Stat405

Data structures

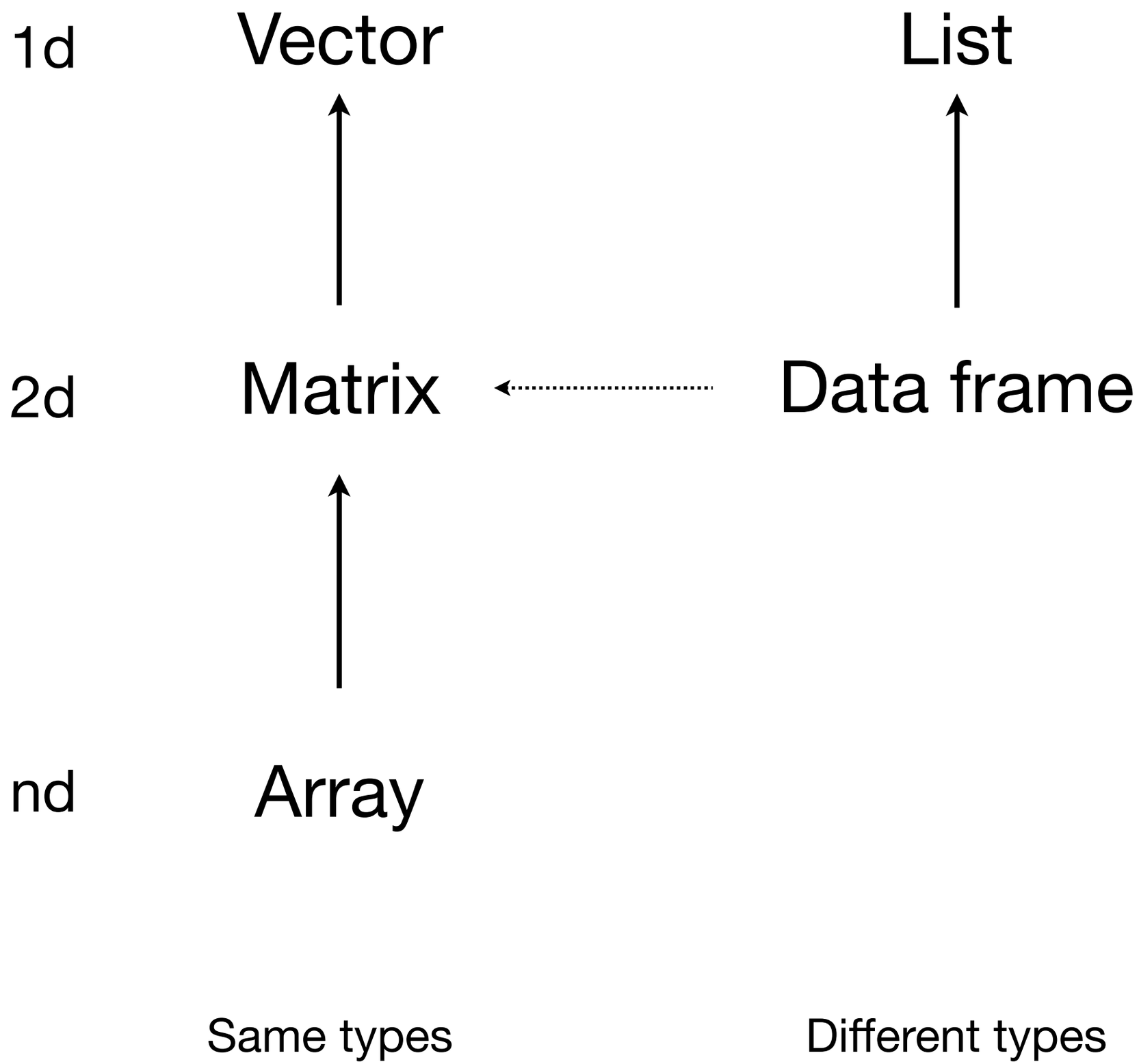
Hadley Wickham

1. Atomic vectors

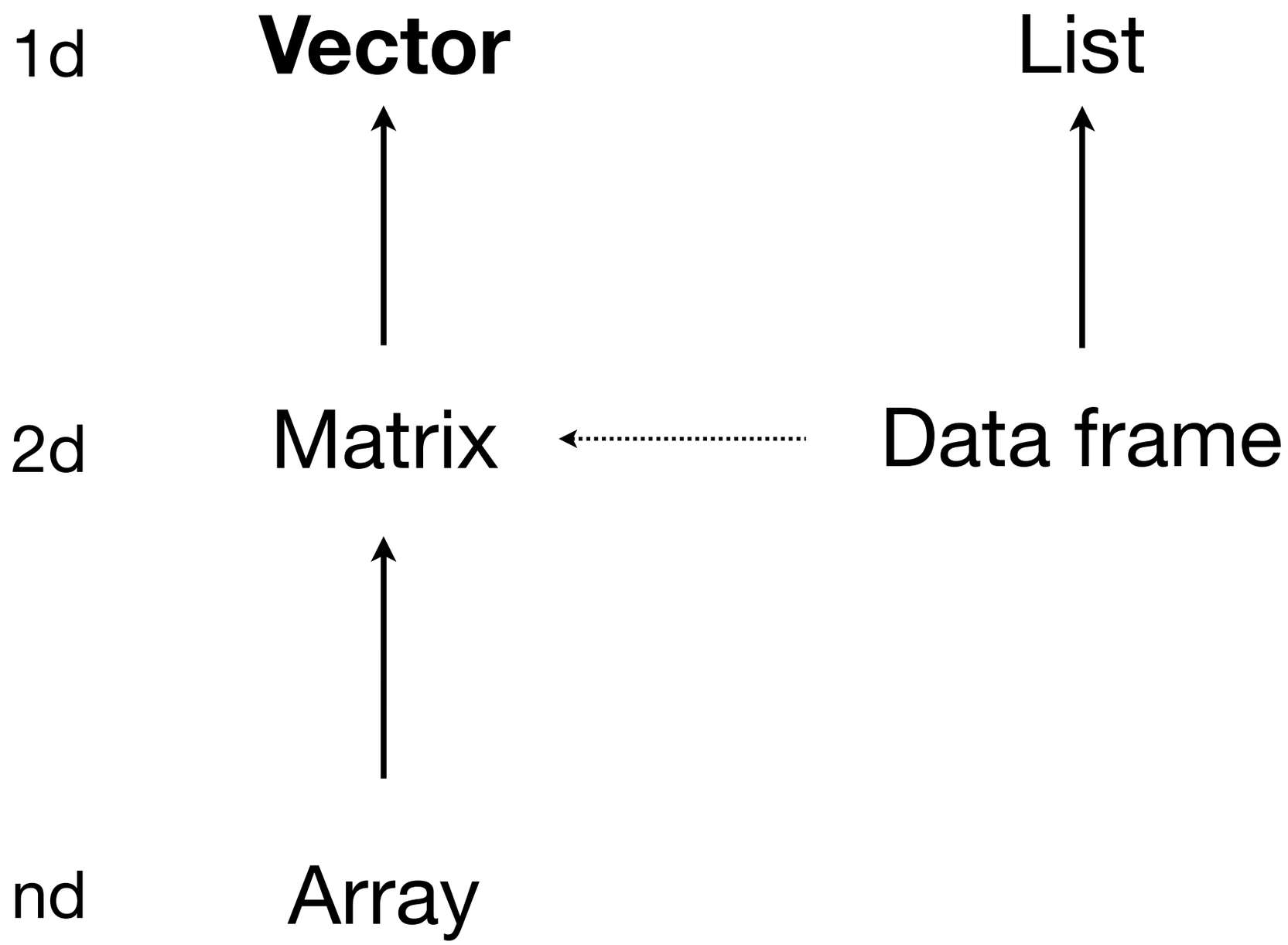
2. Matrices & arrays

3. Lists & data.frames

str()


1d
Vector
List
2d
Matrix
Data frame
nd
Array
Same types
Different types

Atomic vectors


1d
Vector
List
2d
Matrix
Data frame
nd
Array
Same types
Different types

Properties

- `mode()`
- `length()`
- `names()`

Type	Creation	Checking	Coercion
character	`c("a", 'b')`	`as.character`	`is.character`
numeric	`c(1, 2)`	`as.numeric`	`is.numeric`
integer	`c(1L, 2L)`	`as.integer`	`is.integer`
logical	`c(T, F)`	`as.logical`	`is.logical`

