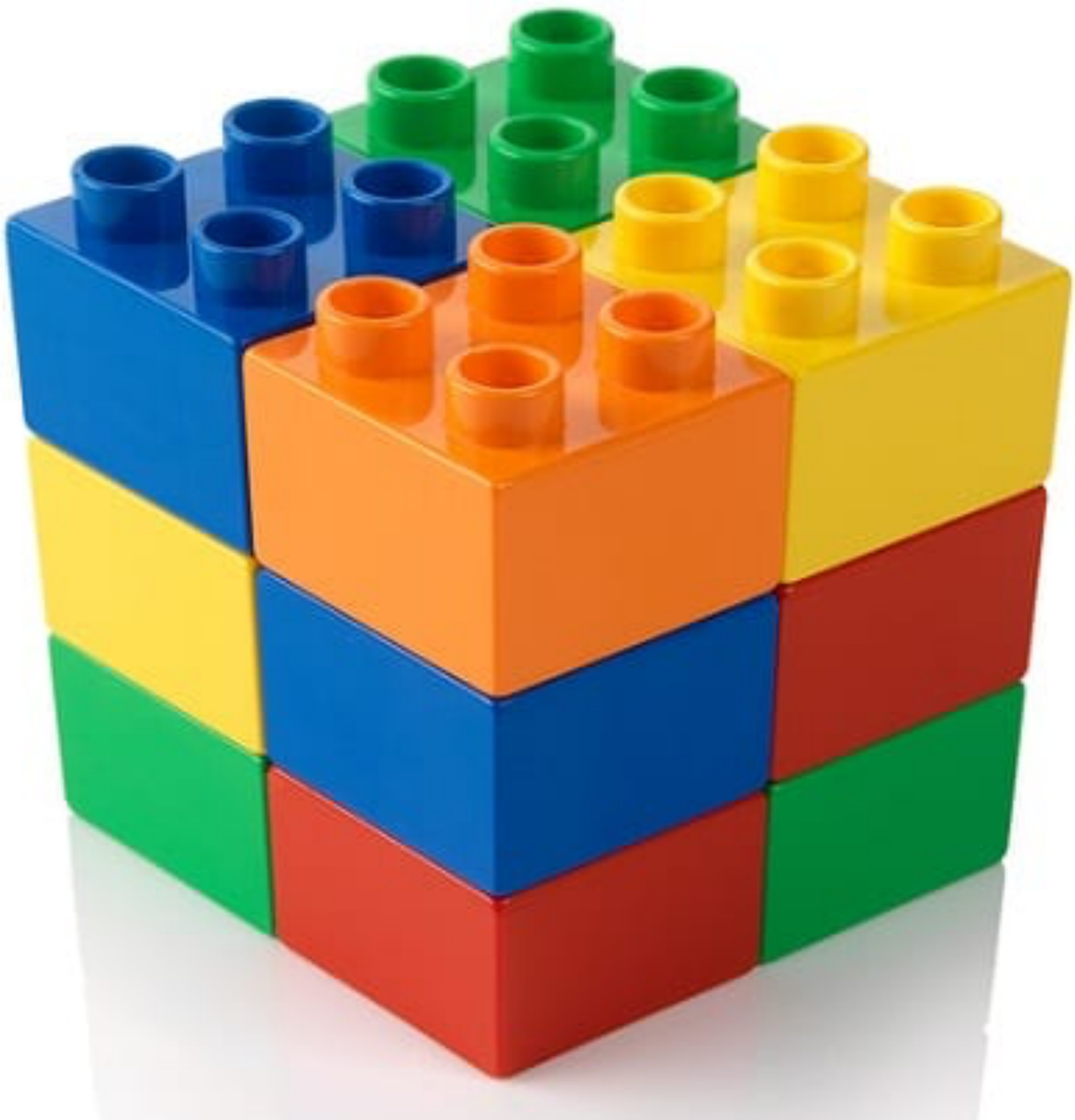
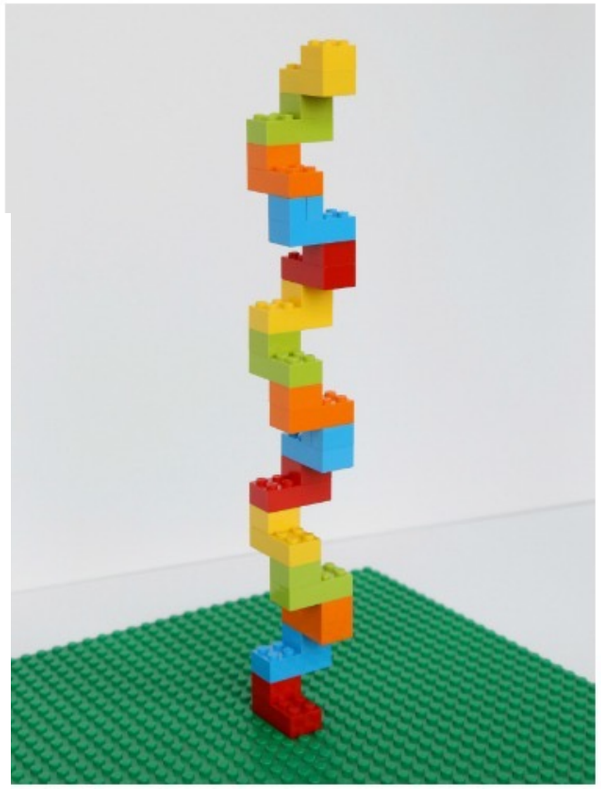
R Bootcamp:
Dataframe access & What is it?

