Data Structures in R: Data Frames part 1

Stat 133 with Gaston Sanchez

Lists reminder

single data type   *multiple data types*

dimensions

1D

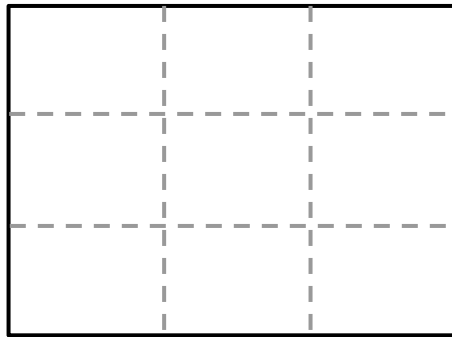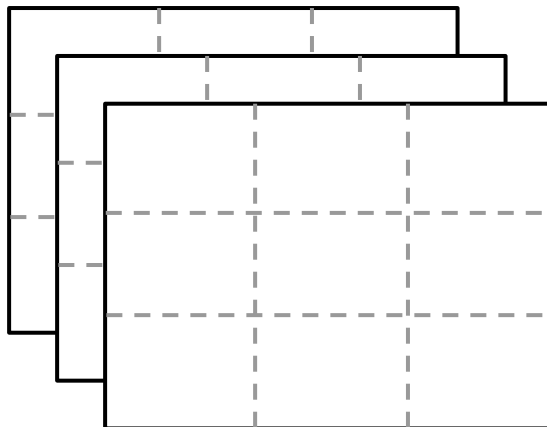### Vector

List

2D

Matrix

Data Frame
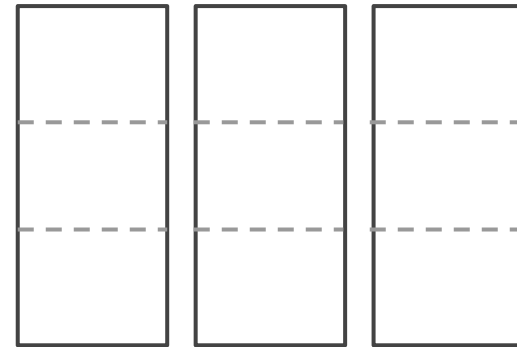
non-atomic
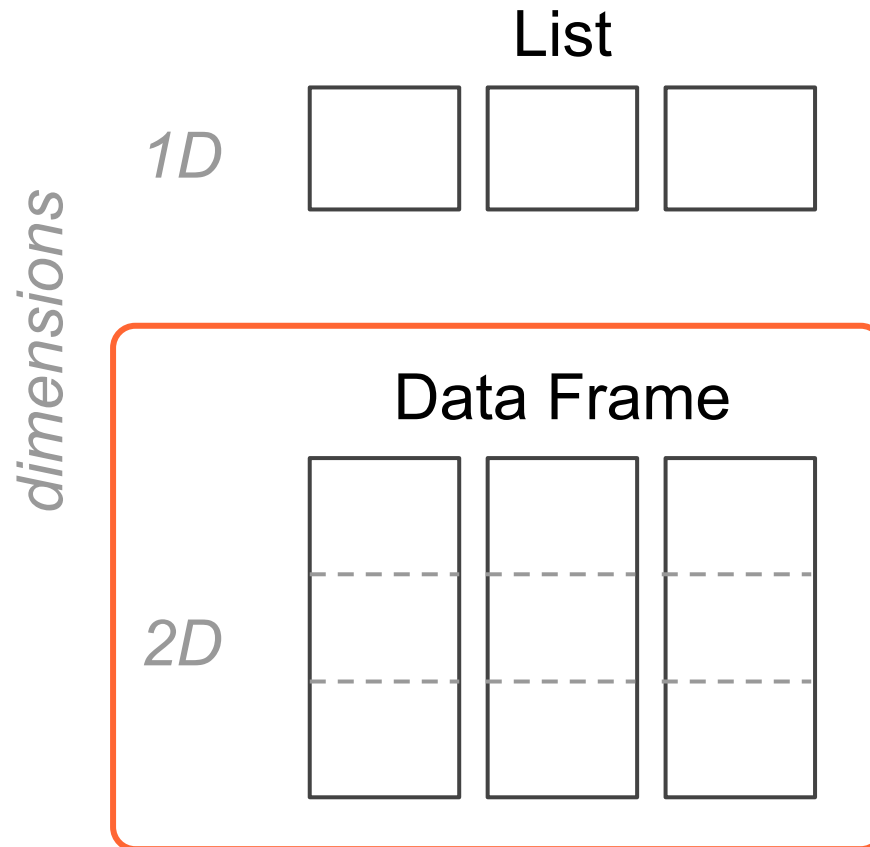structures

nD

Array

R lists

A list is the most general data structure in R

Lists can contain any other type of data structure

Lists can even contain other lists

Data Frames

multiple data types

List

dimensions

1D

Data Frame

2D

R data frames

A data.frame is the primary data structure that R provides for handling tabular data sets

