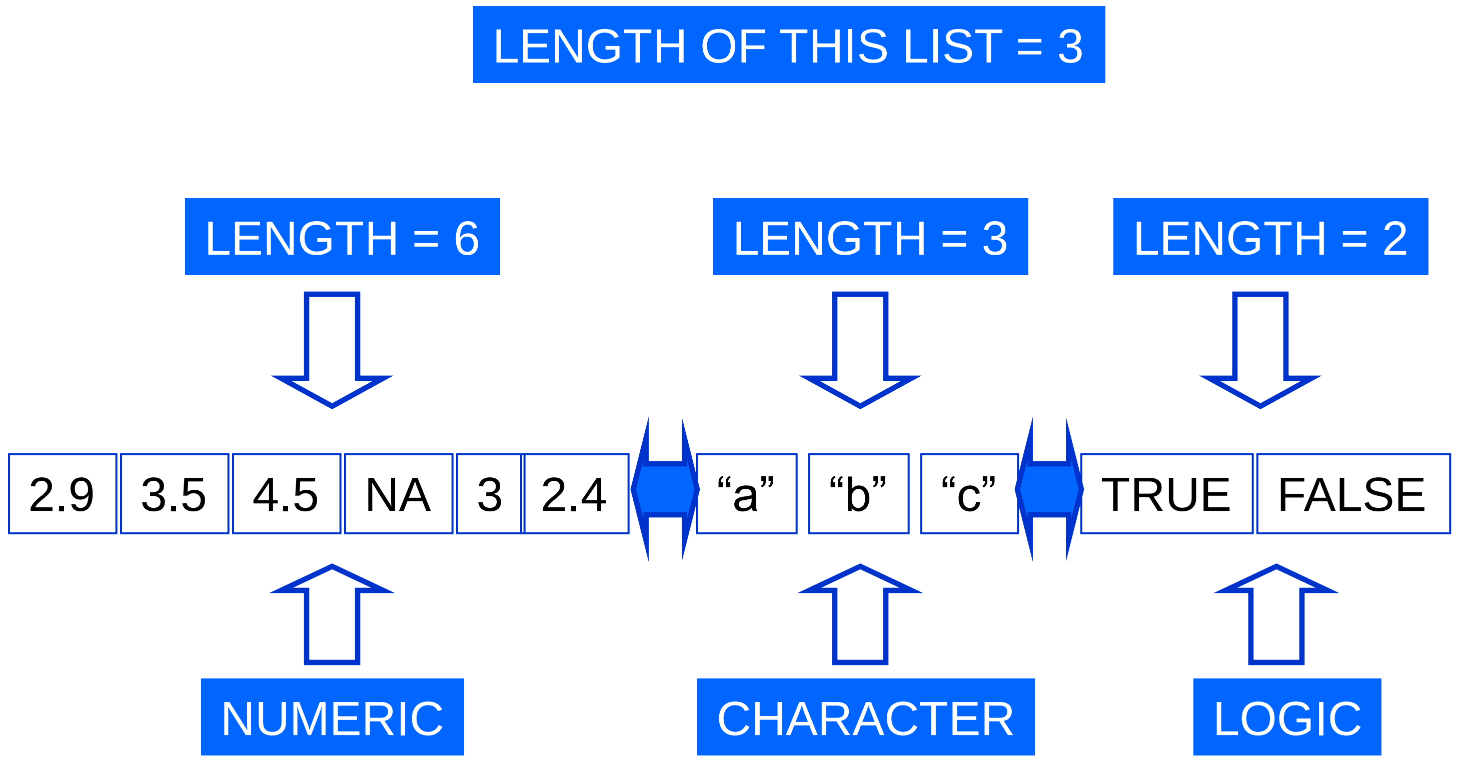
Chapter 3
Lists and Data Frames

Arthur Li

Lists

❖The most often encountered/useful object in R is the list



Creating a List

❖ Create a list – using the **`list`** function

```
> student <- list(name = "John", year = 2, classTaken =
+ c("PM510", "PM511A", "PM511B"), GPA = 3.85)
> student
$name
[1] "John"

$year
[1] 2

$classTaken
[1] "PM510"  "PM511A" "PM511B"

$GPA
[1] 3.85
```