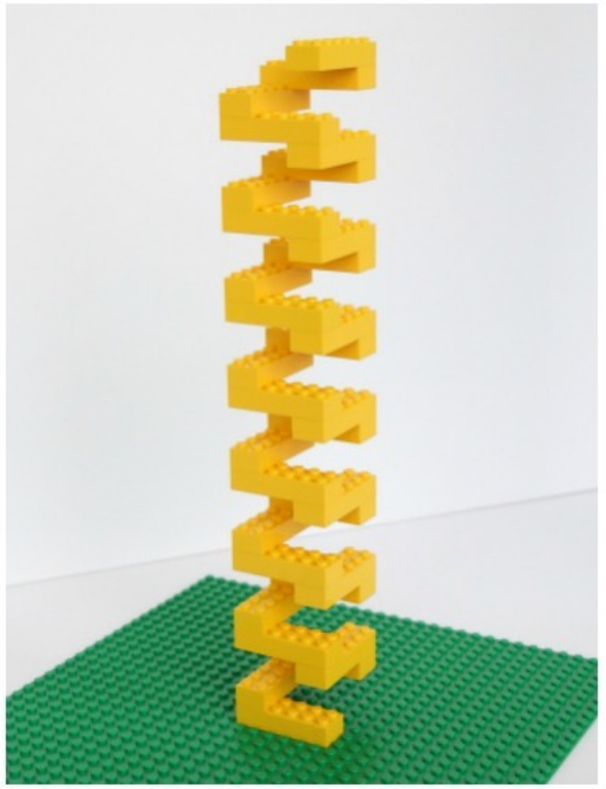
Marguerite Butler
mbutler808@gmail.com

Dept. of Biology
University of Hawaii

Tim Tam

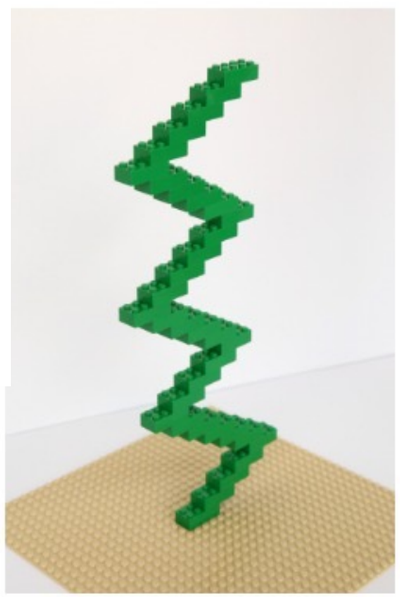


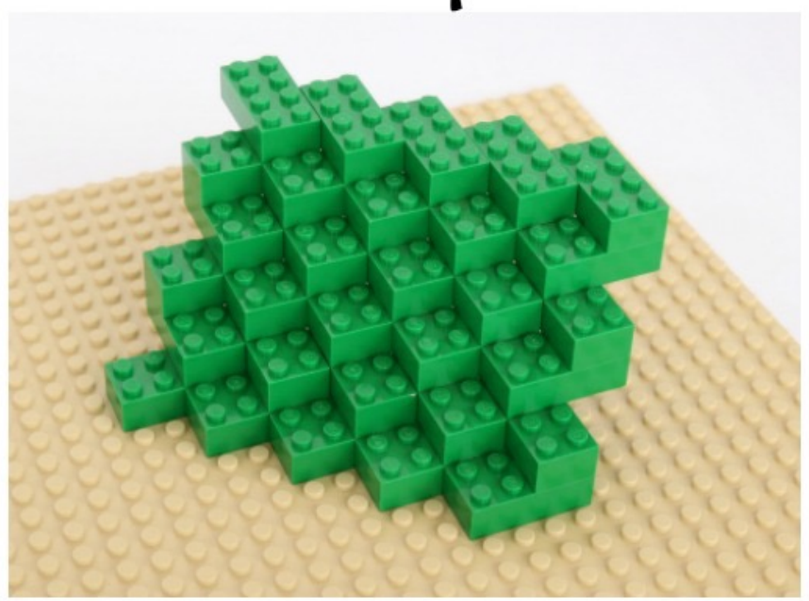




LEGO Math Patterns

Build & Write An Expression





R object types

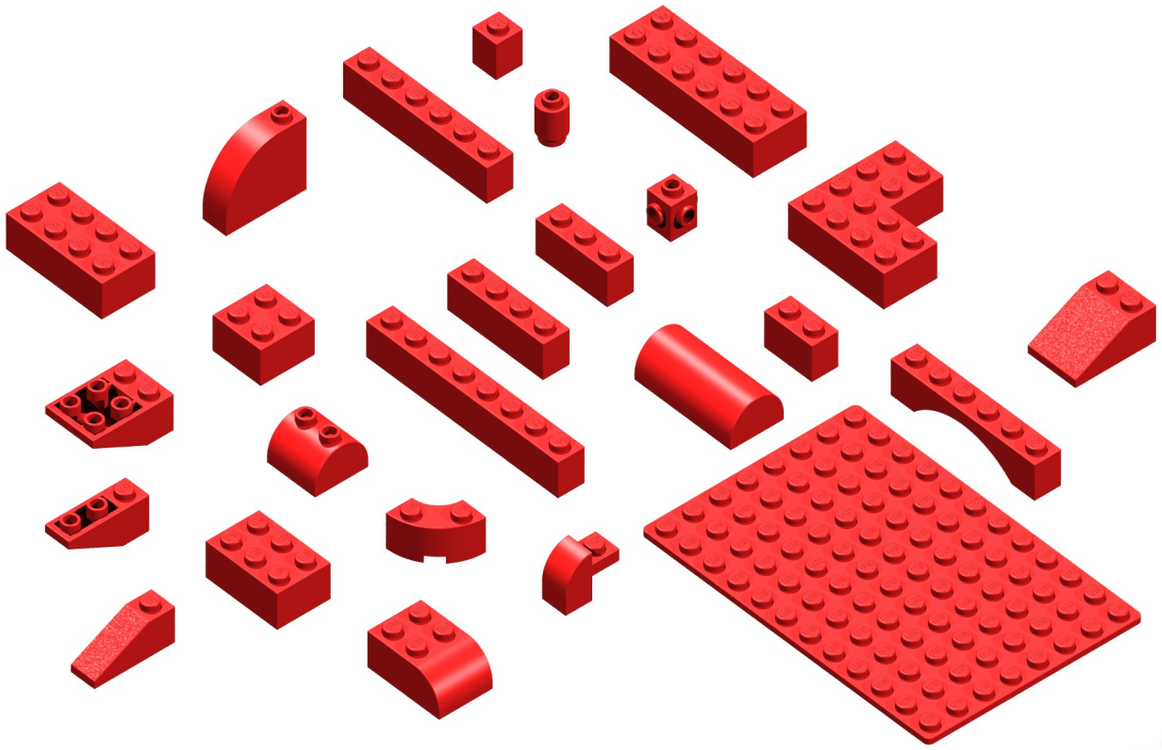
"**Atomic**" basic types for the single element. (can't break it down any further).

