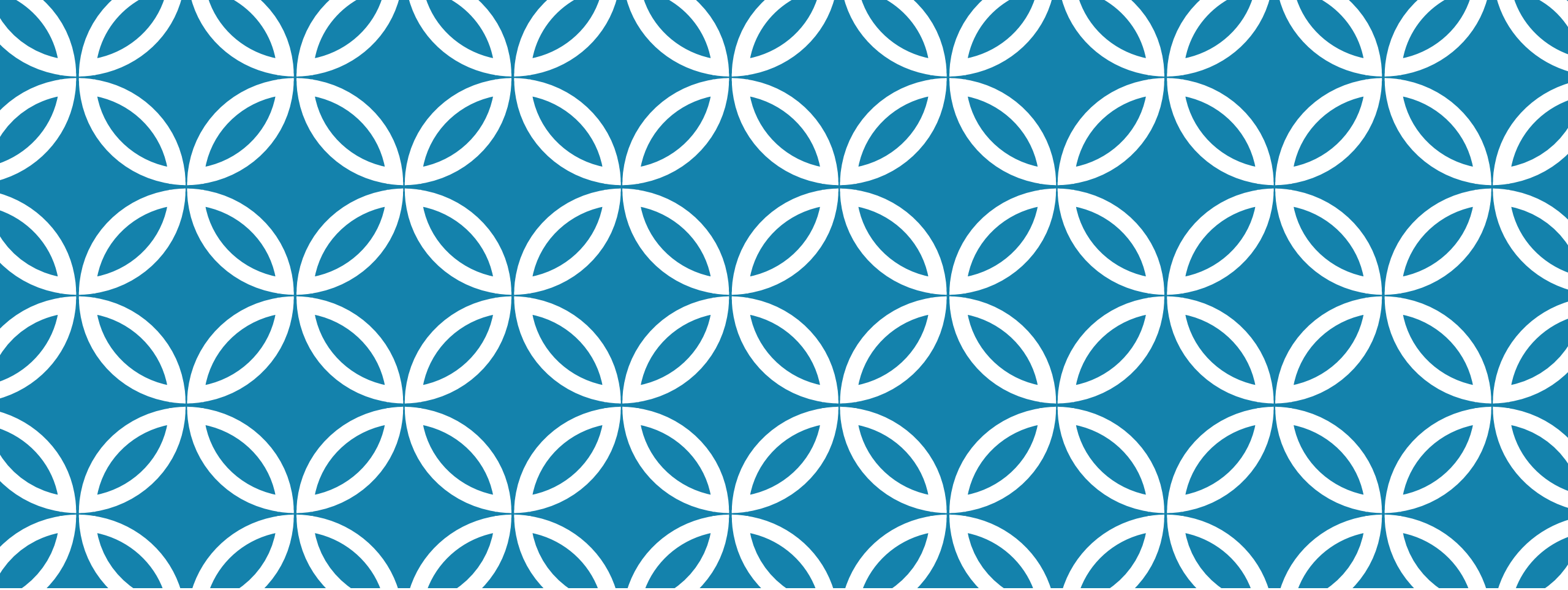


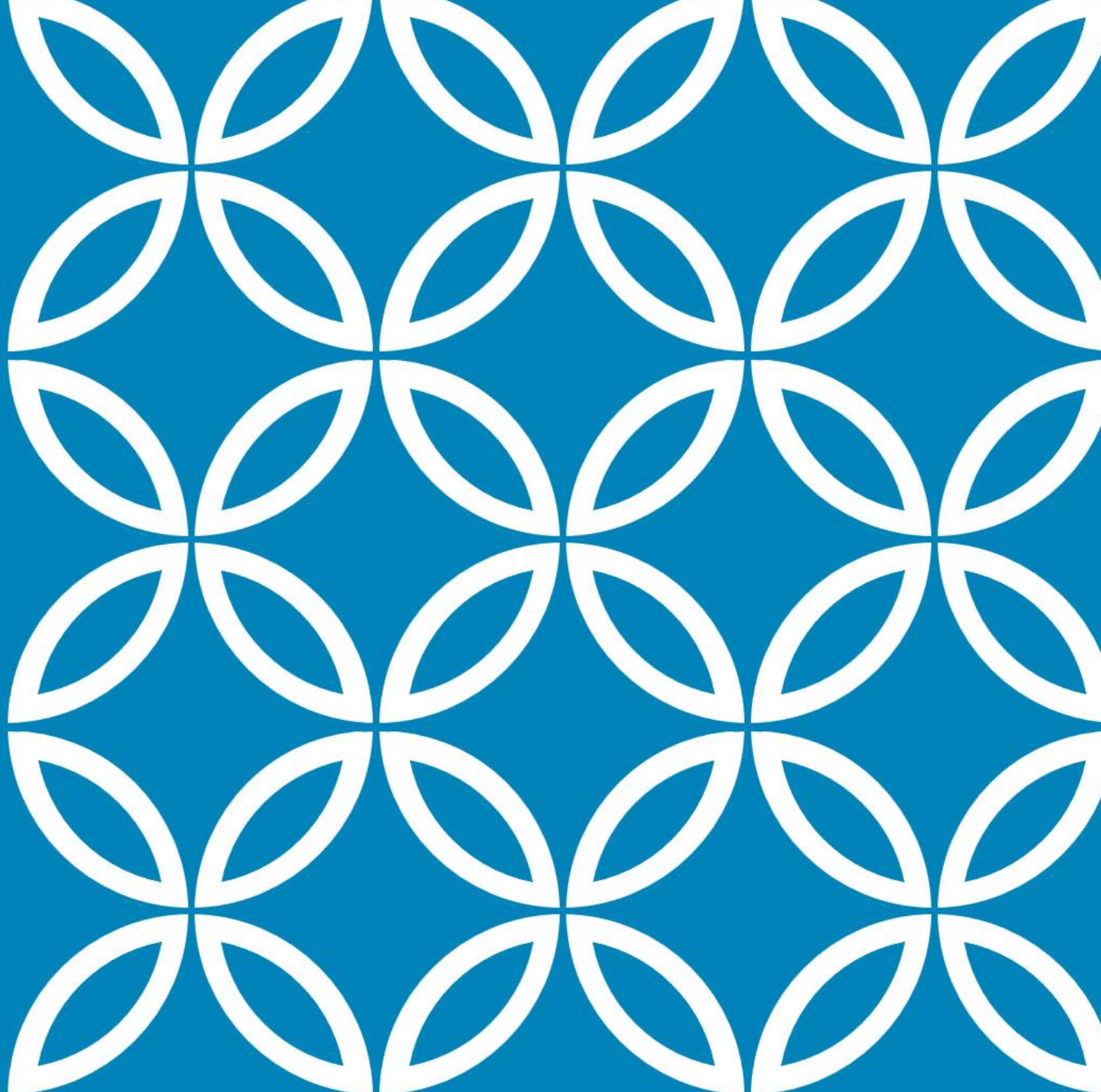
FACTOR, LIST AND MATRIX

mgr Maria Kubara, WNE UW

FACTOR

Special type of a vector, which stores categorical values, with given levels.

Levels can be ordered or unordered. It's important datatype for dividing observations into groups and analyzing them as such.



FACTOR – UNORDERED

Original data (text, categorical)	Coded data (recognizable groups within a categorical variable)
Boat	1
Car	2
Boat	1
Boat	1
Car	2
Device	3

Unordered categorical data – the ordering of levels is not important. The crucial part is the recognition of groups – which observations belong to the same category.

