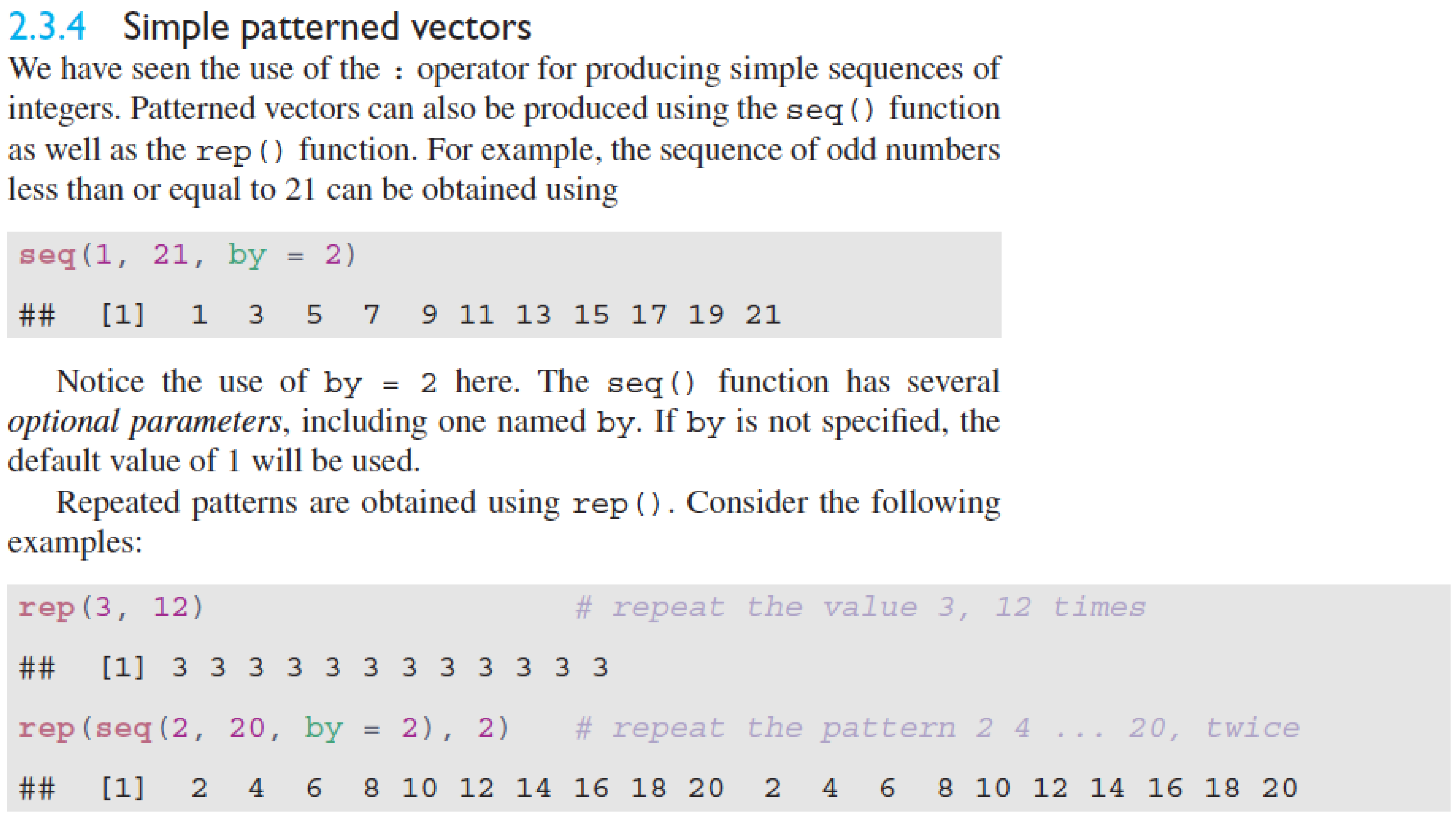
R Programming – Read Data

Lecture 3

- Data Import and Export
- Data Manipulation

Data storage in R



2.3.4 Simple patterned vectors

We have seen the use of the `:` operator for producing simple sequences of integers. Patterned vectors can also be produced using the `seq()` function as well as the `rep()` function. For example, the sequence of odd numbers less than or equal to 21 can be obtained using

```
seq(1, 21, by = 2)

##  [1]  1  3  5  7  9 11 13 15 17 19 21
```

Notice the use of `by = 2` here. The `seq()` function has several *optional parameters*, including one named `by`. If `by` is not specified, the default value of 1 will be used.

