Chapter 2: Data structures

Contents

1 Vectors **1**
- 1.1 Vector arithmetic . . . 2
- 1.2 Logical vectors . . . 4
- 1.3 Character vectors . . . 5
- 1.4 Coercing . . . 5
- 1.5 Missing values . . . 6
- 1.6 Subset of a vector . . . 6
- 1.7 Give names to elements of a vector . . . 6

2 Factors **6**

3 Matrices **7**
- 3.1 Creating a matrix in R . . . 7
- 3.2 Adding labels . . . 8
- 3.3 Calculations on matrices . . . 8
- 3.4 Subset of a matrix . . . 10

4 Data frame **11**
- 4.1 Creating a data frame . . . 11
- 4.2 Taking a subset from a data frame . . . 13

5 List **14**
- 5.1 Creating a list . . . 15
- 5.2 Named list . . . 15
- 5.3 Accessing elements of a list . . . 15

6 Exercises **16**
- 6.1 Vectors . . . 16
- 6.2 Creating sequences . . . 16
- 6.3 Matrix creation . . . 16
- 6.4 Working with data frames . . . 16

R operates on data structures. There are different data structures: vector, factor, matrix, data frame and list.

1 Vectors

- Simplest type of data structure
- **Vector creation**: with function `c`:

```
vec <- c(5, 8, 4.6, 21)
```

- To **see** the content of a vector created and assigned to an object, call the name of this object:

```
vec
```

```
## [1]  5.0  8.0  4.6 21.0
```

OR

use an extra pair of brackets during the creation and assignment of the object:

```
(vec <- c(5, 8, 4.6, 21)) # Note the extra brackets!
```

```
## [1]  5.0  8.0  4.6 21.0
```

1.1 Vector arithmetic

1.1.1 Useful functions for vectors

1 vector: x	2 vectors: x, y
`length(x)`	
`sum(x)`	`sum(x, y)`
`prod(x)`	`prod(x, y)`
`max(x)`	`max(x, y)`
`min(x)`	`min(x, y)`
`sort(x)`	`sort(x, y)`
`rev(x)`	
`diff(x, lag = )`	
`unique(x)`	
`order(x)`	

Creating a vector `x` with 10 random elements drawn from a normal distribution. Some of the above listed functions will be afterwards applied on this vector

```
x <- rnorm(10)
x
```

```
##  [1]  1.16380713 -2.02159814  0.25914675 -0.40558169  0.27371012  1.30685921
##  [7] -0.07173989  1.02192342 -0.37770545 -1.86810075
```

The function `order()` applied on a vector `x` returns a vector with indices. The indices reveal which element of the original vector `x` needs to be put first, second ... in order to sort the vector `x` in ascending (default) or descending order.

```
z <- order(x)
z
```

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

The function `sort()` gives the elements itself instead of the index.

```
sort(x)
```

```
##  [1] -2.02159814 -1.86810075 -0.40558169 -0.37770545 -0.07173989  0.25914675
##  [7]  0.27371012  1.02192342  1.16380713  1.30685921
```

```
# OR, alternative way to obtain the same result
sort(x, decreasing = FALSE)
```

```
##  [1] -2.02159814 -1.86810075 -0.40558169 -0.37770545 -0.07173989  0.25914675
##  [7]  0.27371012  1.02192342  1.16380713  1.30685921
```

The function `rev()` returns a reversed version of its argument.

```
rev(sort(x))
```

```
##  [1]  1.30685921  1.16380713  1.02192342  0.27371012  0.25914675 -0.07173989
```

```
##  [7] -0.37770545 -0.40558169 -1.86810075 -2.02159814
```

OR, alternative way to obtain the same result

```
sort(x, decreasing = TRUE)
```

```
##  [1]  1.30685921  1.16380713  1.02192342  0.27371012  0.25914675 -0.07173989
##  [7] -0.37770545 -0.40558169 -1.86810075 -2.02159814
```

Creation of a vector `y` which will be used as argument for some other examples.

```
y <- c(1, 3, 8, 3, 7, 21)
```

The function `unique()` extracts unique elements.

```
unique(y)
```

```
## [1]  1  3  8  7 21
```

The function `diff()` returns suitably lagged and iterated differences.

```
diff(y, lag = 2)
```

```
## [1]  7  0 -1 18
```

1.1.2 Mathematical functions on numerical vectors

`abs`, `exp`, `log`, `floor`, `ceiling`, `trunc`, `gamma`, `log10`, `round`, `sin`, `cos`, `tan`, `sqrt`

1.1.3 Generating vectors

How to create regular sequences?