numeric
character
logical

All objects have **class, mode** and **length** (the number of vector elements; everything is a vector in R)

names are optional

Derived : combinations of atomic types or with special attributes



R object types

"**Atomic**" basic types for the single element. (can't break it down any further).

numeric
character
logical

All objects have **class, mode** and **length** (the number of vector elements; everything is a vector in R)

names are optional

Derived : combinations of atomic types or with special attributes

factor
dataframe
custom-programmed

Derived objects can have **attributes** such as:

dim (dimensions -- number of rows, columns)

row.names

(other programmer-defined attributes)

R object types

Vector: a one-dimensional array of arbitrary **length**.

Matrix: a two-dimensional array with an arbitrary **number of rows and columns**.

Array: as a matrix, but of **arbitrary dimension** (i.e., more than 2).

Data frame: a set of data organized similarly to a matrix. However each column of the data frame may contain its own type of data. Columns typically correspond to variables in a statistical study, while rows correspond to observations of these variables.

Function: a set of commands that are packaged into a unit with defined input and output (I/O is not necessary, though).

List: an arbitrary collection of other R objects (which may include other lists).

```
> x <- cbind(a=1:3, pi=pi) # simple
                           # matrix w/ dimnames
> x
     a       pi
[1,] 1 3.141593
[2,] 2 3.141593
[3,] 3 3.141593


> class(x)
[1] "matrix"

> attributes(x)
$dim
[1] 3 2


$dimnames
$dimnames[[1]]
NULL

$dimnames[[2]]
[1] "a"  "pi"

> attributes(x) <- NULL
> x # now just a vector of length 6

[1] 1.000000 2.000000 3.000000 3.141593
    3.141593 3.141593

> class(x)      # vector is default mode
[1] "numeric"
```

How does R store data?

R can **save** information in variables or objects

Assignment works by two types of operators:

Equal sign: right side stored in left side

```
> x = 6        (put 6 into x)
```

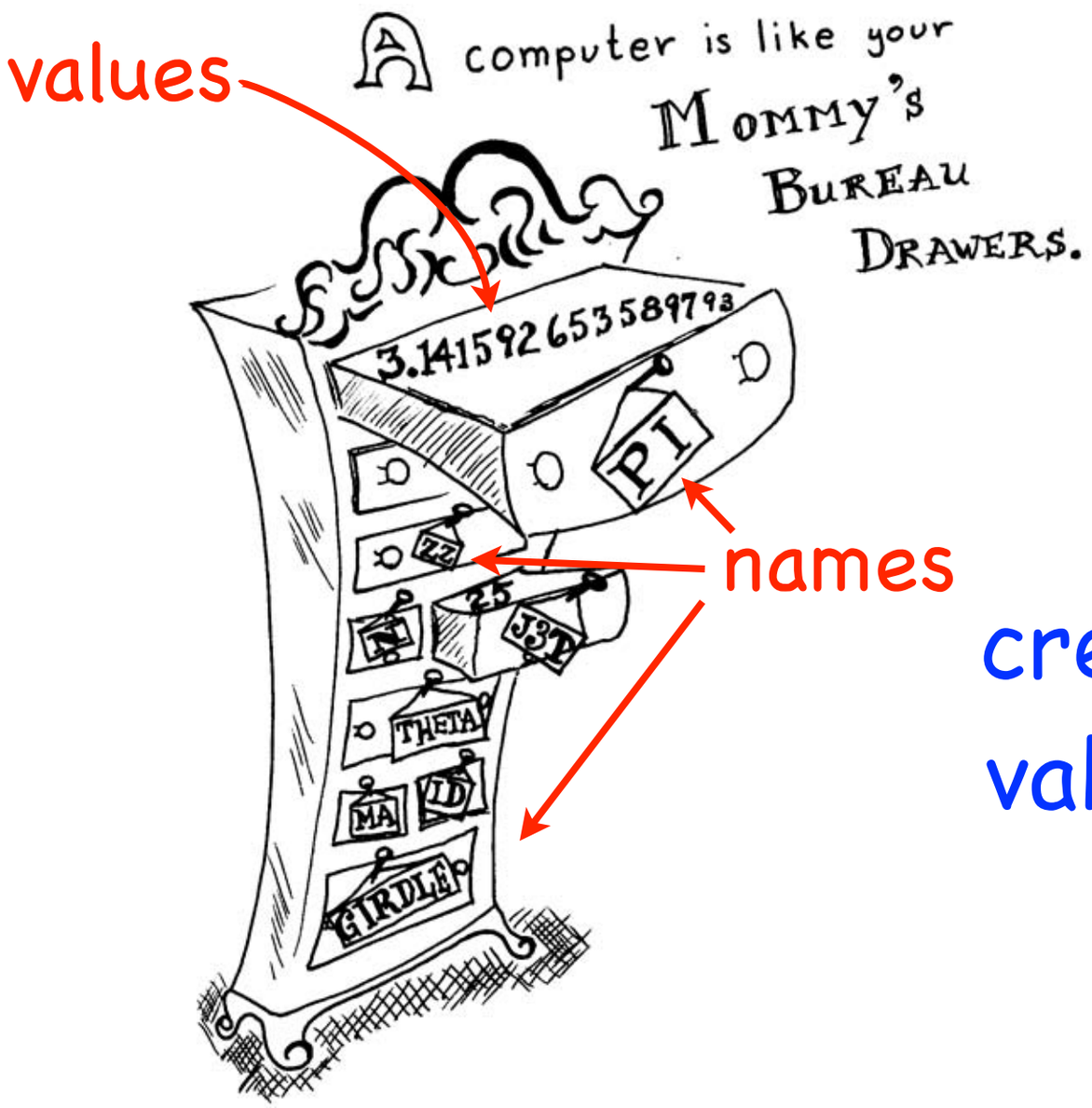
Arrow: assignment direction follows arrow

```
> x <- 6       (put 6 into x)

> 6 -> x       (put 6 into x)

> 6 = x        (error! cannot put x into 6)
```