Creating a data frame

```
# data frame

df <- data.frame(
  name = c('Anakin', 'Padme', 'Luke', 'Leia'),
  gender = c('male', 'female', 'male', 'female'),
  height = c(1.88, 1.65, 1.72, 1.50),
  weight = c(84, 45, 77, 49)
)
```

R data frames

R data frames are special kinds of lists

Stored in R as a list of vectors (or factors)

Columns are typically atomic structures

But since a data frame is a list, you can mix different types of columns

Data frames are NOT matrices but they behave a lot like matrices

Functions for Data Frames

There's a bunch of functions to inspect a data.frame object

Gaston Sanchez

Function	Description
`str()`	Structure
`head()`	First rows
`tail()`	Last rows
`summary()`	Descriptive statistics
`dim()`	Dimensions (# rows, # columns)
`nrow()`	Number of rows
`ncol()`	Number of columns
`names()`	Column names
`colnames()`	Column names
`rownames()`	Row names
`dimnames()`	List with row and column names

Functions to inspect a data frame

```
# display structure
str(airquality)


# display structure but showing
# few elements
str(airquality, vec.len = 1)
```

Gaston Sanchez

Functions to inspect a data frame

```r
# first n rows
head(airquality, n = 5)


# last n rows
tail(airquality, n = 5)
```

Functions to inspect a data frame

```
# column summaries
summary(airquality)


# memory size
object.size(airquality)


# attributes
attributes(airquality)
```

Functions to inspect a data frame

```r
# data frame dimensions
dim(airquality)


# number of rows
nrow(airquality)


# number of columns
ncol(airquality)
```

Functions to inspect a data frame

```
# row names
rownames(airquality)


# column names
colnames(airquality)


# column names
names(airquality)
```

Functions to inspect a data frame

```
# object class ('data.frame')
class(airquality)


# check if object is data.frame
is.data.frame(airquality)


# data.frame is also a list
is.list(airquality)
```

Basic manipulation of Data Frames

Working with data frames

There are many ways in which the elements of a data.frame can be accessed (i.e. retrieved, selected)

Accessing Rows
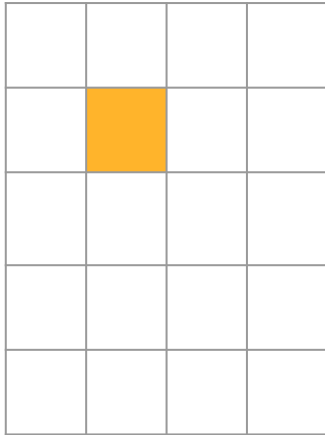

one single
row

consecutive
rows
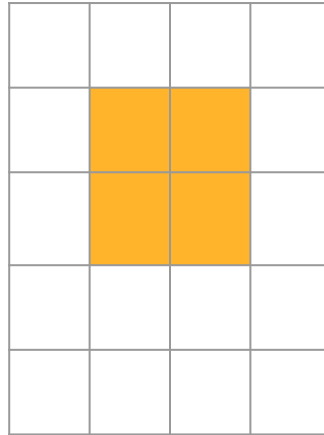
separate
rows

Accessing Columns

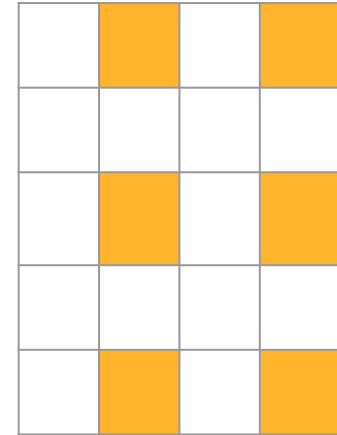one single
column

consecutive
columns

separate
columns

Accessing Cells

one single
cell

consecutive
cells

separate
cells

Data frame `airquality` *(first 10 rows)*

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	NA	NA	14.3	56	5	5
6	28	NA	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	NA	194	8.6	69	5	10

Retrieving elements via Index Values

Numeric Indices in a data frame

columns

	1	2	3	...	p-1	p		p = ncol(df)
1								
2								
:								
n-1								
n								

rows

n = nrow(df)