Colon operator :	→ To create a regular sequence
`seq`	→ To create a regular sequence
`rep`	→ To replicate value

Use of the colon operator (`:`) to create a regular sequence:

```
y <- 1:8
y
```

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

To obtain a descending sequence

```
x <- 20:1
```

Use of the function `seq`

```
z <- seq(from = -2, to = 5, by = 0.5)
z
```

```
##  [1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0
```

Use of the function `rep`

```
u <- rep(2:5, 2)
u
```

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

```
v <- rep(2:5, 2:5)
v
```

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

1.2 Logical vectors

The elements of a logical vector can have the values `T` (`TRUE`) or `F` (`FALSE`).

```
x <- c(9, 10, 8, 5, 9) # numerical vector
y <- x<9 # logical vector
y
```

```
## [1] FALSE FALSE  TRUE  TRUE FALSE
```

1.2.1 Logical expressions

Expression	Meaning
<	Smaller than
<=	Smaller than or equal to
>	Larger than
>=	Larger than or equal to
==	Equal to
!=	Unequal to

1.2.2 Logical operators

Operator	Meaning
&	And
|	Or

1.2.3 Examples

```
T + T
```

```
## [1] 2
```

```
T - F
```

```
## [1] 1
```

```
x <- rnorm(100)
y <- x>0
y
```

```
##   [1] FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE
##  [13]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
##  [25]  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [37] FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
##  [49] FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
##  [61]  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE
##  [73]  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE
##  [85]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE
##  [97] FALSE FALSE  TRUE  TRUE
```

```r
sum(y)
```

```
## [1] 59
```

1.3 Character vectors

```r
test <- c("Leuven Statistics Research Centre", "2020-2021")
test
```

```
## [1] "Leuven Statistics Research Centre" "2020-2021"
```

1.3.1 Character vector operations

some character functions:

1. **nchar(text)**
   This function returns a vector with number of characters in each element of the argument **text**

```r
x <- c("start", "student")
nchar(x)
```

```
## [1] 5 7
```

2. **paste**
   Concatenate vectors after converting to character.

```r
paste(c("X", "Y"), 1:4, sep=" ")
```

```
## [1] "X 1" "Y 2" "X 3" "Y 4"
```

3. **substring(text, start, stop)**
   Extract substrings in a character vector.

```r
x <- c("start", "student")
substring(x, 1, 3)
```

```
## [1] "sta" "stu"
```

1.4 Coercing

Logical, numerical and character values can be used in one and the same vector.

```r
vector <- c(TRUE, -6.05, "Leuven")
vector
```

```
## [1] "TRUE"   "-6.05"  "Leuven"
```

Data type	Example
Logical	`T` or `F`
Numeric	`-6.05`, `86.06`, ...
Character	`Alabama`, `Leuven`, ...

When values of different modes are combined into one object, then R converts all values to a single mode in a way that preserves as much information as possible.
Increasing order information: logical, numeric, character

1.5 Missing values

NA (Not Available) is the symbol used in R to represent missing data (for logical, numerical or character values).

```
y <- c(1,2,3, NA)
is.na(y)
```

```
## [1] FALSE FALSE FALSE  TRUE
```

```
z <- c("Ward", "Wouter", "Lucas", NA)
!is.na(z)
```

```
## [1]  TRUE  TRUE  TRUE FALSE
```

```
x <- c(5,3,8,NA, 6)
is.na(x>5)
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE
```

1.6 Subset of a vector

A part of a vector `x` can be selected by **`x[subscript]`**

```
y <- c(33, 55, 4, 22, 89)
y[c(2,3)]
```

```
## [1] 55  4
```

```
y[-5]
```

```
## [1] 33 55  4 22
```

```
y[y>30]
```

```
## [1] 33 55 89
```

```
x <- c(5, 9, NA)
y <- x[!is.na(x)]
y
```

```
## [1] 5 9
```

1.7 Give names to elements of a vector

```
y <- c(33, 55, 4, 22, 89)
names(y) <- 1:5
y
```

```
##  1  2  3  4  5
## 33 55  4 22 89
```

2 Factors

- A factor is a vector that contains predefined values. This type of data structure is used to store categorical data. The set of allowed values in a factor are defined by the levels.
- How to create?
  - `factor()`: To create a factor
  - `as.factor()`: To encode a vector as a factor

```r
directions <- c("North", "East", "South", "South") # This is a character vector
directions
```

```
## [1] "North" "East"  "South" "South"
```

```r
directions.f <- as.factor(directions) # This is a factor
directions.f
```

```
## [1] North East  South South
## Levels: East North South
```

To create a factor with labels

```r
status <- factor(c("Medium", "Low", "High", "Medium", "Medium", "High"),
                 levels = c("Low", "Medium", "High"),
                 labels = c("Lo", "Me", "Hi"))
status
```

```
## [1] Me Lo Hi Me Me Hi
## Levels: Lo Me Hi
```

3 Matrices

All the values need to be of the same type.