How does R store data?

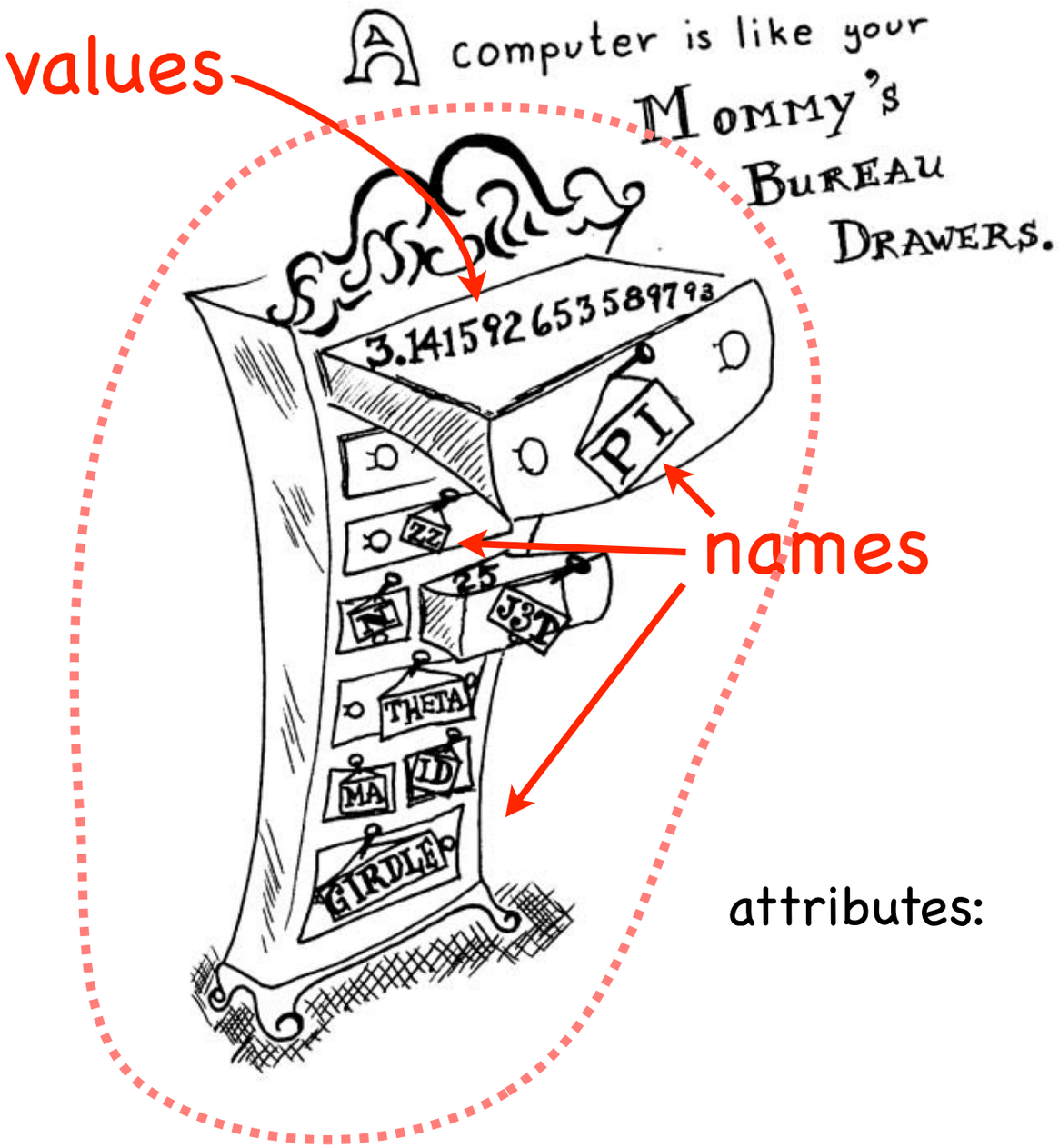


In R, there are drawers for:
- Numbers
- Characters (alphabetical strings)
- Logical (TRUE or FALSE)
- Complex Numbers (don't worry)

Create a new drawer by creating a name, and shoving a value into it - R will assign the "type" or "mode"