Your turn

What is happening in the following cases?

```
104 & 2 < 4
```

```
mean(diamonds$cut == "Good")
```

```
c(T, F, T, T, "F")
```

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

```
2L / 3L
```

```
# Automatic coercion:
c("a", 1)
c("a", 1L)
c("a", T)
c(1, 1L)
c(1, T)
c(1L, T)

# character > numeric > integer > logical

# Numeric vector operations will coerce logical and
# integer to numeric. Logical vector operations will
# coerce integer and numeric to logical.
```

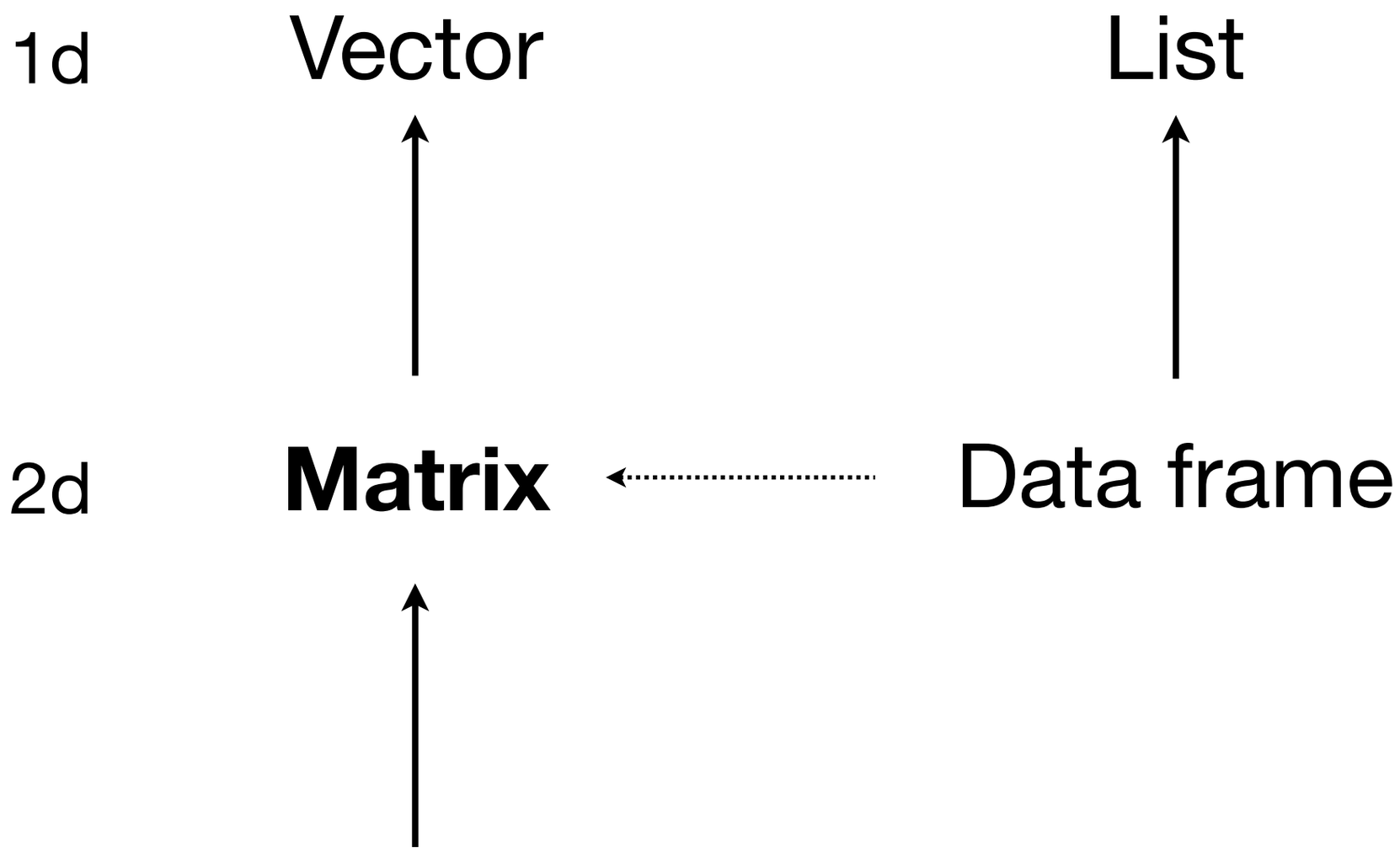
Recall

What are the six things you can use to can subset a vector?