3.1 Creating a matrix in R

How to create?:

- `matrix()`: To create a matrix
- `rbind()`: To add rows
- `cbind()`: To add columns

Example of the use of function `matrix()`

```r
x <- matrix(1:8, nrow = 2, ncol = 4, byrow = FALSE)
x
```

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

Some properties of objects in R are stored as attributes. Examples of attributes are `dim` (the dimension of the object), `dimnames` (the names associated with the dimension of the object.) The attributes of an object can be accessed with the function `attributes`.

```r
attributes(x)
```

```
## $dim
## [1] 2 4
```

Example of the use of function `rbind()`
`rbind()` combines the arguments row-wise.

```r
x <- rbind(1:4, 5:8)
x
```

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

```r
dim(x)
```

```
## [1] 2 4
```

```r
dimnames(x)
```

```
## NULL
```

```r
attributes(x)
```

```
## $dim
## [1] 2 4
```

Example of the use of function `cbind()`
`cbind()` combines the arguments column-wise.

```r
x <- cbind(1:4, 5:8)
x
```

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

3.2 Adding labels

Adding labels to the rows and to the columns of a matrix:

```r
dimnames(x) <- list(paste("row", 1:2), paste("col", LETTERS[1:4]))
```

```r
paste("row", 1:2)
```

```
## [1] "row 1" "row 2"
```

```r
paste("col", LETTERS[1:4])
```

```
## [1] "col A" "col B" "col C" "col D"
```

The row variable names comes first, then the column variable names.

```r
x <- rbind(1:4, 5:8) # Creating matrix
dimnames(x) <- list(paste("row", 1:2), paste("col", LETTERS[1:4])) # Adding labels
x
```

```
##       col A col B col C col D
## row 1     1     2     3     4
## row 2     5     6     7     8
```

3.3 Calculations on matrices

```r
x <- matrix(1:4, nrow = 2) # Creating a matrix x
x
```

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

```r
nrow(x) # Returns number of rows present in x
```

```
## [1] 2
```

```r
ncol(x) # Returns number of columns present in x
```

```
## [1] 2
```

```r
x+x
```

```
##      [,1] [,2]
## [1,]    2    6
## [2,]    4    8
```

Element-wise product:

```r
x*x # Elementwise product
```

```
##      [,1] [,2]
## [1,]    1    9
## [2,]    4   16
```

Element-wise division:

```r
x/x # Elementwise division
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    1    1
```

```r
x^2
```

```
##      [,1] [,2]
## [1,]    1    9
## [2,]    4   16
```

Matrix multiplication: `%*%`

```r
x%*%x # Matrix product
```

```
##      [,1] [,2]
## [1,]    7   15
## [2,]   10   22
```

```r
y <- c(1,5)
y
```

```
## [1] 1 5
```

yxy^T:

```r
y%*%x%*%y
```

```
##      [,1]
## [1,]  126
```

```r
z <- rbind(x, y)
z
```

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

```r
z <- cbind(x, y)
z
```

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

```
a <- rbind(1:4, 5:8)
b <- cbind(1:4, 5:8)
a%*%b # Matrix multiplication
```

```
##      [,1] [,2]
## [1,]   30   70
## [2,]   70  174
```

Matrix specific operations

Function	Function returns
`chol(x)`	Choleski decomposition of `x`
`col(x)`	Matrix of which the elements corresponds to the column number of the elements
`row(x)`	Matrix of which the elements corresponds to the row numbers of the elements
`diag(x)`	Diagonal matrix from vector `x`
`ncol(x)`	Number of columns of matrix `x`
`nrow(x)`	Number of rows of matrix `x`
`qr(x)`	QR matrix decomposition
`solve(x)`	Inverse
`svd(x)`	Singular value decomposition
`var(x)`	Covariance matrix of the columns
`t(x)`	Transpose
`eigen(x)`	Eigenvalues and eigenvectors of `x`