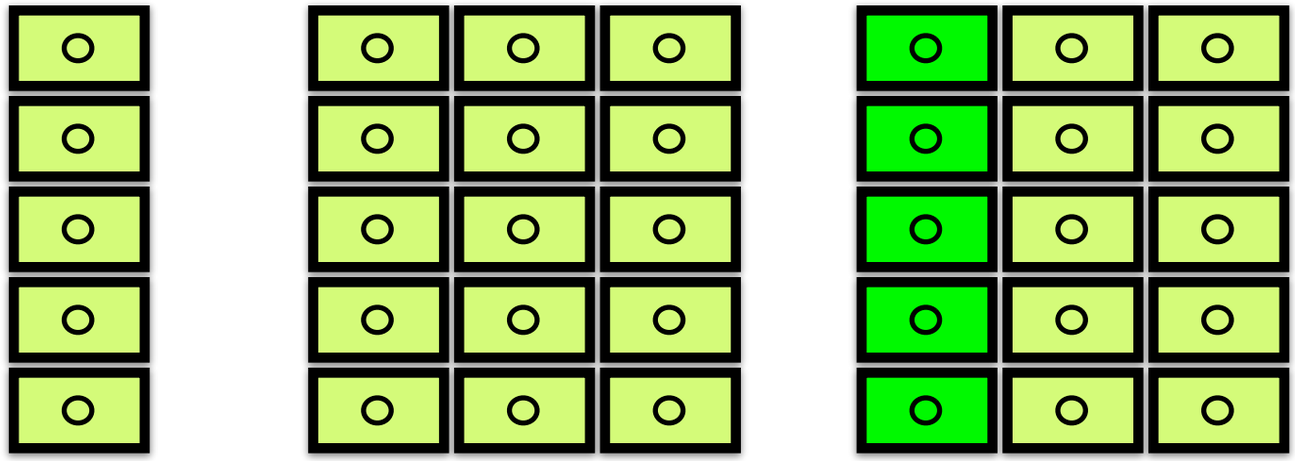
```
> x <- 25
 name    value
> mode(x)
[1] "numeric"
```

How does R store data?



Object

Bureaus can come in different shapes

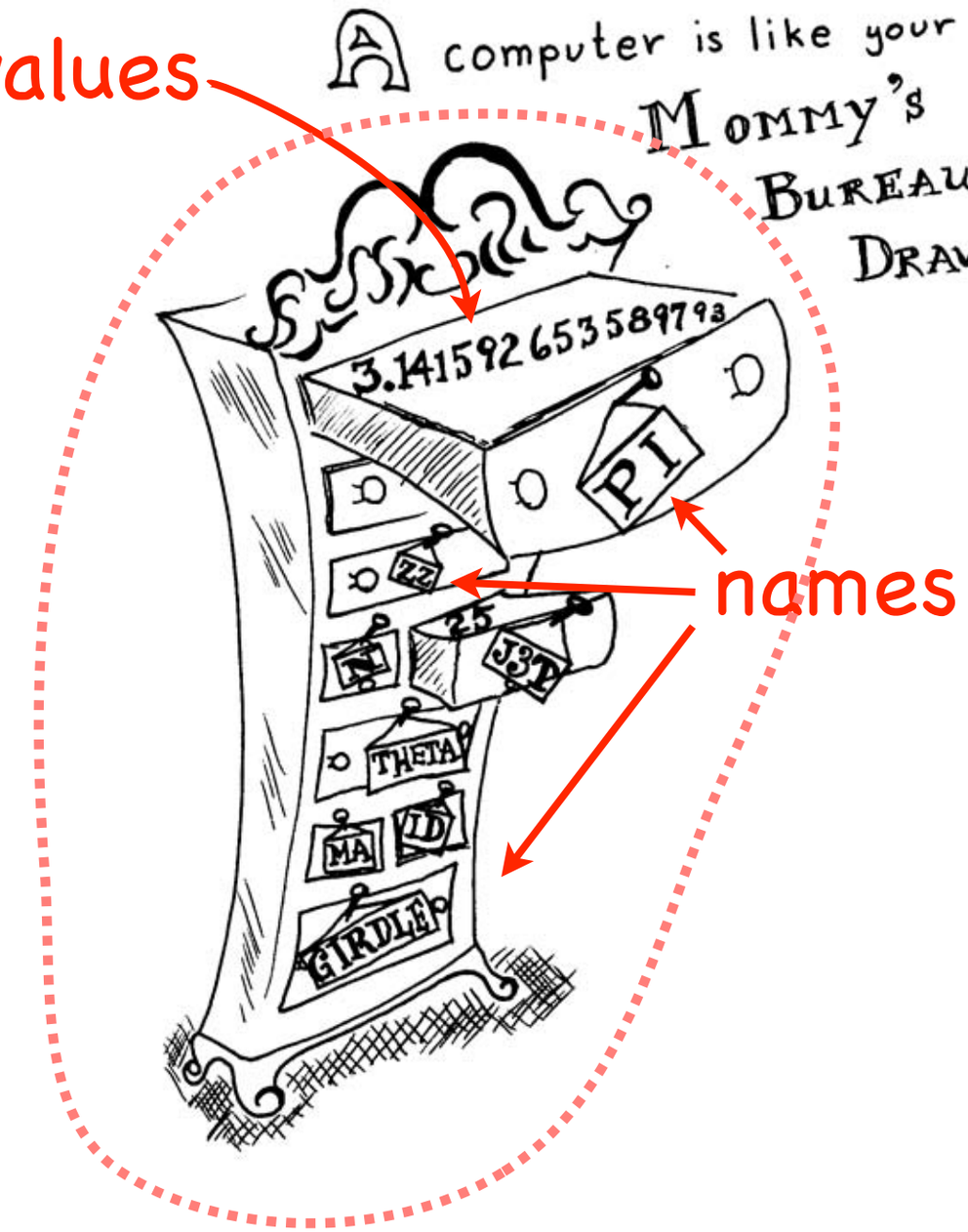


attributes: length dimensions(5 rows, 3 columns)