We are creating a sort of legend for the programming language (programming languages prefer to operate on numbers instead of text).
Legend: 1 – boat, 2 – car, 3 – device

FACTOR – ORDERED

Original data (text, categorical)	Coded data (recognizable groups within a categorical variable)
Small	1
Medium	2
Small	1
Small	1
Medium	2
Large	3

Ordered categorical data – the ordering of levels matters
Small << medium << large

CREATING FACTORS

Creating with a factor() function

First, we set the text vector
then we specify the levels

```
> variableLevel <- factor(c("boat", "car", "boat",
+                           "boat", "car", "device"), # vector with text
+                         levels = c("boat", "car", "device"), # levels
+                         ordered = F) # version - unordered factor
> variableLevel
[1] boat   car    boat   boat   car    device
Levels: boat car device
> str(variableLevel)
 Factor w/ 3 levels "boat","car","device": 1 2 1 1 2 3
> # for R it is no longer a vector with text data
> # this vector now stores numbers with certain labeles (every level has a labe
>
> levels(variableLevel)
[1] "boat"   "car"    "device"
```

For unordered factors option
ordered = FALSE

ORDERED FACTOR

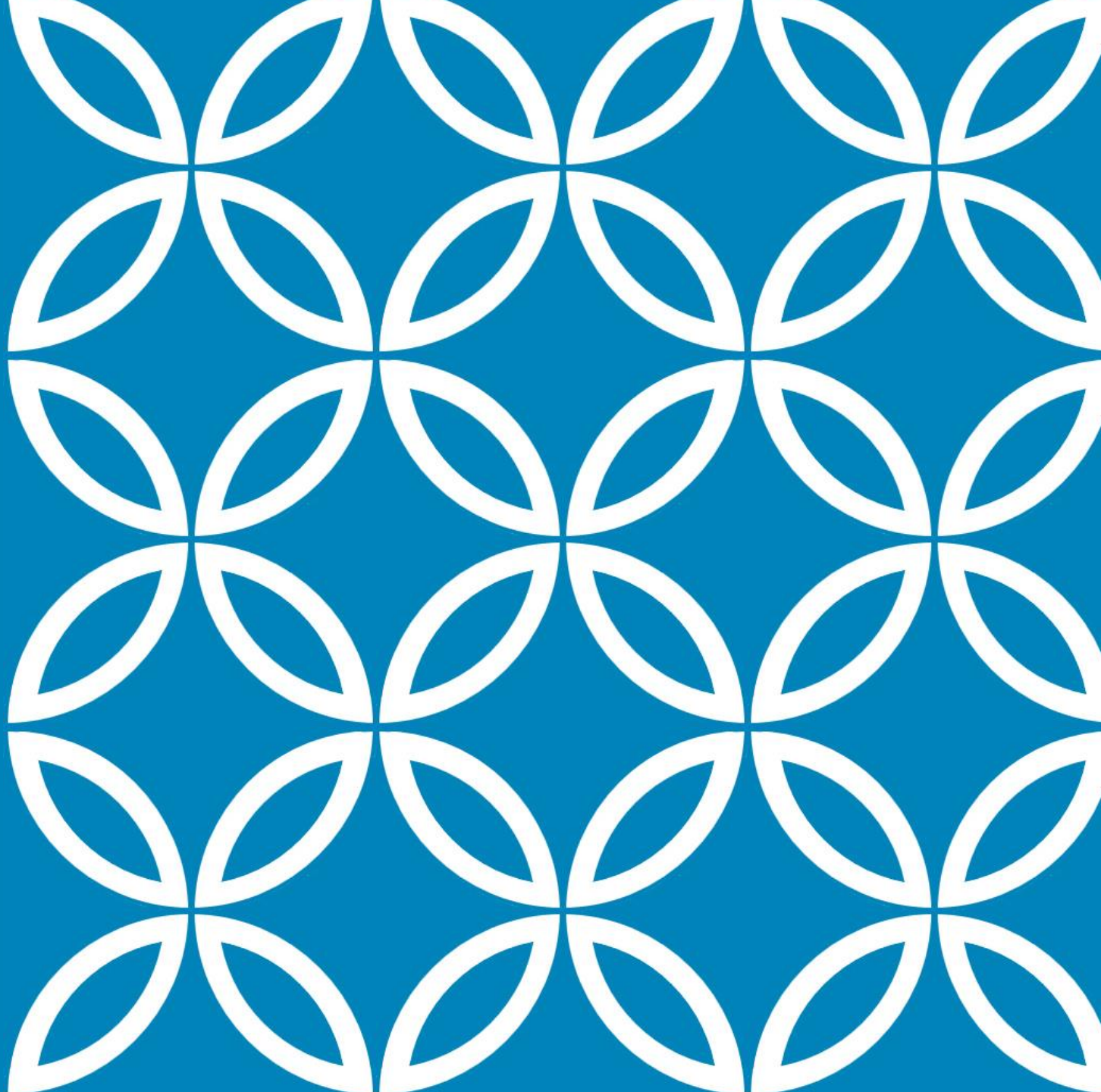
For ordered factor the ordering of levels is important – we set values from the smallest to the largest.

```
> variableLevel2 <- factor(c("small", "medium", "small", "small", "medium", "big"),
+                          levels = c("small", "medium", "big"),
+                          ordered = T) # ordered - ordering of levels is important
>
> str(variableLevel2)  # Ord. factor -> factor variable with ordered levels
 Ord.factor w/ 3 levels "small"<"medium"<..: 1 2 1 1 2 3
> levels(variableLevel2)
[1] "small"  "medium" "big"
```