Brainstorm with your neighbour for 30s.

blank	include all
integer	**+ve**: include **0**: nothing **-ve**: exclude
logical	keep TRUEs
character	lookup by name

Matrices and arrays


1d
Vector
List
2d
Matrix
Data frame
nd
Array
Same types
Different types

Properties

- `mode()`
- `length()` →
  `nrow()`, `ncol()` →
  `dim()`
- `names()` →
  `colnames()`, `rownames()` →
  `dimnames()`

```r
# Can create from a vector

a <- seq_len(12)

matrix(a, nrow = 3)
matrix(a, ncol = 4)

array(a, c(1, 12))
array(a, c(4, 3))
array(a, c(3, 2, 2))
```

```r
# Can create by combining existing

a <- 1:5
b <- 5:1
c <- matrix(sample(25), ncol = 5)

cbind(a, b)
cbind(a, c)
rbind(a, b)
rbind(a, c)

# For arrays, need the special abind package
```

```r
b <- seq_len(10)
a <- letters[b]

# What sort of matrix does this create?
rbind(a, b)
cbind(a, b)

# Why would you want to use a data frame here?
# How would you create it?
```

```
# Checking

a <- seq_len(12)
b1 <- matrix(a, nrow = 3)
b2 <- array(a, c(3, 4))
c <- array(a, c(3, 2, 2))
```

Your turn: complete the table:

	a	b1	b2	c
`is.vector`				
`is.matrix`				
`is.array`				

	a	b1	b2	c
is.vector	T	F	F	F
is.matrix	F	T	T	F
is.array	F	T	T	T

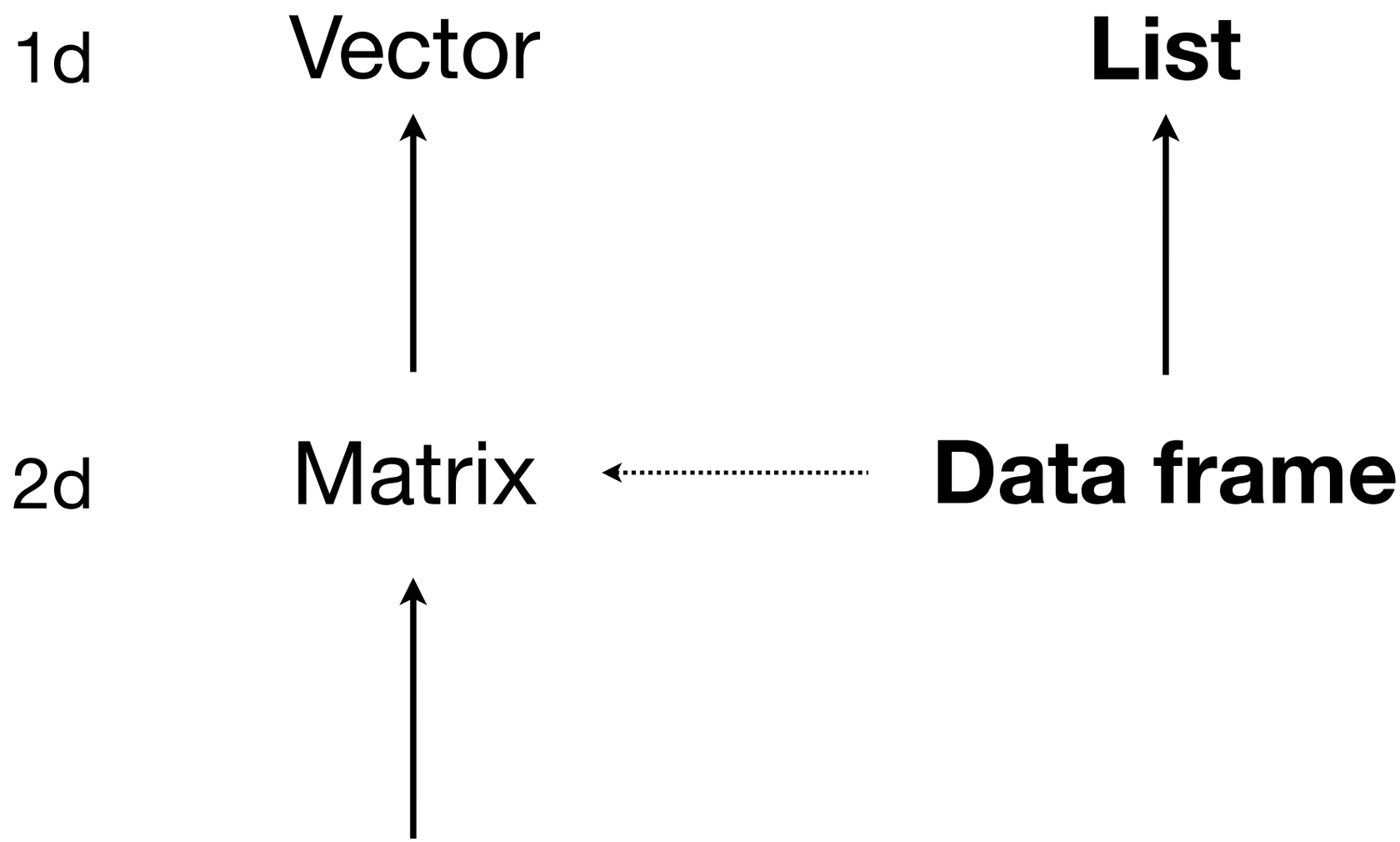
```r
x <- sample(12)

# What's the difference between a & b?
a <- matrix(x, 4, 3)
b <- array(x, c(4, 3))

# What's the difference between x & y
y <- matrix(x, 12)

# How are these subsetting operations different?
a[, 1]
a[, 1, drop = FALSE]
a[1, ]
a[1, , drop = FALSE]
```