Rectangular!
all columns have same length

How does R store data?

values



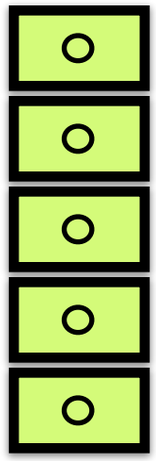
names

Object

Bureaus can come in different shapes

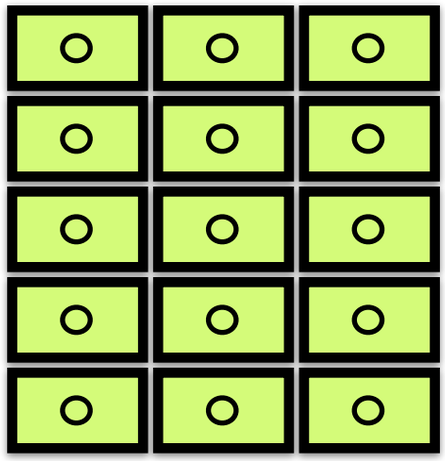
" record format"

Vector

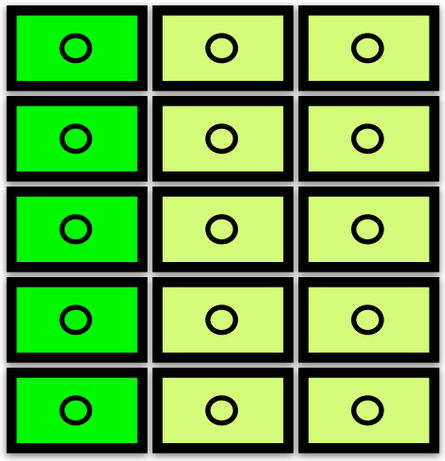


All values same type
numeric vector
or
character matrix
etc.

Matrix



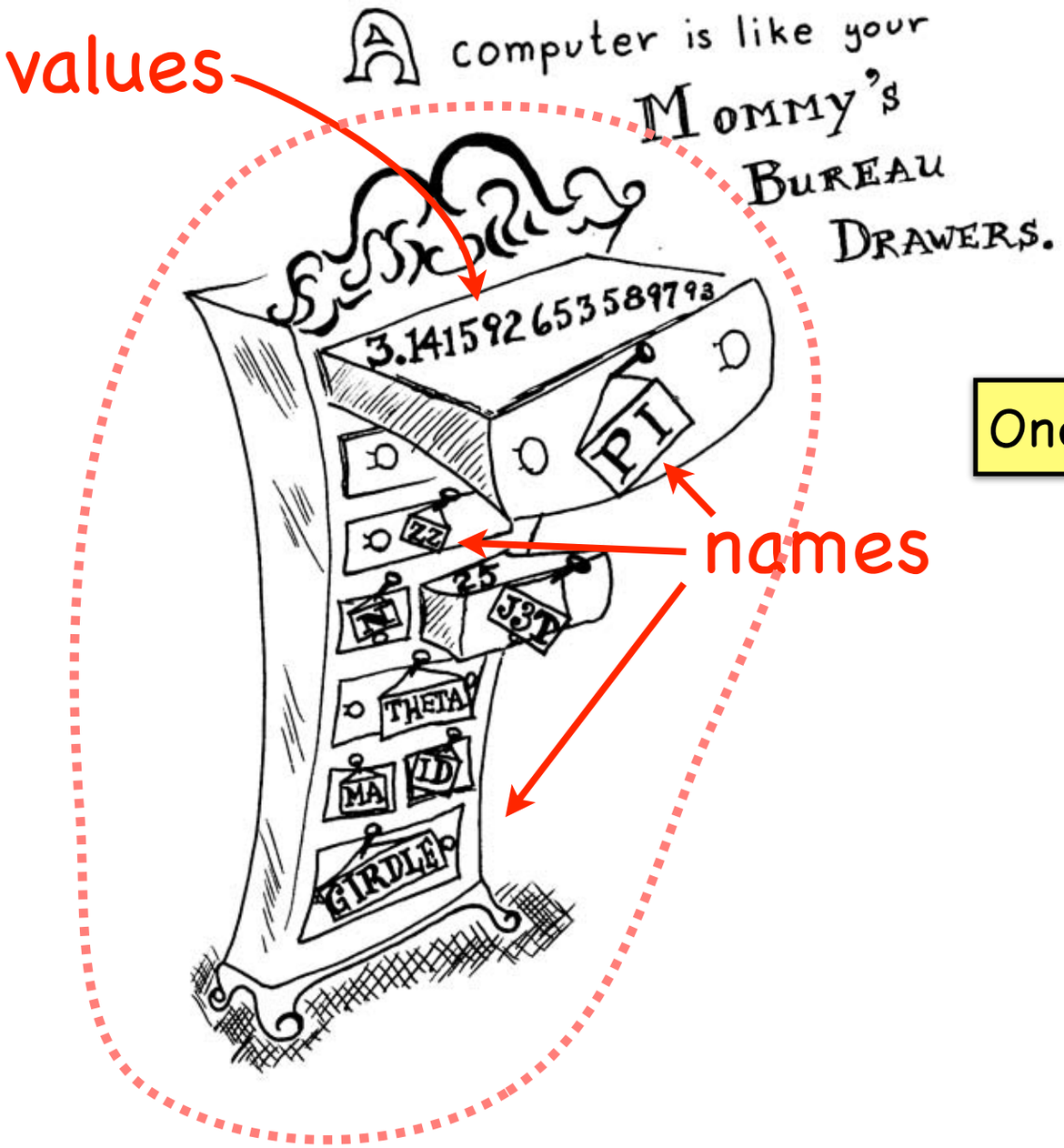
Data Frame



character
e.g. species

numeric
size, mass

How does R store data?