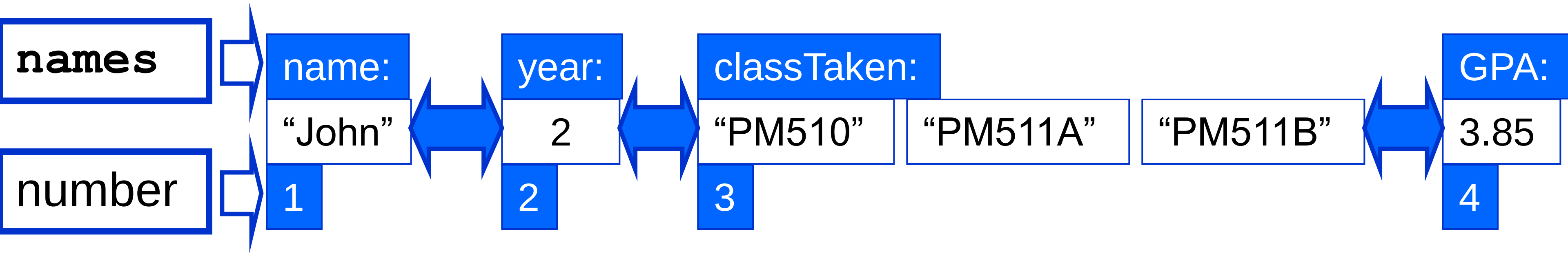
Creating a List

❖ Use **`str(object)`** to display the internal structure

```
> str(student)
List of 4
 $ name      : chr "John"
 $ year      : num 2
 $ classTaken: chr [1:3] "PM510" "PM511A" "PM511B"
 $ GPA       : num 3.85
```

Accessing the Components of a List and the names Attribute

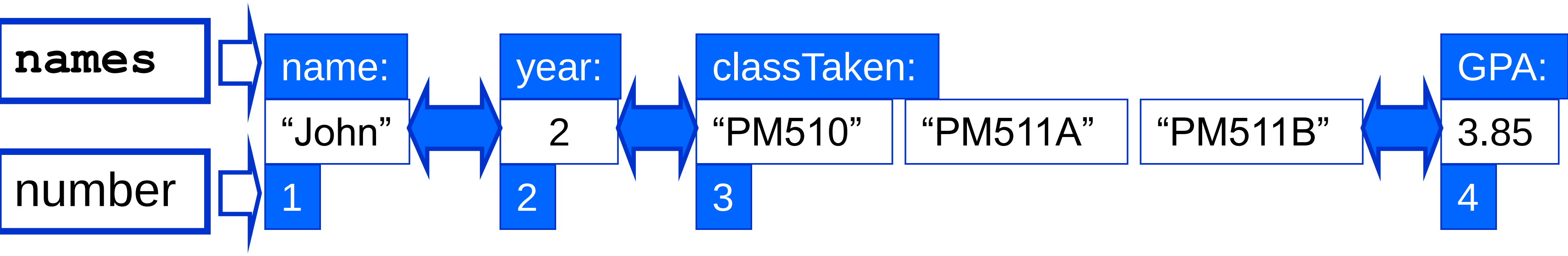
❖ The components of a list are always numbered



```
> length(student)
[1] 4
> student[[1]]
[1] "John"
> student[[3]]
[1] "PM510"  "PM511A" "PM511B"
> names(student)
[1] "name"        "year"        "classTaken" "GPA"
> student$GPA
[1] 3.85
```

Accessing the Components of a List and the names Attribute

❖ The components of a list are always numbered

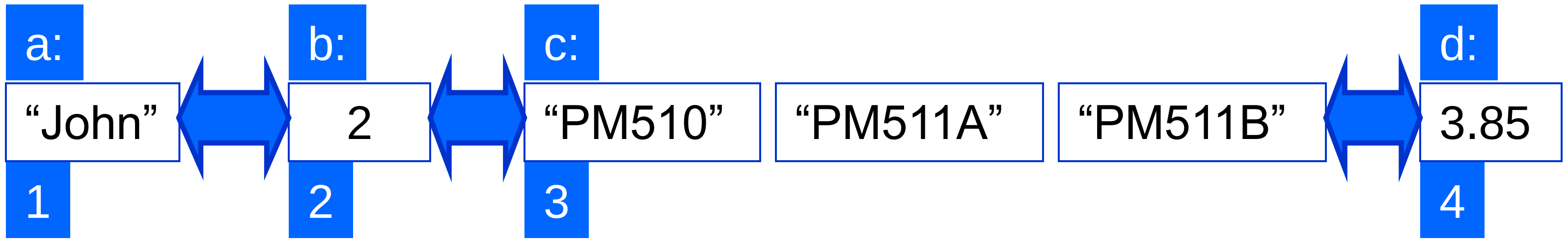


```
> student[[3]][1]
[1] "PM510"
> student$classTaken[1]
[1] "PM510"
```

Accessing the Components of a List and the names Attribute

❖ We can change the **`names`** of the list

```
> names(student) = letters[1:4]
```



The Difference Between [ ] and [[ ]]

- ❖ Using `[]` returns a list with the selected components
- ❖ The result is still a list

```
> student[2]
$b
[1] 2

> student[3:4]
$c
[1] "PM510"  "PM511A" "PM511B"

$d
[1] 3.85
```

The Difference Between [ ] and [[ ]]

- Using `[[]]` extracts or replaces the components of a list
- The result is a vector

```
> student[[3]]
[1] "PM510"   "PM511A" "PM511B"
```

- `student[[3:4]]` is not allowed!