		`x[1]`
	`y[1,]` `y[, 1]`	`y[1, 1]`

`z[1, , ]`
`z[, 1, ]`
`z[, , 1]`
	`z[1, 1, ]`
	`z[1, , 1]`
	`z[, 1, 1]`
		`z[1, 1, 1]`

add `, drop = F` to preserve original dimensions

Lists & Dataframes


1d
Vector
List
2d
Matrix
Data frame
nd
Array
Same types
Different types

List properties

- `mode()`
- `length()`
- `names()`
- New property: each element of a list can be a different type (even another list). Lists are **recursive**.

```
# Creation
list(c("a", "b"), 1:10, c(F, T, F))


list(1:10)
# cf.
as.list(1:10)


x <- as.list(1:10)
as.vector(x)
as.numeric(x)
unlist(x) # generic
```

If list `x` is a train carrying objects, then `x[[5]]` is the object in car 5; `x[4:6]` is a train of cars 4-6.

http://twitter.com/#!/RLangTip/status/118339256388304896

Data frame properties

- A data frame is a cross between a list and a matrix
- It is a list of columns (variables), each a vector of the same length
- Because the vectors are the same length it behaves like a matrix: 2d subsetting, `nrow`, `ncol`, `colnames`, etc.

```
# Creating a data frame
data.frame(x = 1:10, y = letters[1:10])

# But usually

read.table
read.csv
mutate
expand.grid
# ...
```

```r
# How do you convert a matrix to a data frame?
# How do you convert a data frame to a matrix?
# In which direction do you lose data?

# What do these subsetting operations do?
# Why do they work?  (Remember to use str)
diamonds[1]
diamonds[[1]]
diamonds[["cut"]]
diamonds[["cut"]][1:10]
diamonds$cut[1:10]
```

```
diamonds$x
```

```
diamonds[["x"]]
```

```
load(url("http://stat405.had.co.nz/data/quiz.rdata"))

# What is a?  What is b?
# How are they different?  How are they similar?
# How can you turn a in to b?
# How can you turn b in to a?

# What are c, d, and e?
# How are they different?  How are they similar?
# How can you turn one into another?

# What is f?
# How can you extract the first element?
# How can you extract the first value in the first
# element?
```

```
# a is numeric vector, containing the numbers 1 to 10
# b is a list of numeric scalars
# they contain the same values, but in a different format
identical(a[1], b[[1]])
identical(a, unlist(b))
identical(b, as.list(a))

# c is a named list
# d is a data.frame
# e is a numeric matrix
# From most to least general: c, d, e
identical(c, as.list(d))
identical(d, as.data.frame(c))
identical(e, data.matrix(d))
```

```
# f is a list of matrices of different dimensions

f[[1]]
f[[1]][1, 2]
```