Bracket Notation
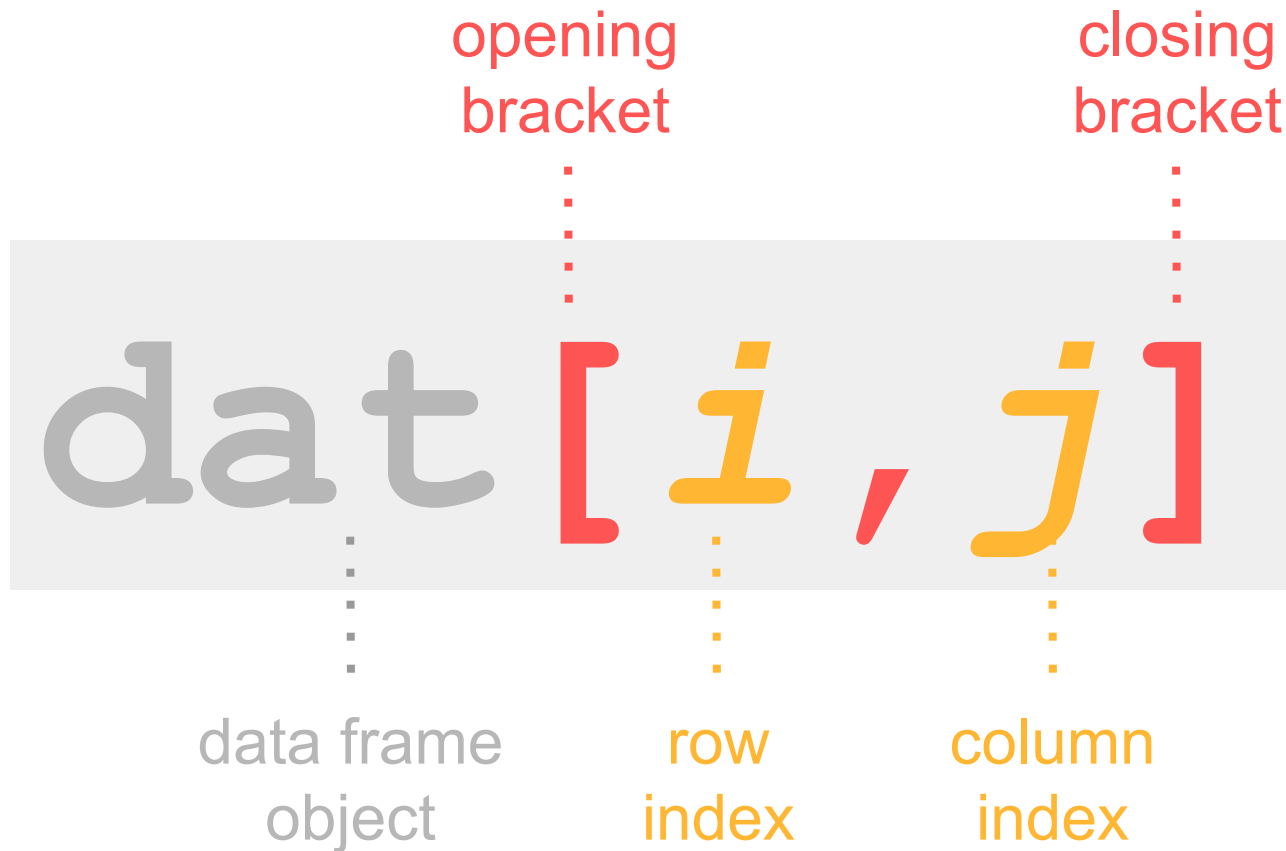
Retrieving Cells

dat[*2,2*]

dat[*2:3,2:3*]

dat[*c(1,3,5),*
c(2,4)]

one single
cell
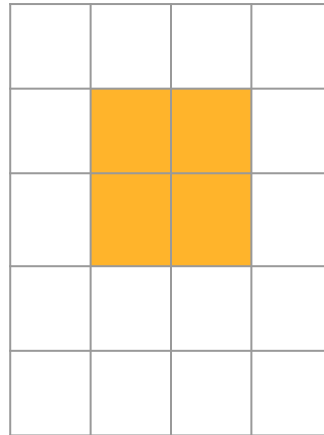
consecutive
cells

separated
cells

Retrieving Cells

```
# first cell 1,1
airquality[1,1]


# cell 9,6
airquality[9,6]


# last cell
airquality[153,6]
```

Retrieving Cells

```r
# various adjacent cells
airquality[1:5,4:6]


# various adjacent cells
#(permuted order)
airquality[5:1,6:4]


# non-adjacent cells
airquality[c(1,50,100),c(3,5)]
```

Retrieving Cells (excluding indices)

df[-2,-2]

one single cell

df[-(2:3), -(2:3)]

consecutive cells

df[-c(1,3), -c(2,4)]

