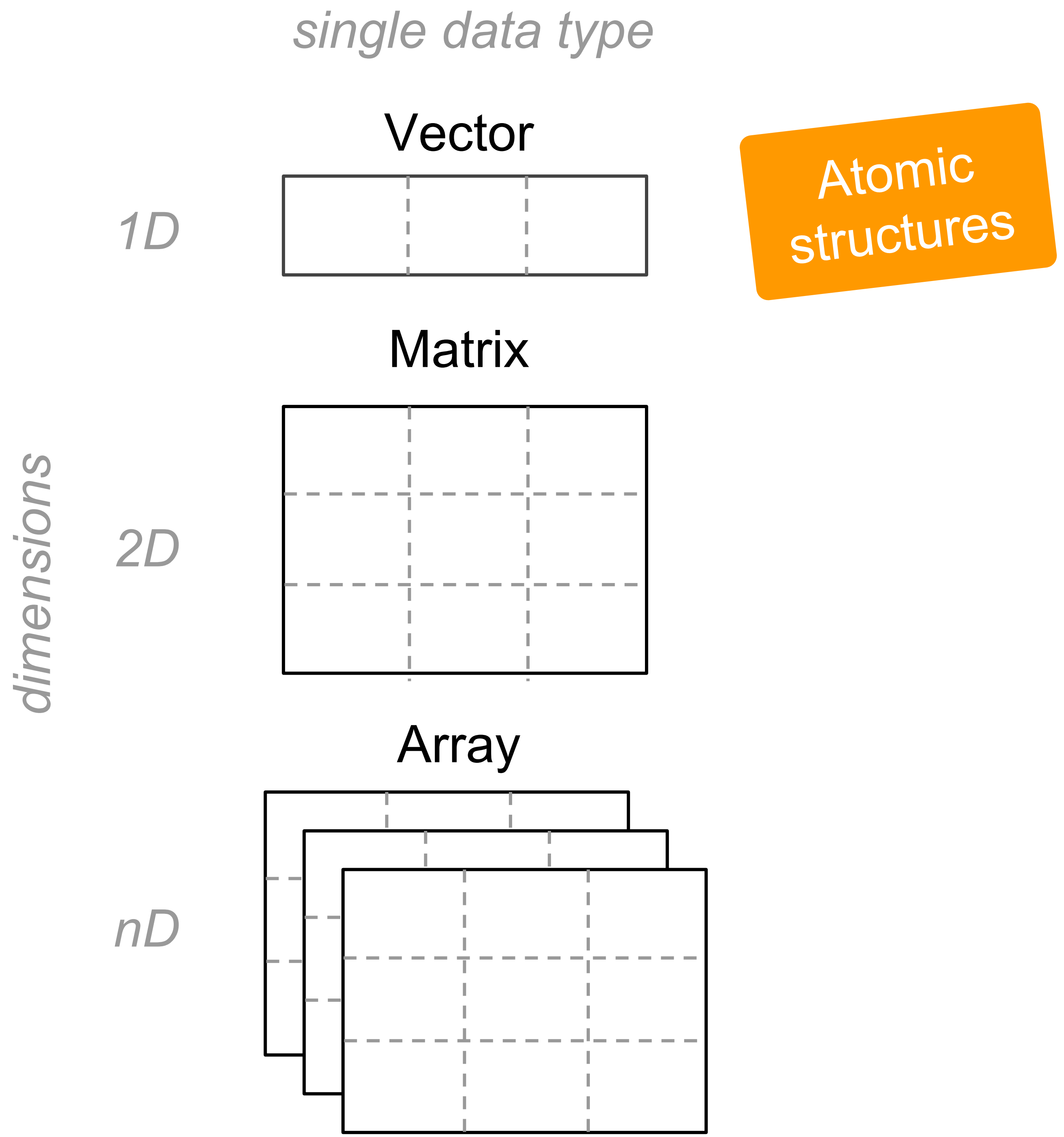
Coercion

Subsetting or Bracket Notation `[index]`

Vectorization

Recycling

Matrices and Arrays


single data type
Vector
1D
Atomic structures
Matrix
dimensions
2D
Array
nD

Arrays and matrices

You can transform a vector in an n-dimensional array by giving it a **dimensions** attribute

```
x <- 1:8

dim(x) <- c(2, 4)
```

Arrays and matrices

The **dimensions** attribute is a numeric vector with as many elements as desired dimensions

```
x <- 1:8

dim(x) <- c(2, 2, 2)
```

In practice, we don't really create matrices and arrays via dim()

Arrays and matrices

To have more control about how a matrix is filled, we use the function `matrix()`

```
a <- 1:8

A <- matrix(a, nrow = 2, ncol = 4)
```

About R matrices

R stores matrices as vectors.

Which means that matrices are also **atomic**.

