Vectors in R (part 2)

Stat 133 with Gaston Sanchez

Atomicity

Vectors are **atomic** structures

Examples

```r
x <- c(1, 2, 3, 4, 5)

y <- c("one", "two", "three")

z <- c(TRUE, FALSE, TRUE)
```

Atomic vectors

Vectors are atomic structures

The values in a vector must be **ALL** of the same type!

Either all integers, or reals, or complex, or characters, or logicals

You CANNOT have a vector of different data types

Coercion

What happens if you mix different data values in a vector?

Mixing data types within a vector?

```
x <- c(1, 2, 3, "four", "five")


y <- c(TRUE, FALSE, 3, 4)


z <- c(TRUE, 1L, 2 + 3i, pi)
```

Implicit Coercion

If you mix different data values, R will **implicitly coerce** them so they are ALL of the same type

```
x <- c(1, 2, 3, "four", "five")


y <- c(TRUE, FALSE, 3, 4)
```

How does R coerce data types in vectors?

R follows two basic rules of implicit coercion

1) If a character is present, R will coerce everything else to characters

2) If a vector contains logicals and numbers, R will convert the logicals to numbers (TRUE to 1, FALSE to 0)

Hierarchy of data types

Logical < Integer < Double < Character

Coercion functions

R provides a set of **explicit** coercion functions that allow you to “convert” one type of data into another

- `as.character()`
- `as.numeric()`
- `as.double()`
- `as.integer()`
- `as.logical()`

Vectorization

Vectorization

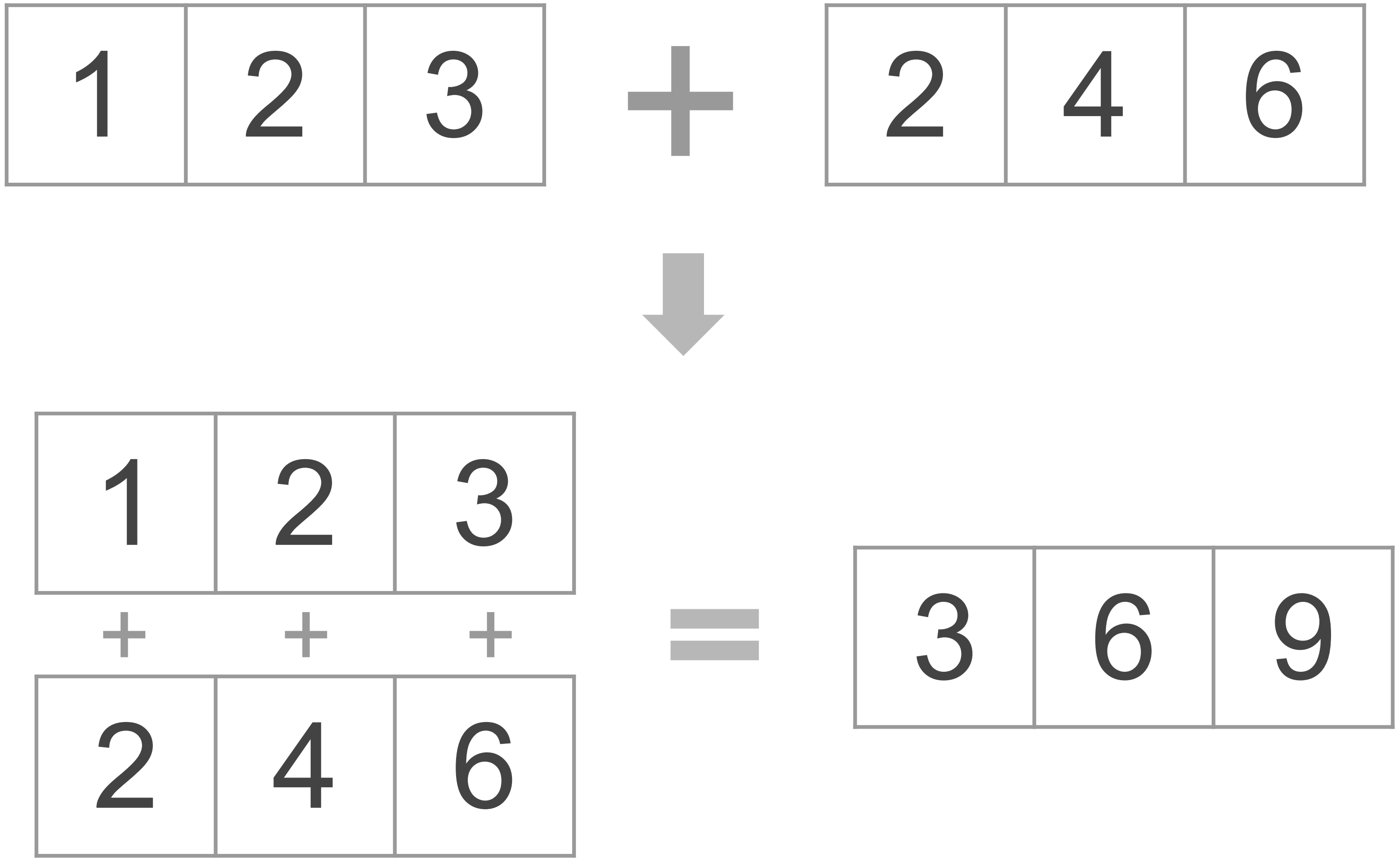
A **vectorized** computation is any computation that when applied to a vector operates on all of its elements

```
c(1, 2, 3) + c(3, 2, 1)

c(1, 2, 3) * c(3, 2, 1)

c(1, 2, 3) ^ c(3, 2, 1)
```