separated cells

Retrieving Cells (excluding indices)

```
# various adjacent cells
airquality[-(1:5),-(4:6)]


# non-adjacent cells
airquality[-c(1,50,100),-c(3,5)]
```
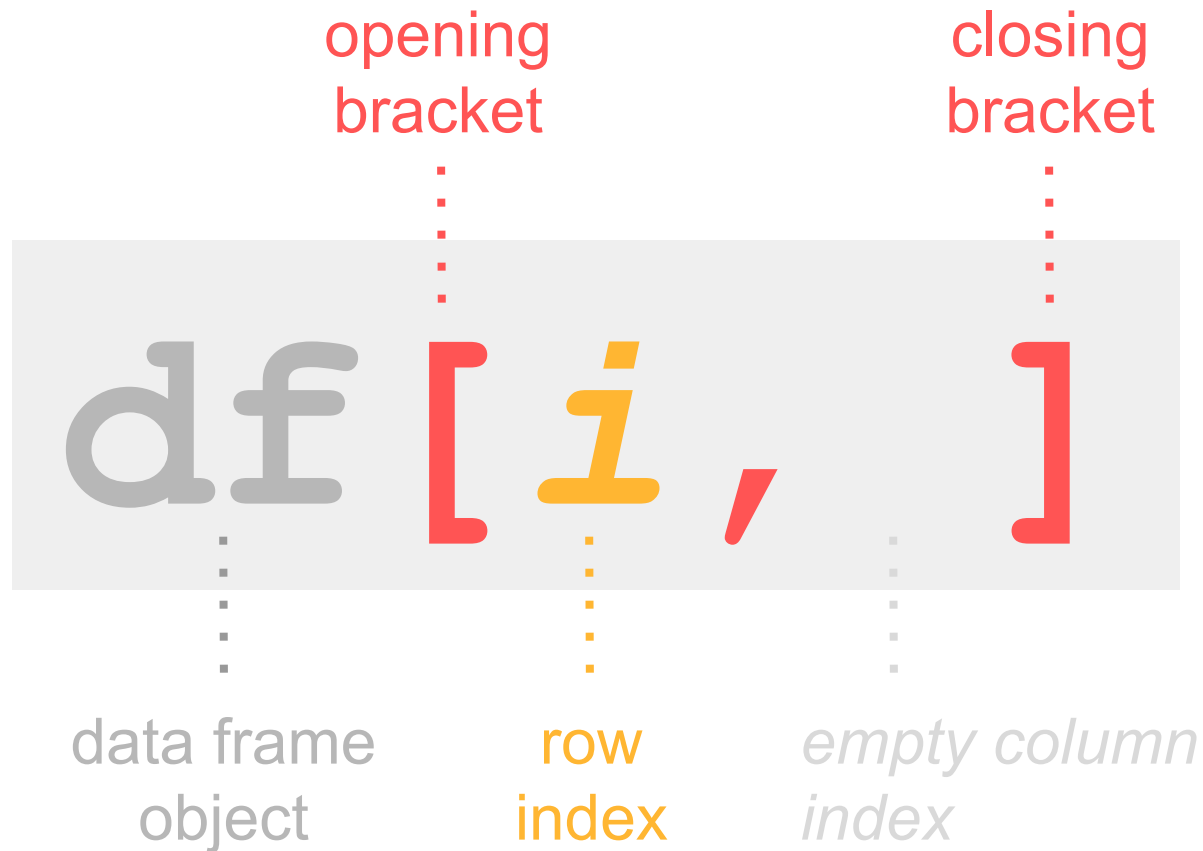
Accessing Cells via Logical Subscripts

df[*ilog,jlog*]

Bracket Notation: retrieving rows

opening
bracket

closing
bracket

df[i,]

data frame
object

row
index

*empty column
index*

Retrieving Rows

df[*1*,]

df[*2:4*,]

df[*c(2,5)*,]

one single
row

consecutive
rows

separate
rows

Retrieving Rows (excluding indices)

df[*-1*,]

df[*-(2:4)*,]

df[*-c(2,5)*,]

one single
row

consecutive
rows

separate
rows

Retrieving Rows

```r
# first row
airquality[1, ]


# rows from 3 to 7
airquality[3:7, ]


# rows 1, 3, 5, 7
airquality[c(1,3,5,7), ]
```

Retrieving Rows (excluding indices)

```r
# all rows except first one
airquality[-1, ]


# rows except from 3 to 7
airquality[-(3:7), ]


# all rows but 1, 3, 5, 7
airquality[-c(1,3,5,7), ]
```