Matrices in R are stored **column-major** (i.e. by columns).

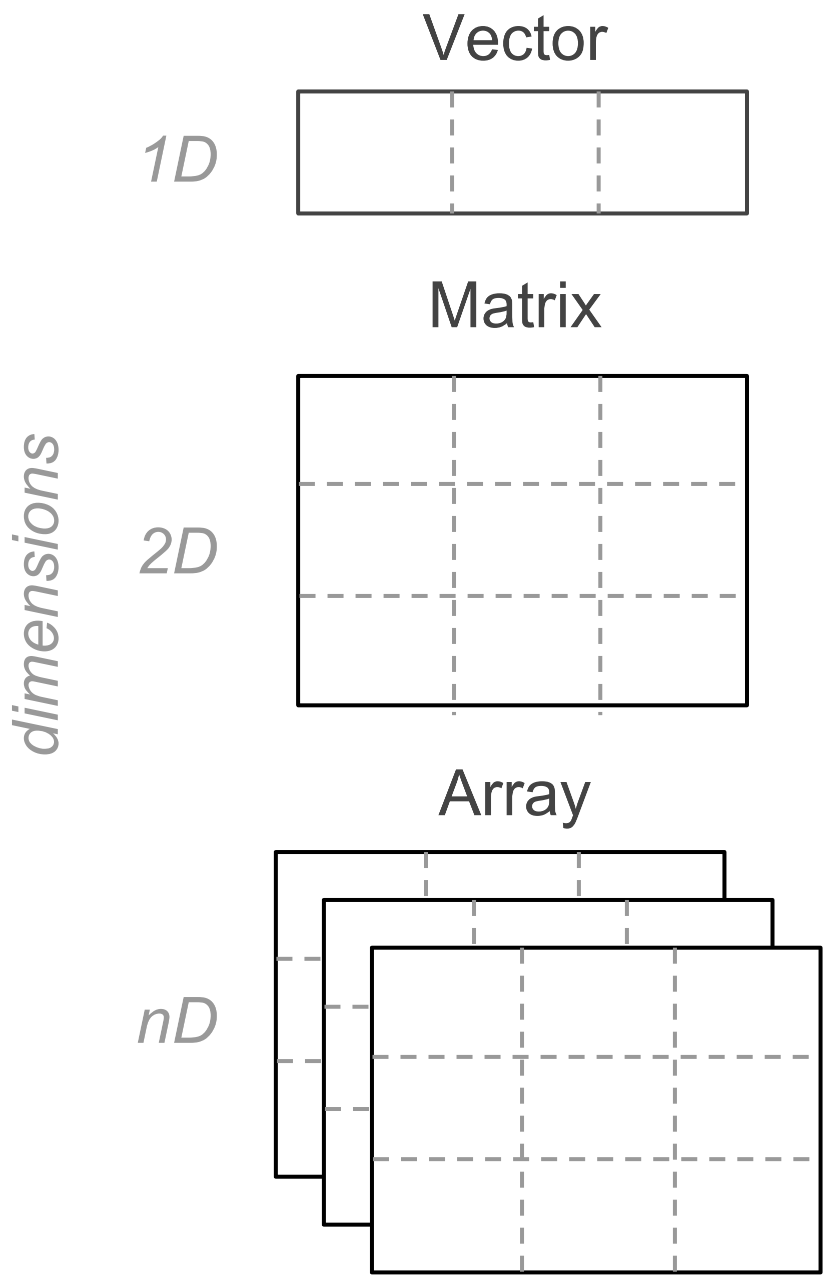
This is like Fortran, Matlab, and Julia, but not like C or Python (e.g. numpy).

Arrays and matrices

If you want to fill a matrix by rows use
`byrow = TRUE`

```
b <- 1:8

B <- matrix(a, nrow = 2, ncol = 4,
            byrow = TRUE)
```



```
object[i]
```

```
object[i,j]
```

```
object[i,j,k]

object[i,j,k,l]
```

So far

Vectors, matrices, and arrays are atomic structures (they can only store one type of data)

Many operations in R need atomic structures to make sure all values are of the same mode

In real life, however, many datasets contain multiple types of information

R provides other data structures for this purpose

Factors

R factors

Another data structure in R are **factors**

A factor is designed to handle categorical data

The name “factor” comes from “Analysis of Variance” (ANOVA) terminology

R factors

To create a factor, typically you pass a vector to the function `factor()`

```
size <- c("sm", "md", "lg", "md")

size <- factor(size)
```

About R factors

Factors are excellent for working with categorical data, especially data with an “ordinal” scale

Factors are **internally stored as vectors of integers**

Factors behave a lot like vectors

But factors have their own special properties

Codification issues

Codification

It is very common that we (humans) codify information (e.g. data, variables) in many interesting ways