MATRIX

Set of ordered elements of the same type in two-dimensional rectangular shape.

$$A = \begin{bmatrix} 2 & 7 & 8 & 11 \\ 4 & 1 & 2 & 3 \end{bmatrix}$$



CREATING A MATRIX

Matrix is created with a matrix() function. For defining a matrix, we need to have a vector of values and specified shape (number of columns and rows).

The option of creating an empty matrix of a given size is often used. It can be helpful for storing the data resulting from statistical operations.

```
> elements <- c(2,7,8,11,4,1,2,3)
> length(elements)
[1] 8
>
> A <- matrix( elements,       # vector with elements
+              nrow=2,         # number of rows
+              ncol=4,         # number of columns
+              byrow = TRUE)   # storing values one by one by row
>
> A
     [,1] [,2] [,3] [,4]
[1,]    2    7    8   11
[2,]    4    1    2    3
>
> B <- matrix( elements,       # vector with elements
+              nrow=4,         # number of rows
+              ncol=2,         # number of columns
+              byrow = FALSE) # storing values one by one by column
>
> B
     [,1] [,2]
[1,]    2    4
[2,]    7    1
[3,]    8    2
[4,]   11    3
>
> C <- matrix(NA,             # creating a matrix with empty values
+             nrow=3,         # of given size (dimensions)
+             ncol=2)
> C
     [,1] [,2]
[1,]   NA   NA
[2,]   NA   NA
[3,]   NA   NA
```

INDEXING VALUES FROM MATRIX

A matrix has two dimensions: the number of rows and the number of columns. To get a value from a matrix, we need to specify its position by these two dimensions.

Getting the values is done by indexes at two positions (as if we were working with a two-dimensional vector).

Rule: [RowNumber, ColumnNumber]

```
> # rule: matrix[RowNumber, ColNumber]
>
> A
     [,1] [,2] [,3] [,4]
[1,]    2    7    8   11
[2,]    4    1    2    3
> A[2,4]    # value from the 2nd row and 4th column
[1] 3
> A[2,]
[1] 4 1 2 3
> A[,4]     # all the values from the 4th column
[1] 11  3
> A[1,2:3]  # values from the 1st row and columns 2nd and 3rd (range)
[1] 7 8
```

Omitting the row (column) index will cause all its elements to show (there was no limitation on the index).

Similarly to taking values from vectors, you can get more elements from the matrix by a vector of indexes (here 2:3 in the column index)

FILLING MATRIX VALUES

Using the indexing of matrix elements, we can assign new values to chosen positions.

Warning: new value overrides the previous value stores in the given place.

```
> # assigning values to the matrix (for given position)
>
> C
     [,1] [,2]
[1,]   NA   NA
[2,]   NA   NA
[3,]   NA   NA
> C[1,2] <- 3
> C
     [,1] [,2]
[1,]   NA    3
[2,]   NA   NA
[3,]   NA   NA
>
> C[,1] <- 4 # vector of 4 (automatically setting the size
> C
     [,1] [,2]
[1,]    4    3
[2,]    4   NA
[3,]    4   NA
>
> C[3,] <- c(9,15)
> C
     [,1] [,2]
[1,]    4    3
```