Accessing Rows via Logical Subscripts

`df[`*`logical,`*` `*`]`*`

Retrieving Rows (logical indexing)

```
# records with Month 5
airquality[airquality$Month==5, ]


# records of 1st day of month
airquality[airquality$Day==1, ]
```
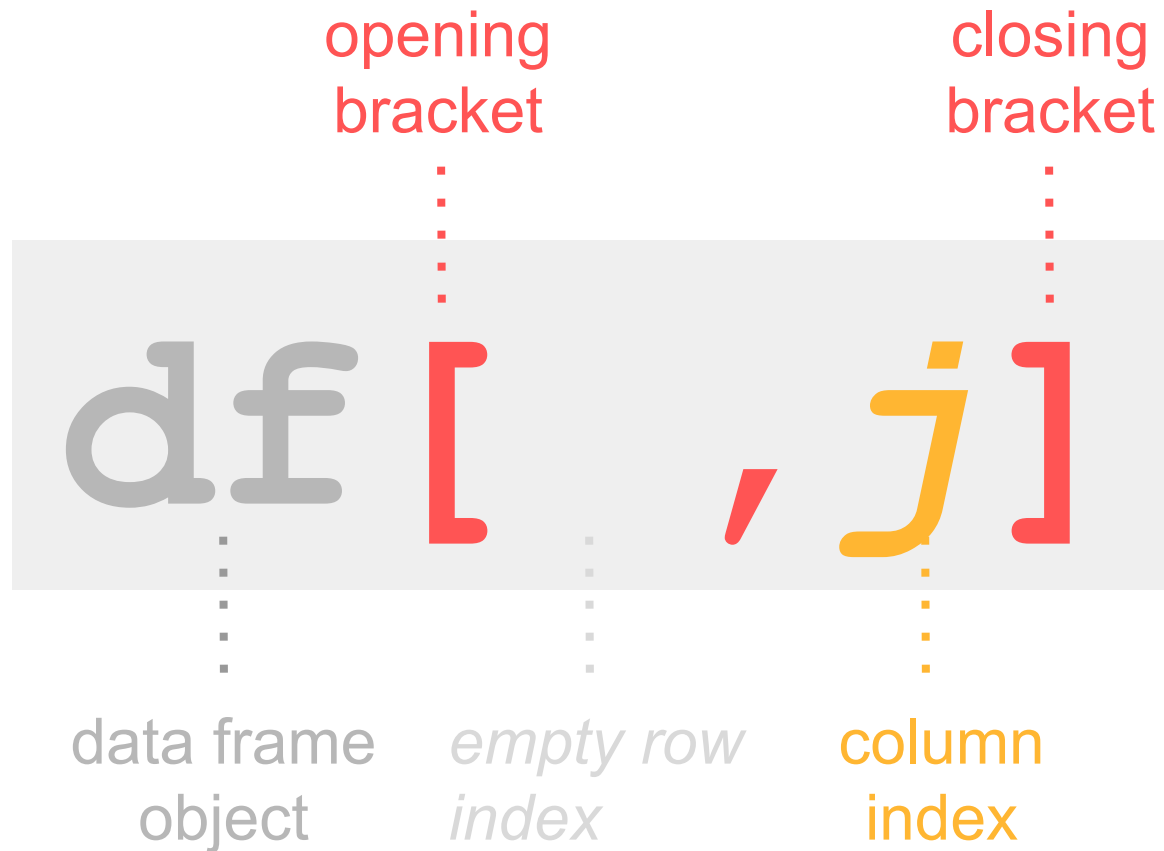
Retrieving Rows (logical indexing)

```r
# vector matching odd numbers
odds = rep(c(TRUE, FALSE),
  length = nrow(airquality))


# odd rows
airquality[odds, ]


# even rows (logical negation)
airquality[!odds, ]
```

Bracket Notation: retrieving columns

opening
bracket

closing
bracket

df[, j]

data frame
object

*empty row
index*

column
index

Retrieving Columns

`df[ ,3]`

`df[ ,1:3]`

`df[ ,c(1,3)]`

one single
column

consecutive
columns

separate
columns

Retrieving Columns

```
# first column
airquality[ ,1]


# columns from 1 to 3
airquality[ ,1:3]


# columns 2, 4, 6
airquality[ ,c(2,4,6)]
```

Retrieving Columns (excluding indices)

df[,-3]   df[,-(1:3)]   df[,-c(1,3)]

one single
column

consecutive
columns

separate
columns

Retrieving Columns (excluding indices)

```r
# excluding first column
airquality[ ,-1]


# columns except 1 to 3
airquality[ ,-(1:3)]


# all columns but 2, 4, 6
airquality[ ,-c(2,4,6)]
```

Accessing Columns via Logical Subscripts

df[,*logical*]

Retrieving Columns (logical indexing)

```r
# look for these names
these = c('Day', 'Wind', 'Rain',
        'Temp', 'XY', 'Snow')


# query logical selection
Q = names(airquality) %in% these


# selecting corresponding columns
airquality[ ,Q]
```