Functions for mathematical computing

Function	Action
`solve(a, b)`	Solve the system ax = b for x
`integrate(f, low = a, high = b)`	Integration
`polyroot`	Optimization: Find zeros of a real or complex polynomial.
`uniroot`	Optimization: Search in a interval for a root (i.e., zero) of a function.
`opimize`	Optimization: Search in a interval for a minimum or maximum of a function.
`approx`	Interpolation
`rnorm`, `pnorm`, `dnorm`, `qnorm`	Random generation, distribution function, density and quantile function for the normal distribution

3.4 Subset of a matrix

```
x <- matrix(1:100, ncol = 5) # Creating a matrix x
x
```

```
##       [,1] [,2] [,3] [,4] [,5]
##  [1,]    1   21   41   61   81
```

```
##  [2,]    2   22   42   62   82
##  [3,]    3   23   43   63   83
##  [4,]    4   24   44   64   84
##  [5,]    5   25   45   65   85
##  [6,]    6   26   46   66   86
##  [7,]    7   27   47   67   87
##  [8,]    8   28   48   68   88
##  [9,]    9   29   49   69   89
## [10,]   10   30   50   70   90
## [11,]   11   31   51   71   91
## [12,]   12   32   52   72   92
## [13,]   13   33   53   73   93
## [14,]   14   34   54   74   94
## [15,]   15   35   55   75   95
## [16,]   16   36   56   76   96
## [17,]   17   37   57   77   97
## [18,]   18   38   58   78   98
## [19,]   19   39   59   79   99
## [20,]   20   40   60   80  100
```

```
x[8:12, 3:4] # Returns subset with rows from 8- 12 and columns 3-4
```

```
##      [,1] [,2]
## [1,]   48   68
## [2,]   49   69
## [3,]   50   70
## [4,]   51   71
## [5,]   52   72
```

```
x[2,] # Returns 2nd row of the matrix
```

```
## [1]  2 22 42 62 82
```

```
x[,2] # Returns 2nd column of the matrix
```

```
##  [1] 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
```

4 Data frame

- Data frames are an extension of matrices. **Data frames can have columns of different data types.** Most statistical routines require a data frame as input.

4.1 Creating a data frame

- To create or change a data frame:
  - `data.frame`: Create data frames
  - `cbind`: Combine data frames by column (add new variables to existing data frame, merge data frames with same individuals)
  - `rbind`: Combine data frames by row (add new individuals to existing data frame, merge data frames with same variables)
  - `merge`: Join data frames by their common data

Create a data frame from vectors by using `data.frame`

```
employee <- c("John", "Peter", "Sylvie") # Data type = character
salary <- c(21000, 23400, 26800) # Data type = numeric
```

```r
employ.df <- data.frame(employee, salary) #Creation of a data frame with two columns.
#These columns have a different data type.
employ.df
```

```
##   employee salary
## 1     John  21000
## 2    Peter  23400
## 3   Sylvie  26800
```

Adding a new row by rbind function

```r
new.df <- data.frame(employee = "Katrien", salary = 25000)
employ2 <- rbind(employ.df, new.df)
head(employ2)
```

```
##   employee salary
## 1     John  21000
## 2    Peter  23400
## 3   Sylvie  26800
## 4  Katrien  25000
```

Adding a new column by cbind function

```r
age <- c(40, 23, 31)
employ3 <- cbind(employ2, age)
head(employ3) # This gives an error
```

```r
age <- c(40, 23, 31, NA) #Note the extra element in this vector (i.e. NA) which is
#crucial, otherwise an error is generated.
employ3 <- cbind(employ2, age)
head(employ3)
```

```
##   employee salary age
## 1     John  21000  40
## 2    Peter  23400  23
## 3   Sylvie  26800  31
## 4  Katrien  25000  NA
```

Create a data frame by using merge function

```r
authors <- data.frame(
    ## I(*) : use character columns of names to get sensible sort order
    surname = I(c("Tukey", "Venables", "Tierney", "Ripley", "McNeil")),
    nationality = c("US", "Australia", "US", "UK", "Australia"),
    deceased = c("yes", rep("no", 4)))

books <- data.frame(
    name = I(c("Tukey", "Venables", "Tierney",
             "Ripley", "Ripley", "McNeil", "R Core")),
    title = c("Exploratory Data Analysis",
              "Modern Applied Statistics ...",
              "LISP-STAT",
              "Spatial Statistics", "Stochastic Simulation",
              "Interactive Data Analysis",
              "An Introduction to R"),
    other.author = c(NA, "Ripley", NA, NA, NA, NA,
                     "Venables & Smith"))
```