Repeated patterns are obtained using `rep()`. Consider the following examples:

```
rep(3, 12)                    # repeat the value 3, 12 times

##  [1] 3 3 3 3 3 3 3 3 3 3 3 3

rep(seq(2, 20, by = 2), 2)    # repeat the pattern 2 4 ... 20, twice

##  [1]  2  4  6  8 10 12 14 16 18 20  2  4  6  8 10 12 14 16 18 20
```

Source: A first course in statistical programming with R - W. John Braun and Duncan J. Murdoch, Cambridge, 2016.

Data storage in R

```
rep(c(1, 4), c(3, 2))     # repeat 1, 3 times and  4, twice

## [1] 1 1 1 4 4

rep(c(1, 4), each = 3)    # repeat each value 3 times

## [1] 1 1 1 4 4 4

rep(1:10, rep(2, 10))     # repeat each value twice

##  [1]  1  1  2  2  3  3  4  4  5  5  6  6  7  7  8  8  9  9 10 10
```

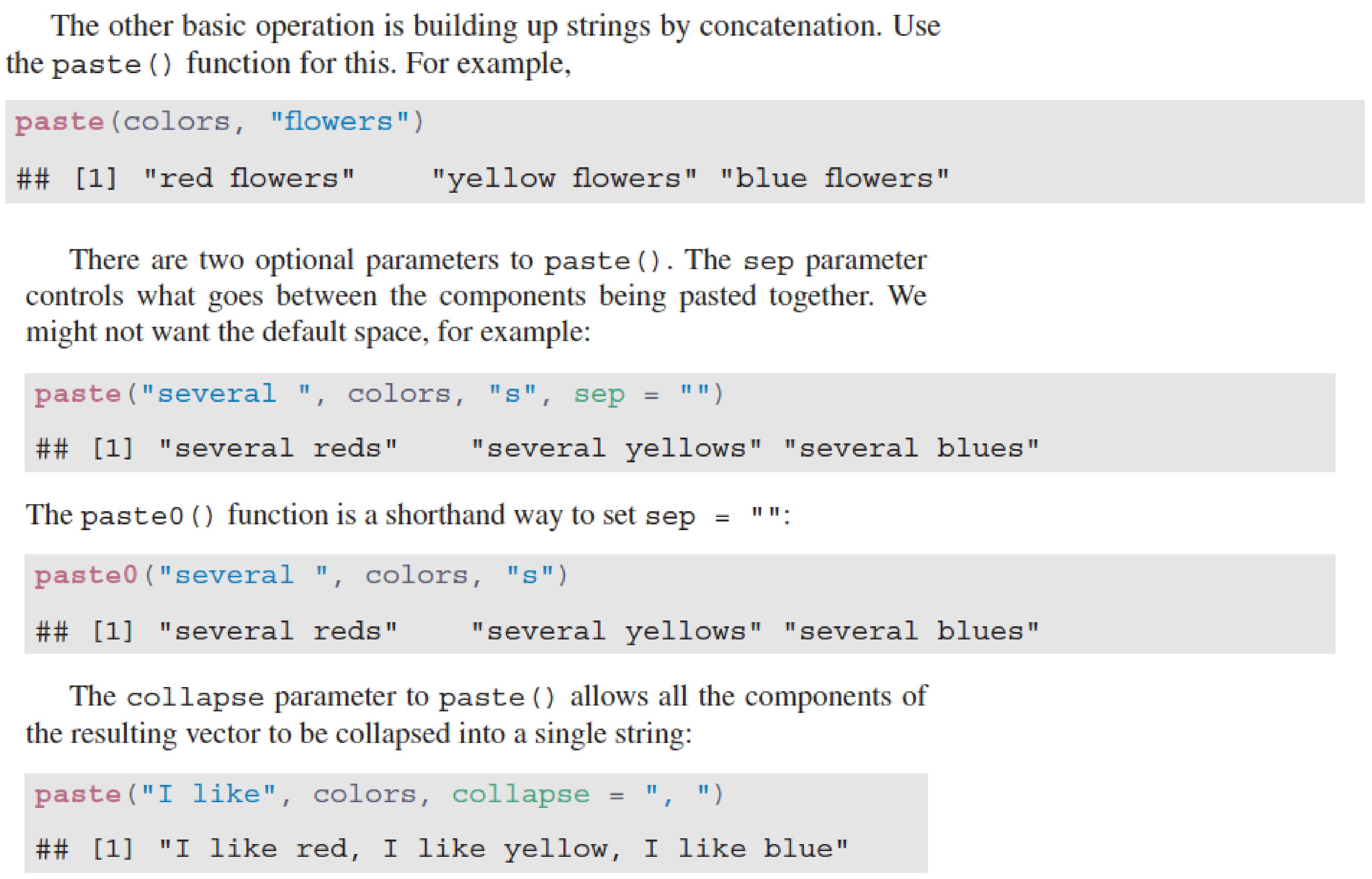
2.3.5 Vectors with random patterns

The `sample()` function allows us to simulate things like the results of the repeated tossing of a 6-sided die.

```
