Lists

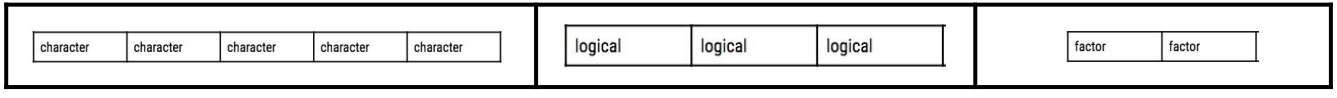
R

Vector **slot**/vector **element** = vector of length 1 of same class

Simple vector

character	character	character	character	character
integer	integer	integer	integer	integer
numeric	numeric	numeric	numeric	numeric
logical	logical	logical	logical	logical
factor	factor	factor	factor	factor

List



List **slot**/list **element** = arbitrary R object

Comparing lists to simple vectors

Creating lists

The `list` function: `list(obj_1, obj_2, ..., obj_n)`

```
> responses_student1 <- list(c(4,20,3), c("bear",
"giraffe"), c("red", "orange", "yellow", "green", "blue",
"purple"))
> responses_student1
[[1]] 1st list slot/element (unnamed)
[1]  4 20  3

[[2]] 2nd list slot/element (unnamed)
[1] "bear"    "giraffe"

[[3]] 3rd list slot/element (unnamed)
[1] "red"    "orange" "yellow" "green"  "blue"   "purple"
```

Adding names to a list (method 1)

Use the `names` function:

```
> names(responses_student1) <- c("numbers", "animals",
"colors")
> responses_student1
$numbers
[1]  4 20  3

$animals
[1] "bear"    "giraffe"

$colors
[1] "red"    "orange" "yellow" "green"  "blue"   "purple"
```

1st list slot/element (named "numbers")

2nd list slot/element (named "animals")

3rd list slot/element (named "colors")

Adding names to a list (method 2)

Using named arguments in the `list` function:

Named arguments

```
> responses_student2 <- list(numbers = 1:5, animals =
c("T-rex", "tiger", "lion"), colors = c("red", "green"))
> responses_student2
$numbers
[1] 1 2 3 4 5

$animals
[1] "T-rex" "tiger" "lion"

$colors
[1] "red"   "green"
```

1st list slot/element (named "numbers")

2nd list slot/element (named "animals")

3rd list slot/element (named "colors")

List of lists

```
> responses_all_students <- list(responses_student1, responses_student2)
> responses_all_students
[[1]]
[[1]]$numbers
[1]  4 20  3

[[1]]$animals
[1] "bear"    "giraffe"

[[1]]$colors
[1] "red"    "orange" "yellow" "green"  "blue"   "purple"


[[2]]
[[2]]$numbers
[1] 1 2 3 4 5

[[2]]$animals
[1] "T-rex" "tiger" "lion"

[[2]]$colors
[1] "red"   "green"
```

List of lists (with names)

```
> list(st1 = responses_student1, st2 = responses_student2)
$st1
$st1$numbers
[1]  4 20  3

$st1$animals
[1] "bear"    "giraffe"

$st1$colors
[1] "red"    "orange" "yellow" "green"  "blue"   "purple"


$st2
$st2$numbers
[1] 1 2 3 4 5

$st2$animals
[1] "T-rex" "tiger" "lion"

$st2$colors
[1] "red"   "green"
```

1st list slot/element (named "st1")

2nd list slot/element (named "st2")

Relationship between lists and data frames

All data frames are lists. Data frames are lists where each list element is a simple vector of the same length.

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

These columns form the individual list elements (all length-6 simple vectors).

Relationship between lists and data frames (as.list)

```
> as.list(head(mtcars))
$mpg
[1] 21.0 21.0 22.8 21.4 18.7 18.1
$cyl
[1] 6 6 4 6 8 6
$disp
[1] 160 160 108 258 360 225
$hp
[1] 110 110  93 110 175 105
$drat
[1] 3.90 3.90 3.85 3.08 3.15 2.76
$wt
[1] 2.620 2.875 2.320 3.215 3.440 3.460
$qsec
[1] 16.46 17.02 18.61 19.44 17.02 20.22
$vs
[1] 0 0 1 1 0 1
$am
[1] 1 1 1 0 0 0
$gear
[1] 4 4 4 3 3 3
$carb
[1] 4 4 1 1 2 1
```

Each list element is named with the corresponding column name of the data frame.