Vectorized code



Recycling

Recycling

When vectorized computations are applied, some conflicts may occur when dealing with two vectors of different length

```
c(2, 1) + c(1, 2, 3)

c(1, 2, 3, 4) + c(1, 2)
```

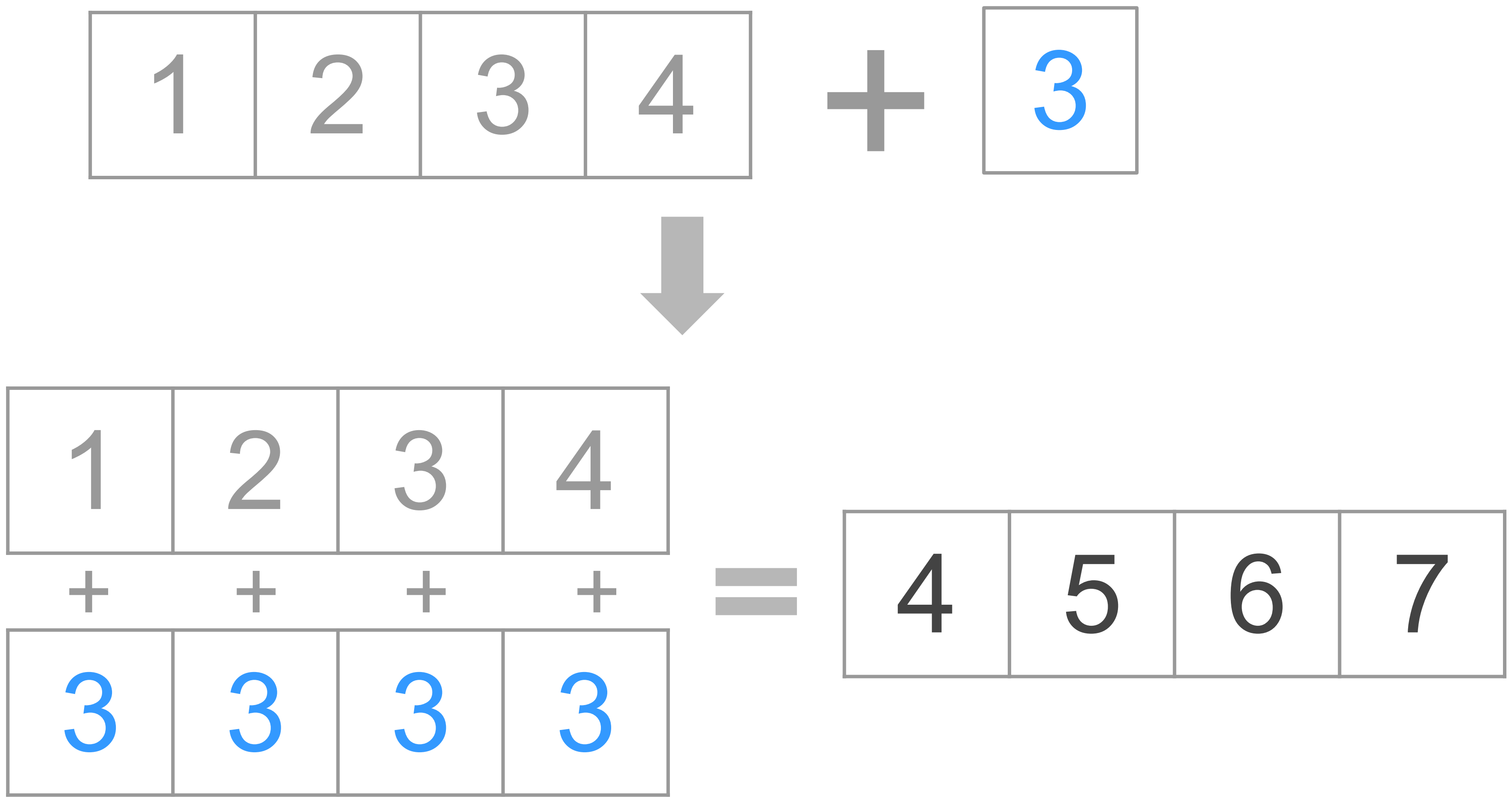
Recycling Rule

The recycling rule can be very useful, like when operating between a vector and a “scalar”