sample(1:6, size = 8, replace = TRUE)  # an imaginary die is tossed 8 times

## [1] 1 5 1 5 3 1 2 1
```

Source: A first course in statistical programming with R - W. John Braun and Duncan J. Murdoch, Cambridge, 2016.

Data storage in R



The other basic operation is building up strings by concatenation. Use the `paste()` function for this. For example,

```
paste(colors, "flowers")

## [1] "red flowers"    "yellow flowers" "blue flowers"
```

There are two optional parameters to `paste()`. The `sep` parameter controls what goes between the components being pasted together. We might not want the default space, for example:

```
paste("several ", colors, "s", sep = "")

## [1] "several reds"    "several yellows" "several blues"
```

The `paste0()` function is a shorthand way to set `sep = ""`:

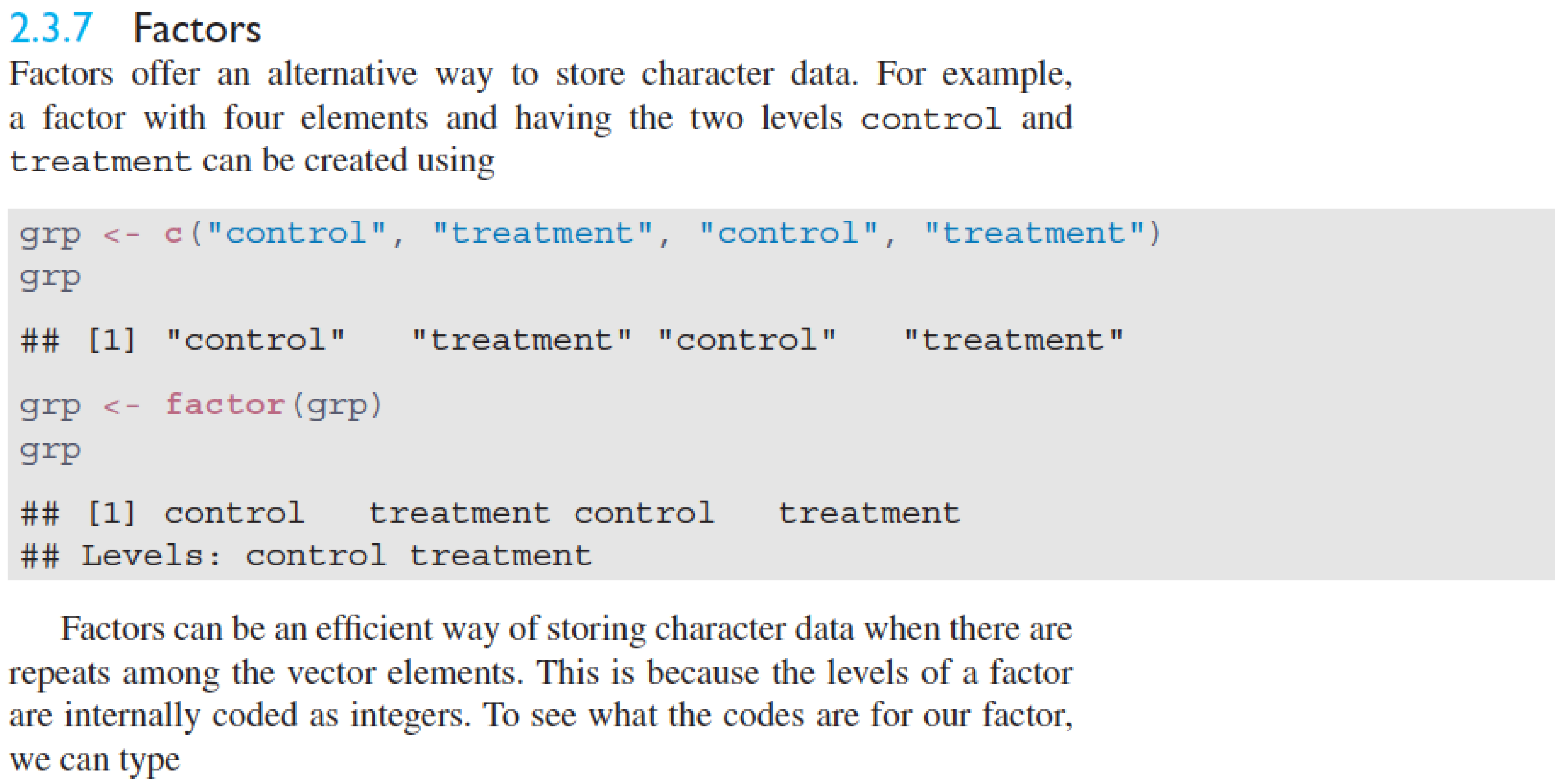
```
paste0("several ", colors, "s")

## [1] "several reds"    "several yellows" "several blues"
```