Retrieving Columns (logical indexing)

```r
# logical vector
cols3 = c(rep(TRUE, 3),
          rep(FALSE, 3))


# first 3 columns
airquality[ ,cols3]


# last 3 columns (logical neg)
airquality[ ,!cols3]
```
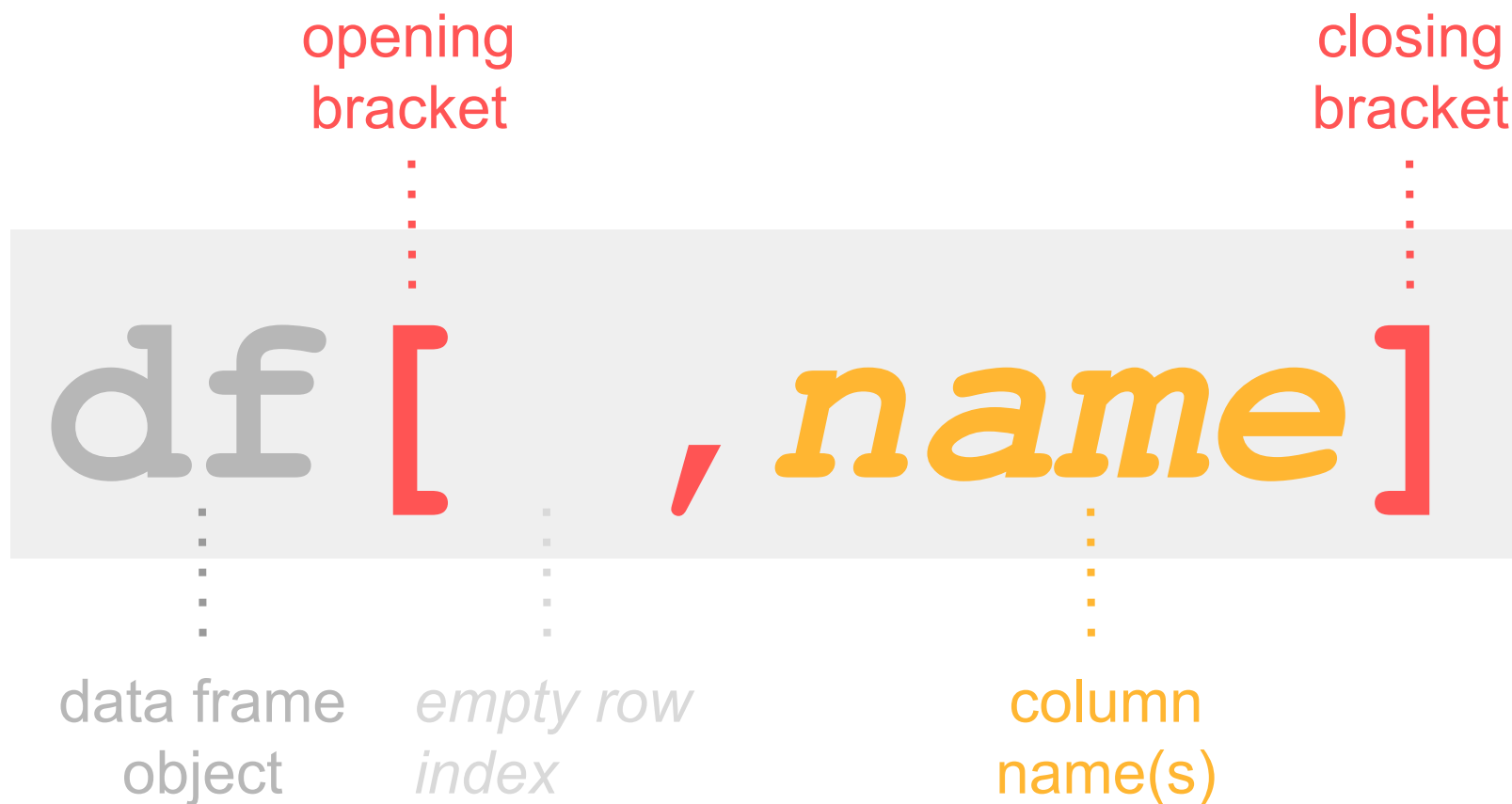
More options to access columns

Bracket Notation: retrieving columns via names

opening
bracket

closing
bracket

df [, *name*]