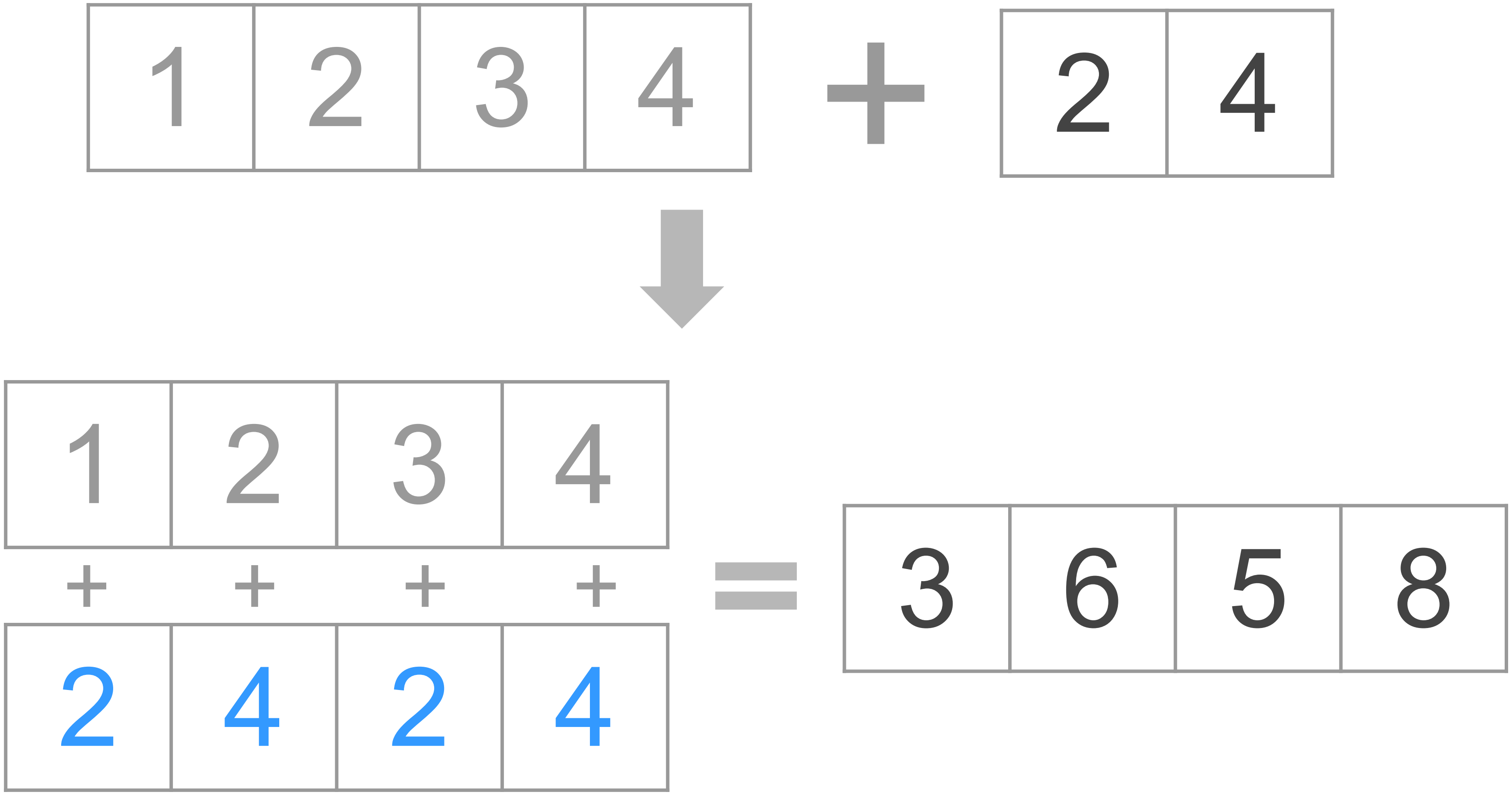
```
x <- c(2, 4, 6, 8)

x + 3
```

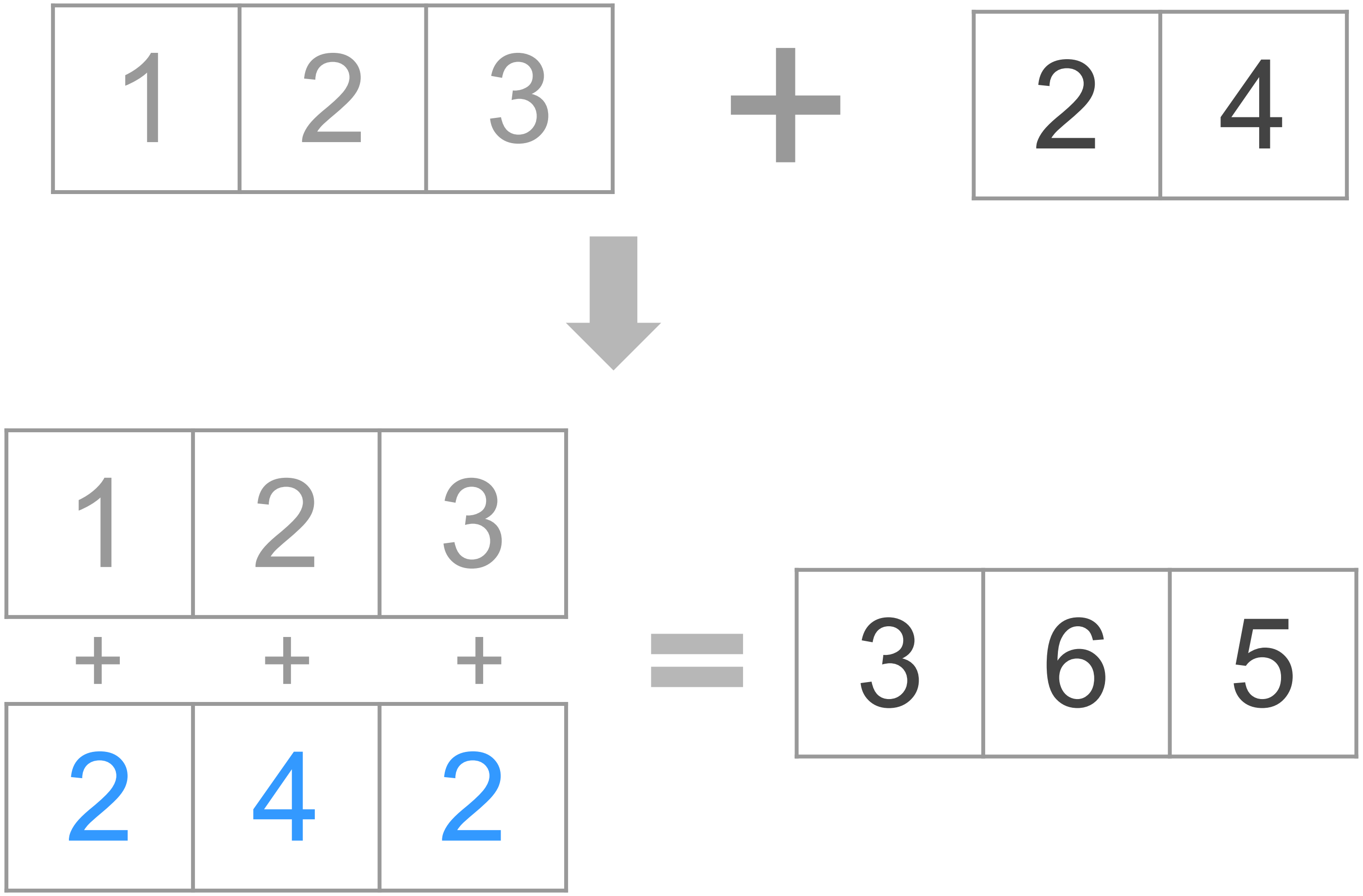
Recycling (and vectorization)



Recycling (and vectorization)



Recycling (and vectorization)



Subsetting and Indexing

Bracket notation for vectors

`vec[index]`

Bracket Notation System

To extract values from R objects use brackets: **[ ]**

Inside the brackets specify vector(s) of indices

Use as many indices, separated by commas, as dimensions in the object

Vector(s) of indices can be *numbers*, *logicals*, and sometimes *characters*

Bracket Notation System

```
# some vector
x <- c(2, 4, 6, 8)

# adding names
names(x) <- letters[1:4]
```

Numeric index

```
# first element
x[1]

# second element
x[2]

# last element
x[length(x)]
```

Numeric index

```
# first 3 elements
x[1:3]

# non-consecutive elements
x[c(1, 3)]

# different order
x[c(3, 2, 4, 1)]
```

Logical index

```
# first element
x[c(TRUE, FALSE, FALSE, FALSE)]

# elements equal to 2
x[x == 2]

# elements different to 2
x[x != 2]
```

Character index

```
# element names "a"
x["a"]

# "b" and "d"
x[c("b", "d")]

# what about this?
x[rep("a", 5)]
```

Logical index

```
# elements greater than 1
x[x > 1]

# try this
x[TRUE]

# what about this?
x[as.logical(c(0, 1, pi, -10))]
```