The `collapse` parameter to `paste()` allows all the components of the resulting vector to be collapsed into a single string:

```
paste("I like", colors, collapse = ", ")

## [1] "I like red, I like yellow, I like blue"
```

Source: A [illegible]

Data storage in R



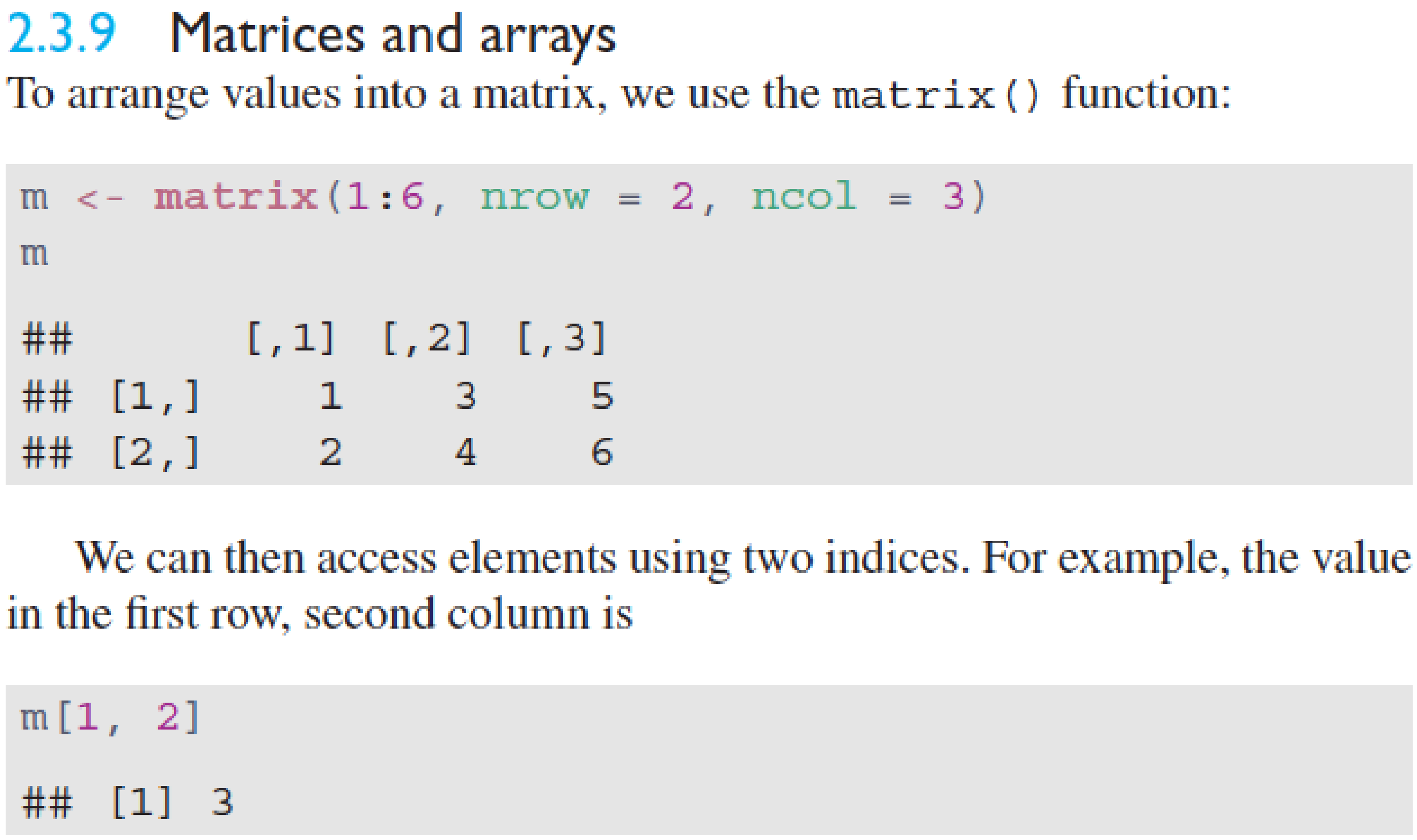
2.3.7 Factors

Factors offer an alternative way to store character data. For example, a factor with four elements and having the two levels `control` and `treatment` can be created using

```
grp <- c("control", "treatment", "control", "treatment")
grp

## [1] "control"   "treatment" "control"   "treatment"

grp <- factor(grp)
grp

## [1] control   treatment control   treatment
## Levels: control treatment
```

Factors can be an efficient way of storing character data when there are repeats among the vector elements. This is because the levels of a factor are internally coded as integers. To see what the codes are for our factor, we can type

Data storage in R



2.3.9 Matrices and arrays

To arrange values into a matrix, we use the `matrix()` function:

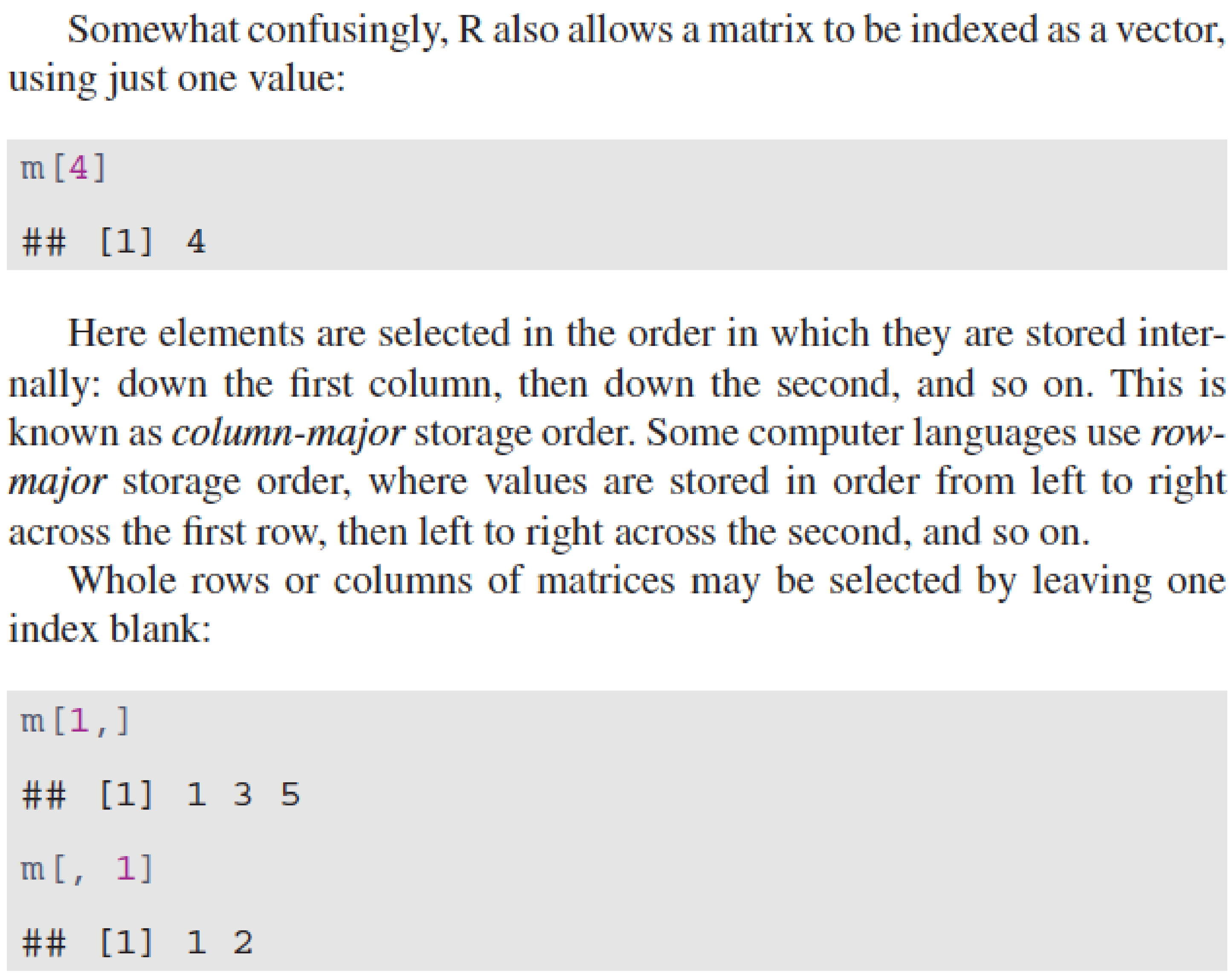
```
m <- matrix(1:6, nrow = 2, ncol = 3)
m

##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

We can then access elements using two indices. For example, the value in the first row, second column is

```
m[1, 2]

## [1] 3
```

Source: A first course in statistical programming with R - W. John Braun and Duncan J. Murdoch, Cambridge, 2016.

Data storage in R



Somewhat confusingly, R also allows a matrix to be indexed as a vector, using just one value:

```
m[4]