data frame
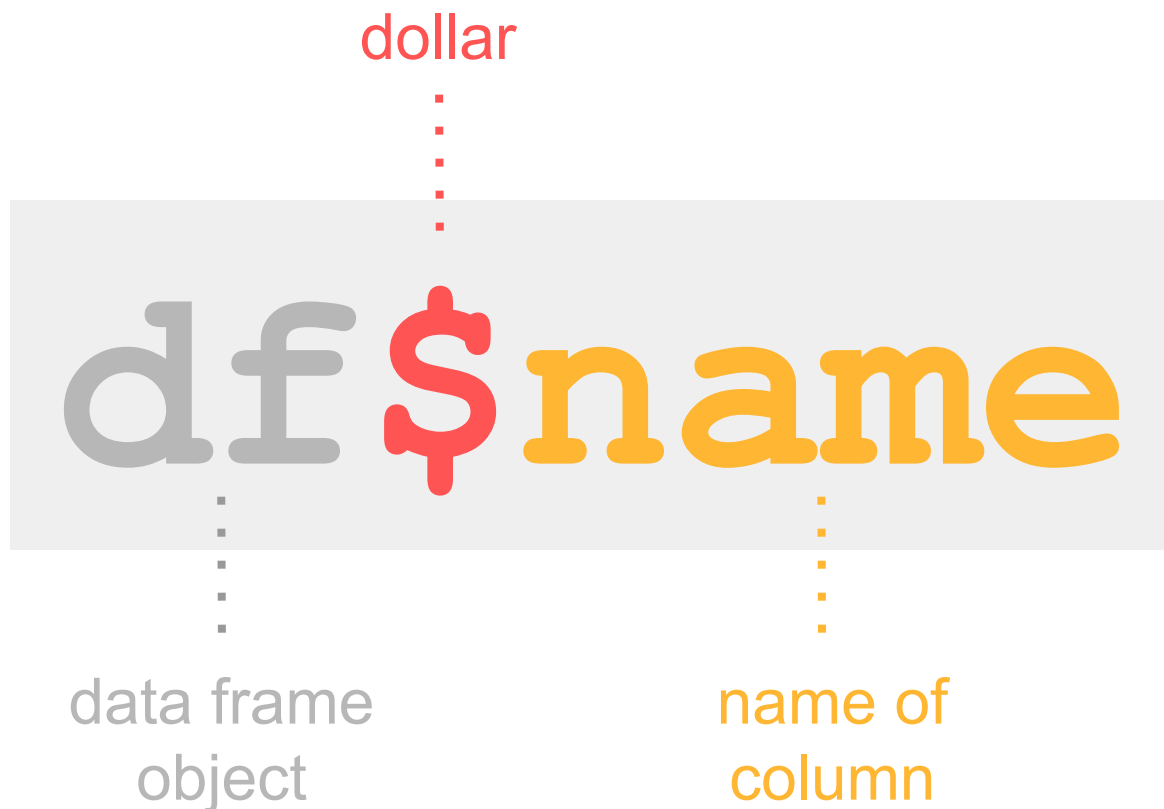object

*empty row
index*

column
name(s)

Retrieving Columns (using names)

```r
# column Ozone
airquality[ ,"Ozone"]


# columns Wind and Temp
airquality[ ,c("Wind","Temp")]
```

Dollar Notation: retrieving columns via names

dollar

d f $ name

data frame
object

name of
column

Accessing One Column

```
# column Ozone
airquality$Ozone


# equivalently
airquality$"Ozone"


# equivalently
airquality$'Ozone'
```

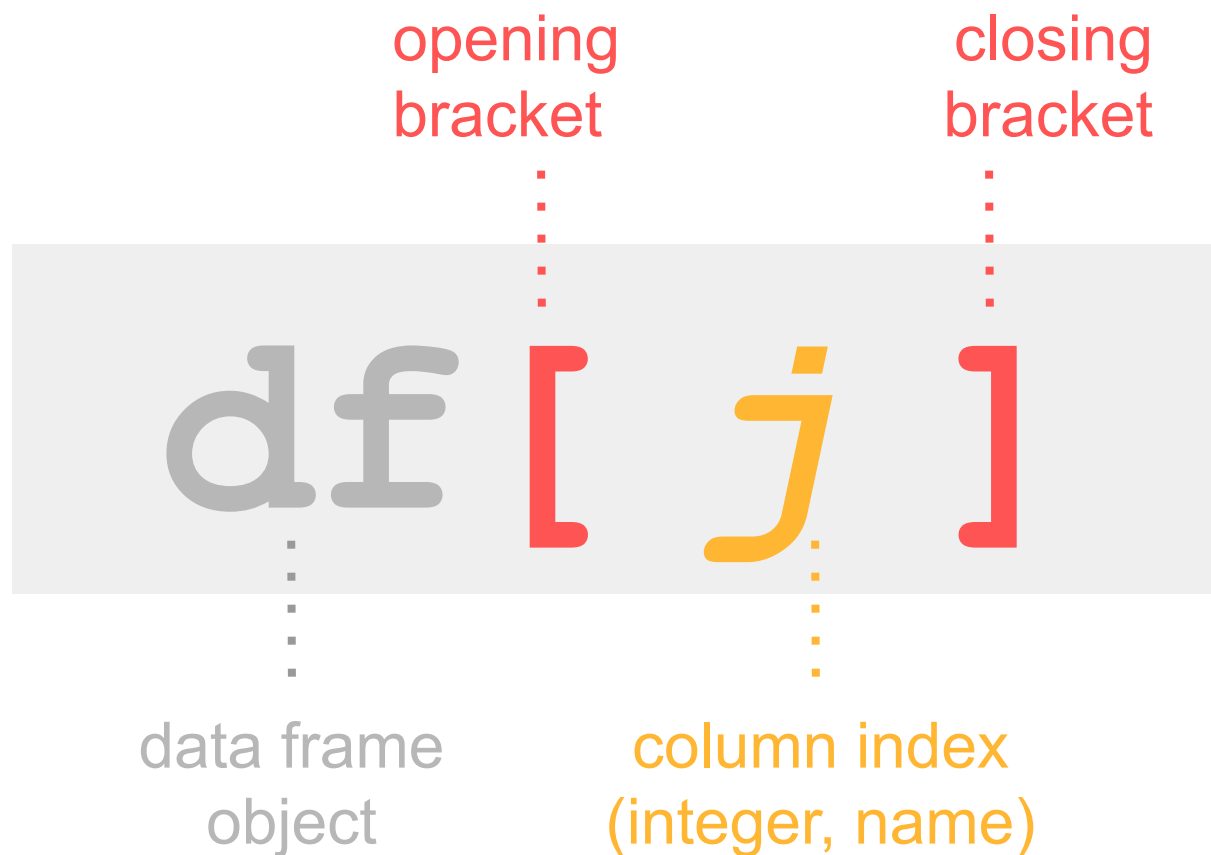# Selecting columns with double brackets

double opening brackets

double closing brackets

df[[j]]