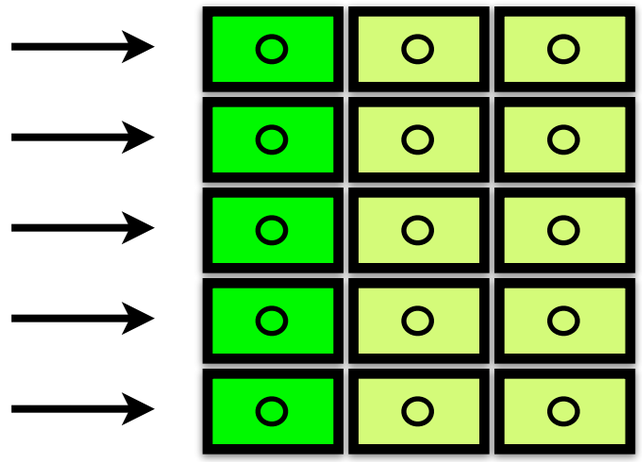


Object

Bureaus can come in different shapes

"record format"
Data Frame

One row = one observation



For example:

species name	size	mass
character vector	numeric vector	numeric vector

values in each vector same type,
but vectors can be different types

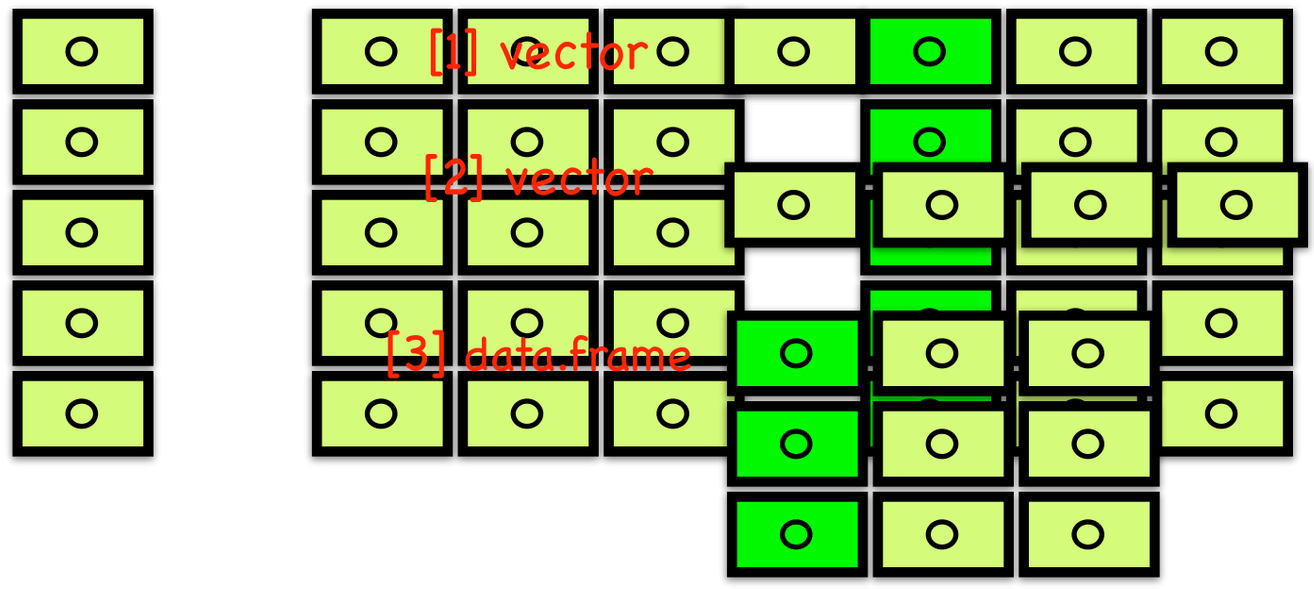
How does R store data?

But R also uses addresses in memory

classes of objects

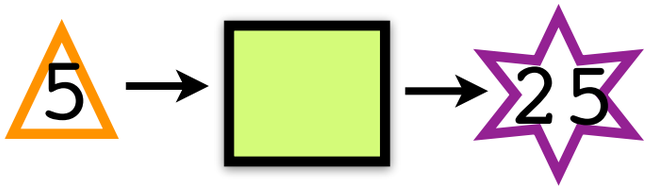
" record format"

Vector Matrix List Data Frame



a "list" of objects
- offers more flexibility
- often used for model output
- R has many functions that operate on lists

Functions



input output

Common Sources of Error

1) Typos! Computers are very anal that way.

```
> length = 6 # is not the same as
> lengths = 6
```

2) R is case sensitive

```
> length != Length
```

3) Using ( ) when should use [ ] and vice versa

```
> mean(x) # use () for functions
> mean[x] # error
> x[5]    # select an element of a vector, matrix, data.frame, etc.
> x(5)    # error
```

4) No comma or comma in the wrong place

```
> x[5,3]  # fifth row, third column of x
> x[5 3]  # error
> x[5,3,] # error
```

Common Sources of Error

5) Forgetting quotes for character strings (R will assume it's another named object or variable)

```
> treatment = c("a", "b", "c")
> treatment == a    # error - R thinks a is another object
```

What is it?