Concatenating Lists

❖ You can use the **`c`** function to add components to a list

```
> student = c(student, age = 25)
> student
$a
[1] "John"

$b
[1] 2

$c
[1] "PM510"  "PM511A" "PM511B"

$d
[1] 3.85

$age
[1] 25
```

Concatenating Lists

❖ The c function has a **`recursive`** argument

❖ Setting **`recursive = TRUE`** will unlist the arguments first before joining them

```
> list2 = c(list(x=letters[1:3], y=2:4), list(z=c(1, 2.0, 3.5)),
+ recursive=TRUE)
> list2
   x1     x2     x3     y1     y2     y3     z1     z2     z3 
  "a"    "b"    "c"    "2"    "3"    "4"    "1"    "2"  "3.5" 
> mode(list2)
[1] "character"
```

❖ The numeric values → characters

The `unlist` Function

❖ The **`unlist`** function converts a list to a vector

```
> unlist(student)
        a          b          c1          c2          c3           d        age 
  "John"        "2"   "PM510" "PM511A" "PM511B"    "3.85"       "25" 
> 
> unlist(student, use.names = F)
[1] "John"    "2"        "PM510"   "PM511A" "PM511B" "3.85"    "25"
```

Handling the NULL Value in Lists

❖ To remove the components of a list, we can do the following

```
> student1 = student2 = student3 = list(name = "John", year = 2,    +
classTaken = c("PM510", "PM511A", "PM511B"), GPA = 3.85)
>
> student1["year"] = NULL
> student1
$name
[1] "John"

$classTaken
[1] "PM510"  "PM511A" "PM511B"

$GPA
[1] 3.85
```

Handling the NULL Value in Lists

❖ To remove the components of a list, we can do the following

```
> student2[["year"]] = NULL
> student2
$name
[1] "John"

$classTaken
[1] "PM510"  "PM511A" "PM511B"

$GPA
[1] 3.85
```

❖ Instead of using **`names`**, we can also use the number

```
> student1[2] = NULL
> student2[[2]] = NULL
```

Handling the NULL Value in Lists

❖ To set the **`year`** components to **`NULL`** …

```
> student3["year"] = list(NULL)
> student3
$name
[1] "John"

$year
NULL

$classTaken
[1] "PM510"  "PM511A" "PM511B"

$GPA
[1] 3.85
```

Data Frames
Creating a Data Frame from Existing Vectors

❖Data frame = data set

❖A data frame = special case of a list; length of each components are the same

```
> sex = c("M", "F", "F", "M", "M")
> height = c(65, 63, 60, 62, 57)
> weight = c(150, 140, 135, 165, 175)
> live.on.campus = c(TRUE, TRUE, FALSE, FALSE, FALSE)
> d = data.frame(sex, height, weight, live.on.campus)
> d
  sex height weight live.on.campus
1   M     65    150           TRUE
2   F     63    140           TRUE
3   F     60    135          FALSE
4   M     62    165          FALSE
5   M     57    175          FALSE
```

❖All the character columns → factors, unless using **`I()`**

```
> d1 = data.frame(I(sex), height, weight, live.on.campus)
```

The `rownames` and `colnames` of the Data Frame

❖ To find **`colnames`** (variable names) or **`rownames`**…

```
> colnames(d)
[1] "sex"            "height"          "weight"
"live.on.campus"
> rownames(d)
[1] "1" "2" "3" "4" "5"
> names(d)
[1] "sex"            "height"          "weight"
"live.on.campus"
```

❖ To assign a meaning **`rownames`**…

```
> id = c(2345, 1236, 2986, 6543, 6544)
> rownames(d) = id
> d
     sex height weight live.on.campus
2345   M     65    150           TRUE
1236   F     63    140           TRUE
2986   F     60    135          FALSE
6543   M     62    165          FALSE
6544   M     57    175          FALSE
```

❖Matrices can have **`rownames`** and **`colnames`**

Data Frame Index

- Data frame can be indexed in the same way as matrices

```
> d[1:3, c(1,3,4)]
     sex weight live.on.campus
2345   M    150           TRUE
1236   F    140           TRUE
2986   F    135          FALSE
```

Accessing a Variable

❖ Use $ to access a variable

```
> d$height[1:3]
[1] 65 63 60
>
> d$live.on.campus
[1]  TRUE  TRUE FALSE FALSE FALSE
>
> d$sex
[1] M F F M M
Levels: F M
>
> d1$sex
[1] "M" "F" "F" "M" "M"
```