## [1] 4
```

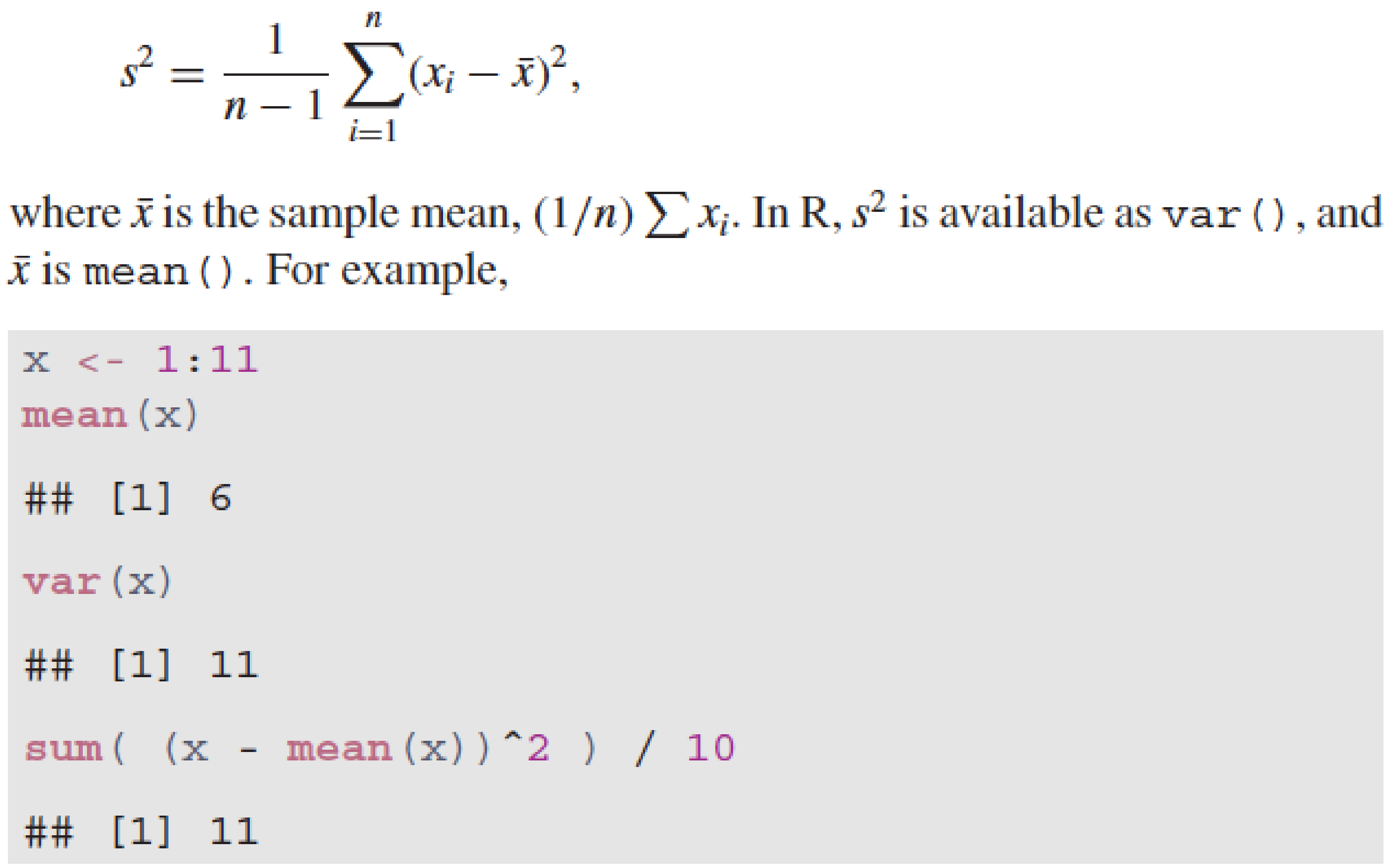
Here elements are selected in the order in which they are stored internally: down the first column, then down the second, and so on. This is known as *column-major* storage order. Some computer languages use *row-major* storage order, where values are stored in order from left to right across the first row, then left to right across the second, and so on.

Whole rows or columns of matrices may be selected by leaving one index blank:

```
m[1,]

## [1] 1 3 5

m[, 1]