Assigning new single value to specific position

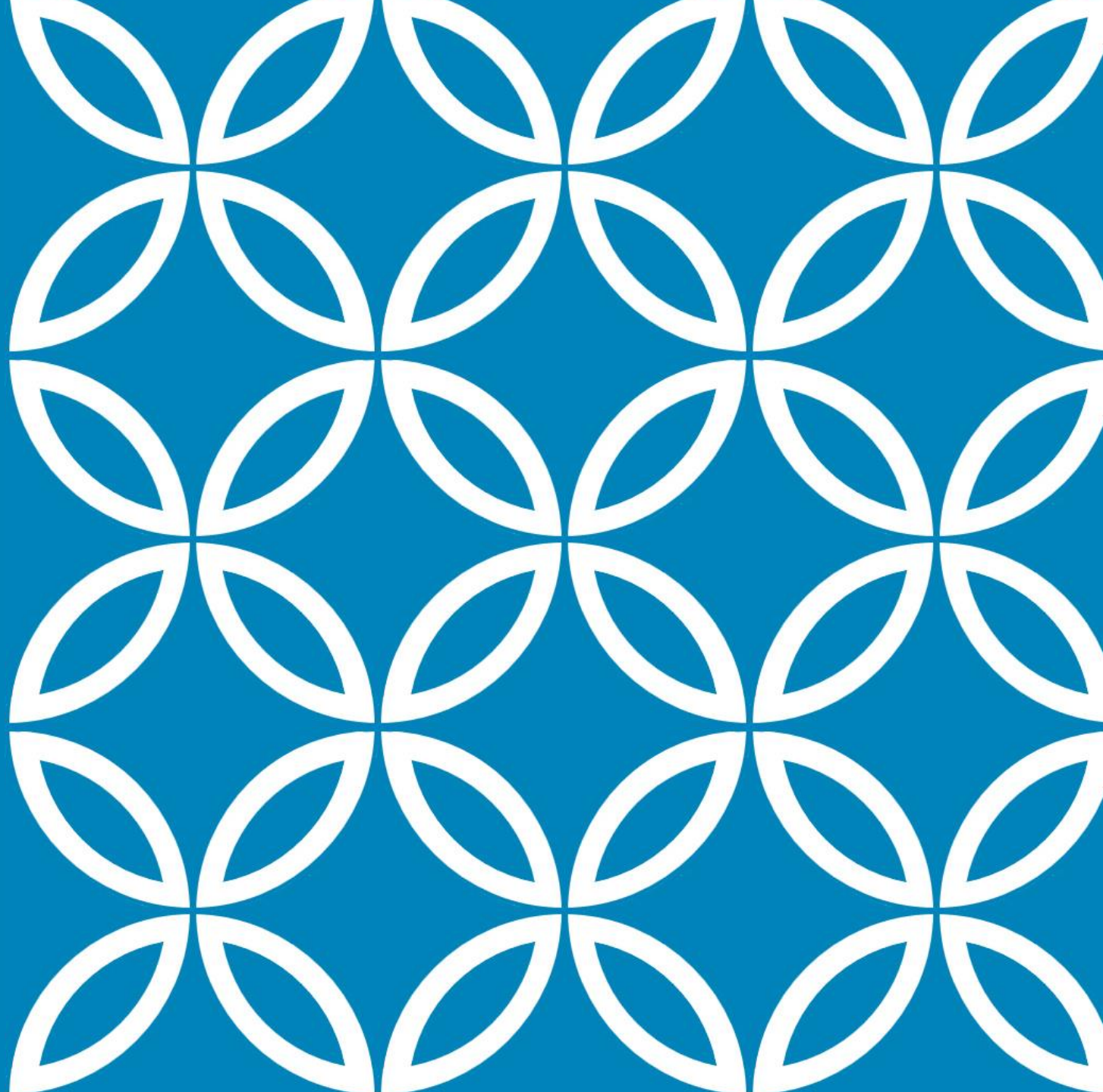
Filling the 1st column with a vector of 4 (in this case R is adjusting the length of the input to the size of the column – according to the recycling rule)

Filling values with a two-element vector to the 3rd row of the matrix

LIST

List can be seen as a more general vector, which can store diverse elements inside (a list can store a set of vectors, matrices, data frames, etc.)