It can make completely sense to us

But not necessarily to the computer

Binary scale variable

Example	R mode
`TRUE, FALSE`	logical
`0, 1`	numeric
`"yes", "no"`	character
`yes, no`	factor

Nominal scale variable

Example	R mode
`1, 2, 3`	numeric
`"blue", "white", "red"`	character
`blue, white, red`	factor

Odinal scale variable

Example	R mode
`1, 2, 3`	numeric
`"small", "medium", "large"`	character
`small, medium, large`	factor

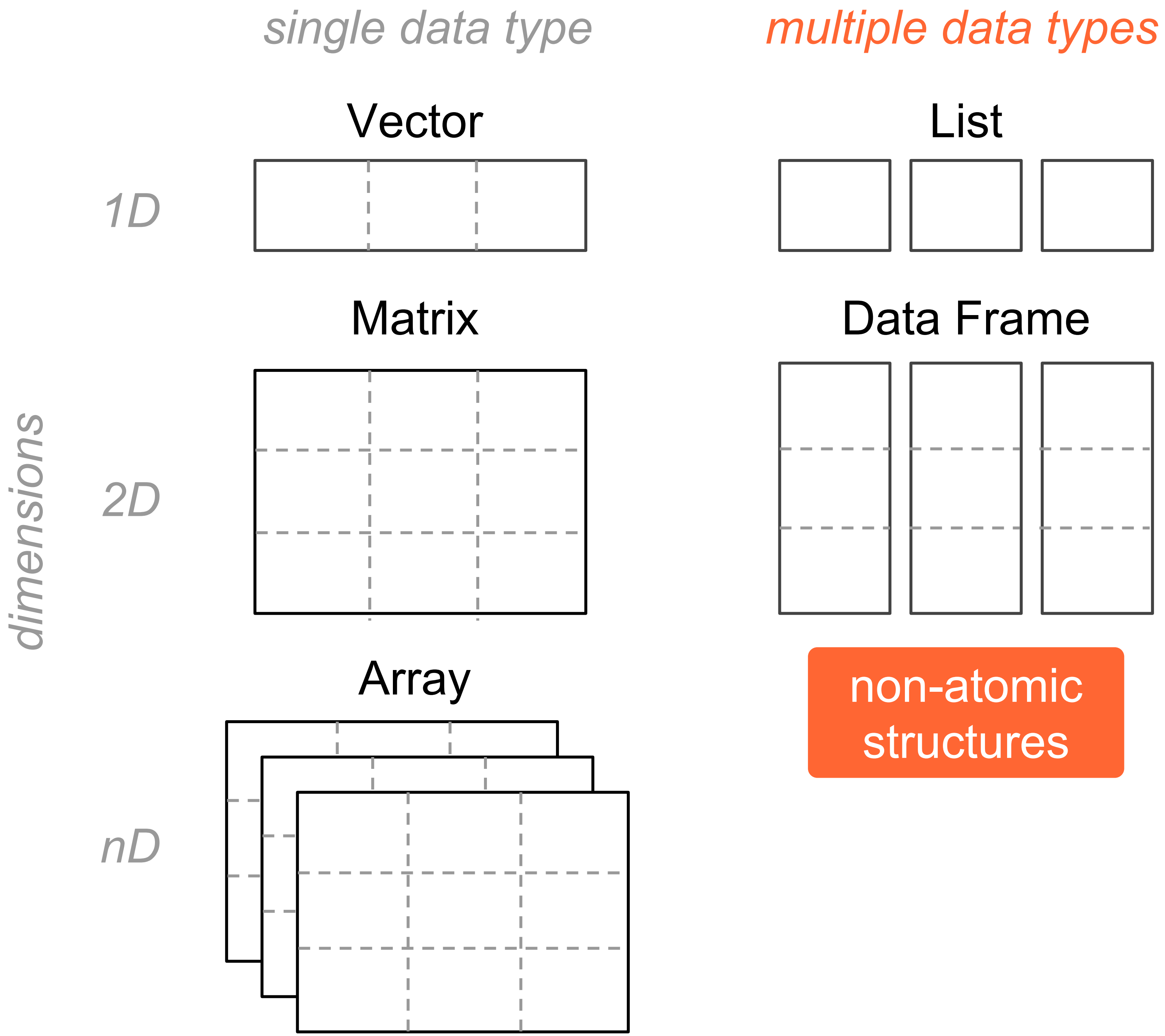
Interval / Ratio scale variables in R

Example	R mode
`1.1, -2.5, 100`	numeric
`1/4, pi, exp(1)`	numeric

Missing Values

Example	R mode
`NA`	logical
`-999`	numeric
`-99999`	numeric
`"?"`	character
`"  "`	character
`"na"`	character

Next


single data type
multiple data types
dimensions
1D
2D
nD
Vector
List
Matrix
Data Frame
Array
non-atomic structures