## [1] 1 2
```

Source: A first course in statistical programming with R - W. John Braun and Duncan J. Murdoch, Cambridge, 2016.

Data storage in R



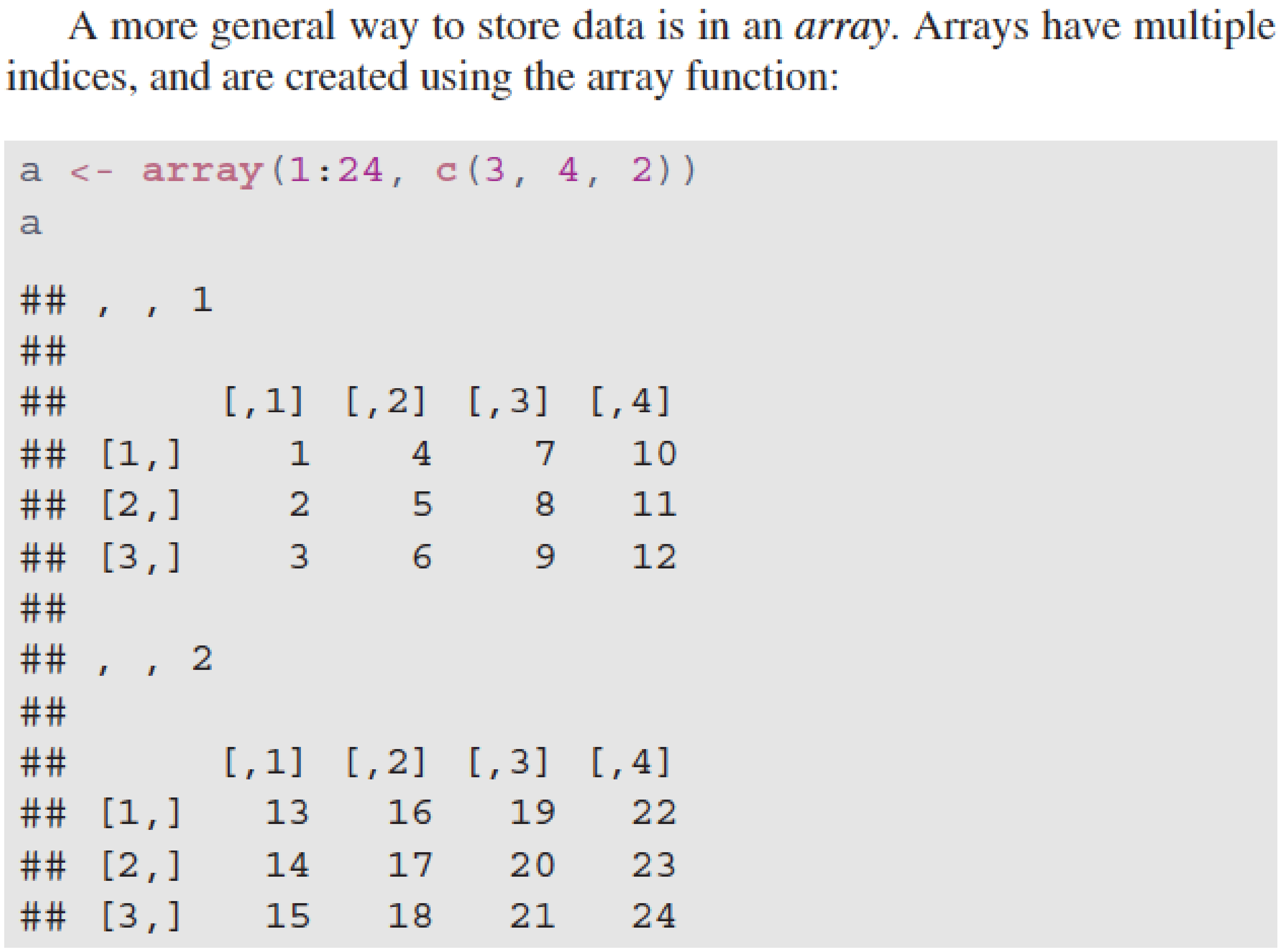
$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2,$$

where $\bar{x}$ is the sample mean, $(1/n)\sum x_i$. In R, s^2 is available as `var()`, and $\bar{x}$ is `mean()`. For example,

```
x <- 1:11
mean(x)

## [1] 6

var(x)

## [1] 11

sum( (x - mean(x))^2 ) / 10

## [1] 11
```

Source: A first course in statistical programming with R - W. John Braun and Duncan J. Murdoch, Cambridge, 2016.

Data storage in R



A more general way to store data is in an *array*. Arrays have multiple indices, and are created using the array function:

```
a <- array(1:24, c(3, 4, 2))
a

## , , 1
##
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
##
## , , 2
##
##      [,1] [,2] [,3] [,4]
## [1,]   13   16   19   22
## [2,]   14   17   20   23
## [3,]   15   18   21   24
```

Source: A first course in statistical programming with R - W. John Braun and Duncan J. Murdoch, Cambridge, 2016.

Common Data File Formats

- .txt text file
- .csv comma separated
- .xls excel file
- .ascii text file

How to load scientific data

- HOPE<- read.table("Fraser_Hope.txt" , head=TRUE)
- SHEL<- read.table("Fraser_Shelley.txt" , head=TRUE)
- x <- read.csv("./enso_index.csv", header = TRUE)
- file.show("obs.csv")
- head(x)

Source: A first course in statistical programming with R - W. John Braun and Duncan J. Murdoch, Cambridge, 2016.

Lab Work

- Calculate bias, Correlation and RMSE of given data
  - Use model and observed data available at course website
  - model_mean_temperature_precipitation_1950-2012
  - observed_mean_temperature_precipitation_1950-2012
-