Important – in a vector all elements need to be of the same type. In a list we can store diverse types. List's elements can be of any length or dimension.



CREATING A LIST

Lists are created with list() function, which takes the elements of the newly created list as arguments.

Elements of the list are stored as copies of the original objects – changing a list element doesn't change the original object.

```
> first <- c("I'm", "a vector", "of type", "character")
> second <- c(1,7,20.5) # vector with numbers (numeric)
> third <- c(TRUE, TRUE, TRUE, FALSE) # logical vector
> fourth <- 0 # single number
> fifth <- matrix(5, nrow=2, ncol=3)
>
> listObject <- list(first, second, third, fourth, fifth)
>
> listObject
[[1]]
[1] "I'm"       "a vector"  "of type"   "character"

[[2]]
[1]  1.0  7.0 20.5

[[3]]
[1]  TRUE  TRUE  TRUE FALSE

[[4]]
[1] 0

[[5]]
     [,1] [,2] [,3]
[1,]    5    5    5
```

Elements of the list can have different types and structures

INDEXING LIST ELEMENTS (BOXES)

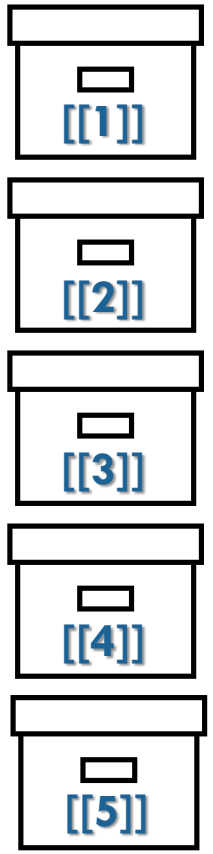
```
> listObject
[[1]]
[1] "I'm"       "a vector"  "of type"   "character"

[[2]]
[1]  1.0  7.0 20.5

[[3]]
[1]  TRUE  TRUE  TRUE FALSE

[[4]]
[1] 0

[[5]]
     [,1] [,2] [,3]
[1,]    5    5    5
```



Informal box visualization ☺

You can imagine a list as a set of numbered boxes stored in a warehouse – every box can store different things.

In order to take an item from that warehouse, we need to specify the number of the box we want to search in, and then the exact position of the element we want from that box.

INDEXING LIST ELEMENTS

Indexing elements of the list (*boxes*) works as follows:

ListObject[1] – element from the first position (*entire first box*)

ListObject[[1]] – content of the element on the first position (*only the content of the first box*)

ListObject[[1]][2] – second element of the object, stored at the first position of the list (*second element from the first box*)

```
> # Indexing the list elements
> listObject[1]
[[1]]
[1] "I'm"       "a vector"  "of type"   "character"

> listObject[[1]]
[1] "I'm"       "a vector"  "of type"   "character"
> listObject[[1]][2]
[1] "a vector"
```

MODIFYING LIST ELEMENTS

```
> # Modifying the list elements
> listObject[1] <- c("Exchanging", "first", "vector") # Error! We want to change the content
Warning message:
In listObject[1] <- c("Exchanging", "first", "vector") :
  number of items to replace is not a multiple of replacement length
> listObject[[1]] <- c("Exchanging", "first", "vector")
> listObject[[5]][2,3] <- 20 # exchanging the value within the matrix stored at the fifth position
>
> listObject[[6]] <- c("Adding", "a new", "element")
>
> listObject
[[1]]
[1] "Exchanging" "first"      "vector"

[[2]]
[1]  1.0  7.0 20.5

[[3]]
[1]  TRUE  TRUE  TRUE FALSE

[[4]]
[1] 0

[[5]]
     [,1] [,2] [,3]
[1,]    5    5    5
[2,]    5    5   20

[[6]]
[1] "Adding"  "a new"   "element"
```

We need to target the content of the element – using double squared brackets

Adding a new element (*box*) to the list – by adding new index and declaring its content