List subsetting

How can we extract certain parts of a list?

- Double square brackets: `[[`
- Dollar sign notation: `$`
- Single square brackets: `[`

Subsetting: double square brackets

Within the brackets, specify an integer index or a character string.

For lists:

Named list

```
> l <- list(a = 1:7, b = c("foo", "bar", "biz"))
> res1 <- l[[2]]
> res2 <- l[["b"]]
> res1
[1] "foo" "bar" "biz"
> res2
[1] "foo" "bar" "biz"
> class(l)
[1] "list"
> class(res1)
[1] "character"
> class(res2)
[1] "character"
```

Integer index

Character string (requires list elements to be named)

The extracted objects are **character vectors**, not lists.

Subsetting: double square brackets

Within the brackets, specify an integer index or a character string.

For data frames:

```
> iris_subset <- head(iris, 3)
> iris_subset
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
> iris_subset[[2]]
[1] 3.5 3.0 3.2
> iris_subset[["Sepal.Width"]]
[1] 3.5 3.0 3.2
```

Data frame with 3 rows, 5 columns

Integer index

Character string

The extracted objects are **numeric vectors**, not lists.

Subsetting: dollar sign notation

After the `$`, specify a name in the list (no quotes).

For lists:

```
> l <- list(a = 1:7, b = c("foo", "bar", "biz"))
> l$b
[1] "foo" "bar" "biz"
```

No quotes around b

The extracted object is a **character vector**, not a list.

Subsetting: dollar sign notation

After the `$`, specify a name in the list (no quotes).

For data frames:

```
> iris_subset
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
> iris_subset$Sepal.Width
[1] 3.5 3.0 3.2
```

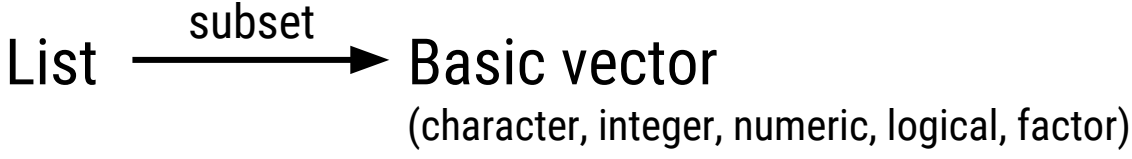
No quotes around Sepal.Width

The extracted object is a **numeric vector**, not a list.

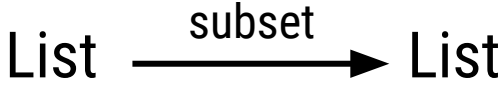
Subsetting: single square brackets

The previous forms of subsetting ([[and $) **simplify** the output.

- Original object is a more complex object (list) than the extracted objects (basic vectors).



Single square bracket subsetting **preserves** the class of the output.



Subsetting: single square brackets

Just like `[[`, specify an integer index or a character string.

For lists:

```
> l <- list(a = 1:7, b = c("foo", "bar", "biz"))
> l[2]
$b
[1] "foo" "bar" "biz"

> l["b"]
$b
[1] "foo" "bar" "biz"

> class(l[2])
[1] "list"
> class(l["b"])
[1] "list"
```

Seeing this indicates that the extracted output is a list

Verifying that the output is a list

Subsetting: single square brackets

Just like `[[`, specify an integer index or a character string.

For data frames:

```
> iris_subset[2]
  Sepal.Width
1         3.5
2         3.0
3         3.2
> iris_subset["Sepal.Width"]
  Sepal.Width
1         3.5
2         3.0
3         3.2
> class(iris_subset[2])
[1] "data.frame"
> class(iris_subset["Sepal.Width"])
[1] "data.frame"
```

Extracted output is displayed as a column which indicates that it is a data frame

Verifying that the output is a data frame

Summary

- Lists are a flexible way to store complex data.
  - Can be created from scratch with the `list` function
- Subsetting lists
  - Double brackets
    - Specify integer index or character string (in quotes)
    - Class-simplifying operation
  - Dollar sign
    - Specify name (no quotes)
    - Class-simplifying operation
  - Single bracket
    - Specify integer index or character string (in quotes)
    - Class-preserving operation