data frame object

single column index (integer, name)
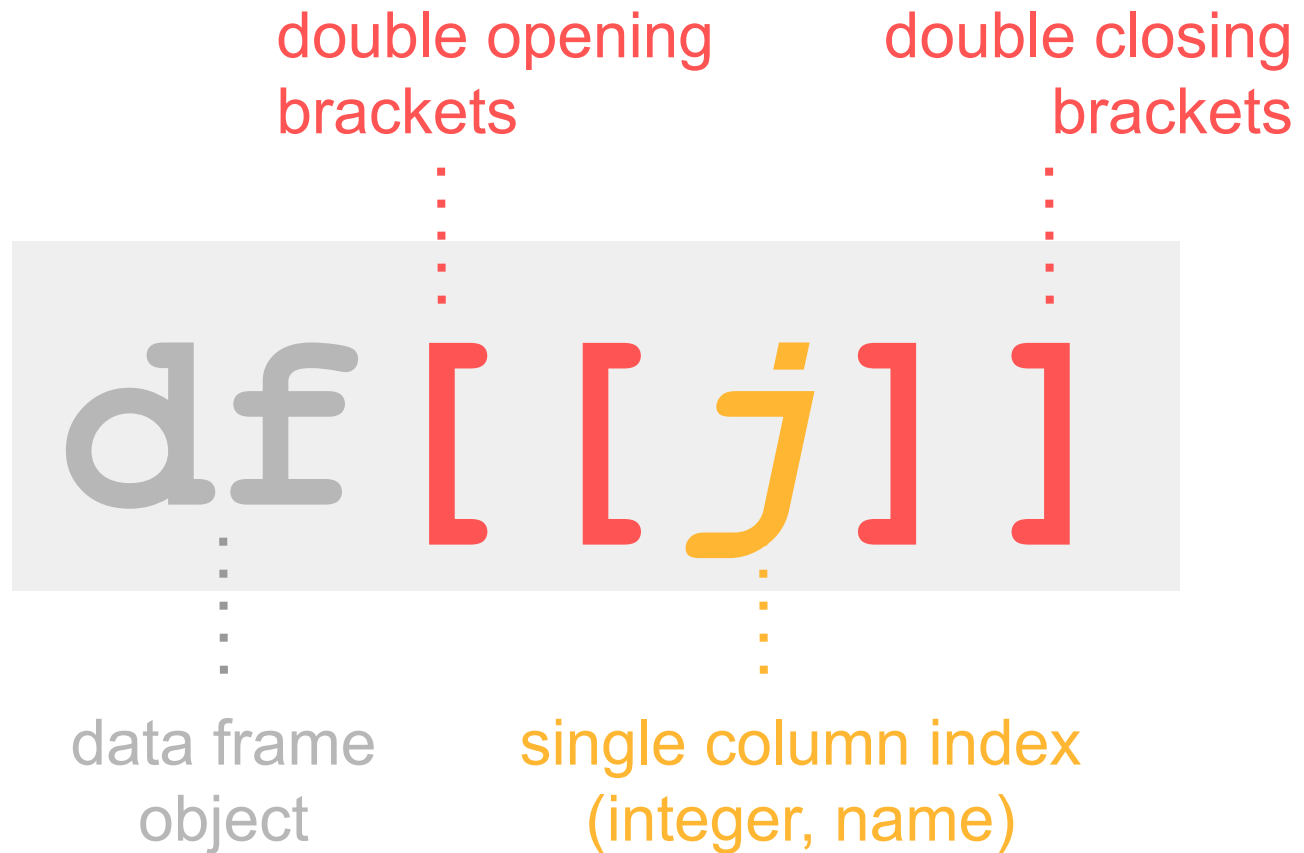
Accessing One Column

```
# first column
airquality[[1]]


# column Wind
airquality[["Wind"]]
```

Selecting columns with vector notation

opening bracket

closing bracket

df[j]

data frame object

column index (integer, name)

Accessing Columns with vector notation

```r
# first column
airquality[1]


# columns from 1 to 3
airquality[1:3]


# columns 2, 4, 6
airquality[c(2,4,6)]
```

Be careful when using this type of syntax since it may create confusion for other users reading your code

Accessing Columns with `list` syntax

```
# column Ozone
airquality["Ozone"]


# columns Ozone and Wind
airquality[c("Ozone","Wind")]
```

Be careful when using this type of syntax since it may create confusion for other users reading your code

Argument **drop** when selecting one column

$$df[i,j,drop=\begin{matrix}TRUE\\FALSE\end{matrix}]$$

drop

TRUE (default) returns result into a vector

FALSE keeps values as a column

Use **drop** to keep result as a column

```
# first column
airquality[ ,1,drop=FALSE]


# column Ozone
airquality[ ,"Ozone",drop=FALSE]
```