mode : the atomic data type (**class** = **mode** if object is atomic)

numeric
character
logical

class : classes can be derived or atomic -- (and which methods are applied to the object are determined by its class)

factor
dataframe
custom-programmed

When working with a new package,
you want to know what kind of objects you are dealing with:

A common source of error is trying to input
the **wrong class** of object
into a function

Accessing parts of your dataframe

How matters, because of Inheritance

by index: mydat[row #, column #]

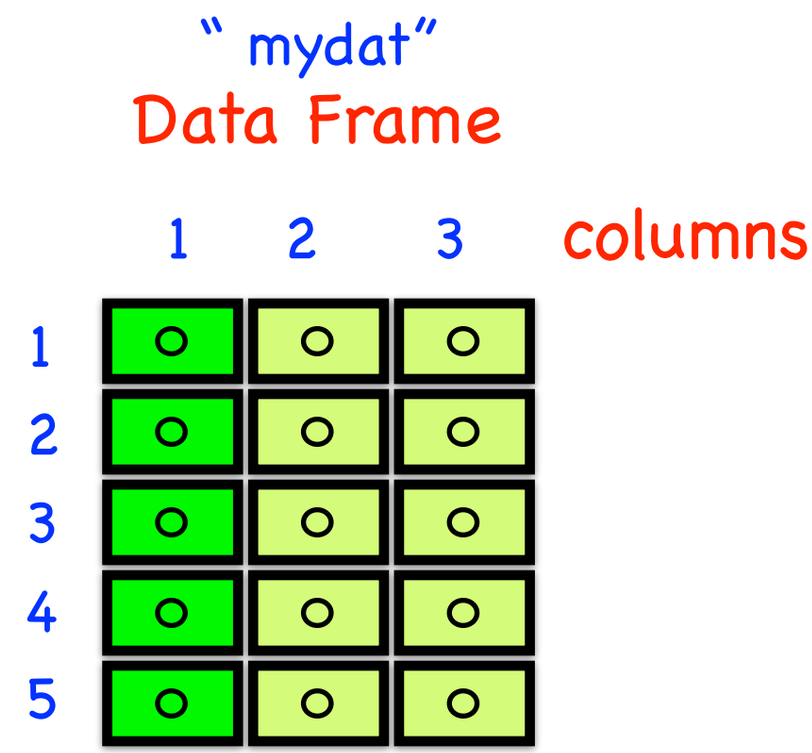
```
> mydat[2, 1]   # row 2, col 1
> mydat[2, ]    # entire row 2
> mydat[, 1]    # entire first col
```

by name:

```
> mydat[, “species”] # species col
> mydat[“species”]   # same
> mydat[2, “size”]   # row 2, col 2
```

using $:

```
> mydat$species # species col
> mydat$mass    # mass col
```



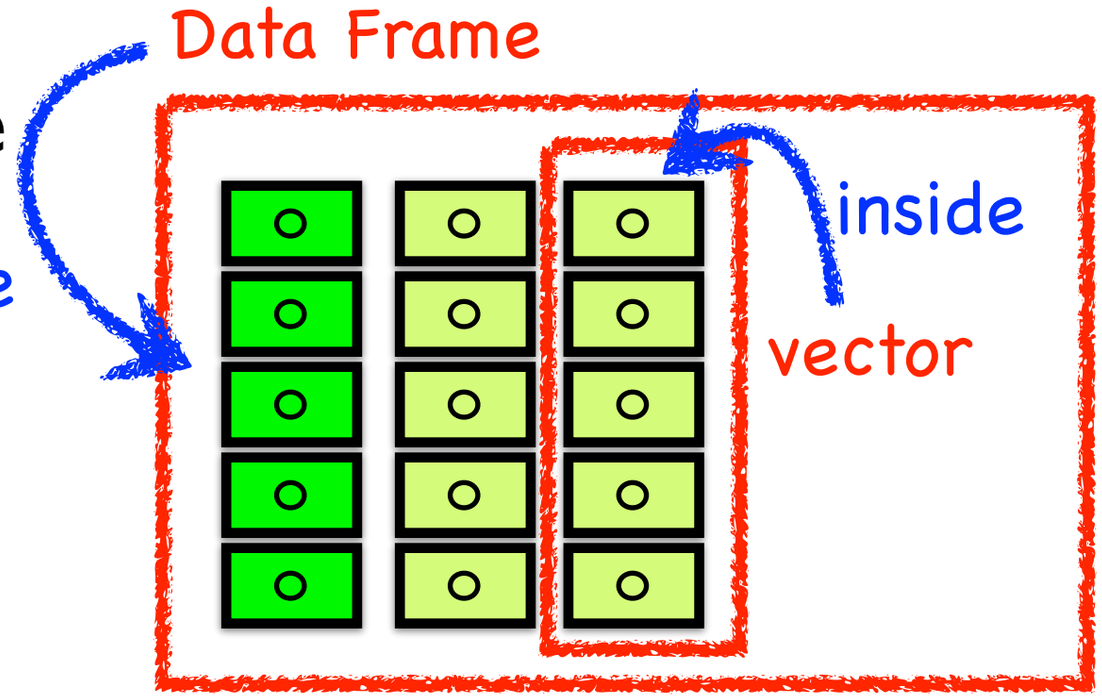
What is it?

The class of the new object depends on HOW you grab it
Classes are inherited

Psst: A dataframe is a list of vectors

using the [ ] method gets a **subset**
of the dataframe = smaller dataframe

```
> mydat[2, 1]
> mydat[, “species”]
```



using the $ method grabs a **vector**
from inside the dataframe = vector

```
> mydat$species # species col
```

[[ ]] method: grabs the first level
element “inside” the dataframe = vector

```
> mydat[[1]] # first column
```