Examples where the data frames `authors` and `books` are merged with the `merge` function.

```
authors
```

```
##    surname nationality deceased
## 1    Tukey          US      yes
## 2 Venables   Australia       no
## 3  Tierney          US       no
## 4   Ripley          UK       no
## 5   McNeil   Australia       no
```

```
books
```

```
##       name                           title     other.author
## 1    Tukey     Exploratory Data Analysis             <NA>
## 2 Venables Modern Applied Statistics ...           Ripley
## 3  Tierney                     LISP-STAT             <NA>
## 4   Ripley            Spatial Statistics             <NA>
## 5   Ripley         Stochastic Simulation             <NA>
## 6   McNeil     Interactive Data Analysis             <NA>
## 7   R Core           An Introduction to R Venables & Smith
```

```
m1 <- merge(authors, books, by.x = "surname", by.y = "name")
# by.x and by.y specify the columns used for merging
m1
```

```
##    surname nationality deceased                          title other.author
## 1   McNeil   Australia       no     Interactive Data Analysis         <NA>
## 2   Ripley          UK       no            Spatial Statistics         <NA>
## 3   Ripley          UK       no         Stochastic Simulation         <NA>
## 4  Tierney          US       no                     LISP-STAT         <NA>
## 5    Tukey          US      yes     Exploratory Data Analysis         <NA>
## 6 Venables   Australia       no Modern Applied Statistics ...       Ripley
```

```
m2 <- merge(books, authors, by.x = "name", by.y = "surname")
m2
```

```
##       name                           title other.author nationality deceased
## 1   McNeil     Interactive Data Analysis         <NA>   Australia       no
## 2   Ripley            Spatial Statistics         <NA>          UK       no
## 3   Ripley         Stochastic Simulation         <NA>          UK       no
## 4  Tierney                     LISP-STAT         <NA>          US       no
## 5    Tukey     Exploratory Data Analysis         <NA>          US      yes
## 6 Venables Modern Applied Statistics ...       Ripley   Australia       no
```

4.2 Taking a subset from a data frame

1. Take a subset from a data frame like a **matrix**

```
sub1 <- authors[1:3, 1:2]
sub1
```

```
##    surname nationality
## 1    Tukey          US
## 2 Venables   Australia
## 3  Tierney          US
```

2. Taking a subset from a data frame like a **list** (to be explained later)

```
sub2 <- authors$nationality
sub2
```

```
## [1] US        Australia US        UK        Australia
## Levels: Australia UK US
```

Remark:
To sort the data frame `airquality` by two variables, use the function `orderBy` from the package `doBy`.

More information about the `airquality` data:

```
?airquality
```

```
## starting httpd help server ... done
```

airquality {datasets} R Documentation

New York Air Quality Measurements

Description

Daily air quality measurements in New York, May to September 1973.

Usage

```
airquality
```

Format

A data frame with 153 observations on 6 variables.

`[,1] Ozone`	numeric Ozone (ppb)
`[,2] Solar.R`	numeric Solar R (lang)
`[,3] Wind`	numeric Wind (mph)
`[,4] Temp`	numeric Temperature (degrees F)
`[,5] Month`	numeric Month (1--12)
`[,6] Day`	numeric Day of month (1--31)

Details

Daily readings of the following air quality values for May 1, 1973 (a Tuesday) to September 30, 1973.

- `Ozone`: Mean ozone in parts per billion from 1300 to 1500 hours at Roosevelt Island
- `Solar.R`: Solar radiation in Langleys in the frequency band 4000–7700 Angstroms from 0800 to 1200 hours at Central Park
- `Wind`: Average wind speed in miles per hour at 0700 and 1000 hours at LaGuardia Airport
- `Temp`: Maximum daily temperature in degrees Fahrenheit at La Guardia Airport.

```
sort_air <- orderBy(~Month + Temp, data = airquality)
head(sort_air)
```

```
##    Ozone Solar.R Wind Temp Month Day
## 5     NA      NA 14.3   56     5   5
## 18     6      78 18.4   57     5  18
## 25    NA      66 16.6   57     5  25
## 27    NA      NA  8.0   57     5  27
## 15    18      65 13.2   58     5  15
## 26    NA     266 14.9   58     5  26
```

5 List

- A **list** can contain elements of different data types.

- It is an ordered collection of components.

5.1 Creating a list

Creation of a list with the function `list`.

```
employee <- c("John", "Peter", "Sylvie") # creation of a character vector
salary <- c(21000, 23400, 26800) # creation of a numeric vector
employ.df <- data.frame(employee, salary) # creation of a data frame

y <- letters[1:5] # creation of a character vector
z <- 1:3 # creation of a numeric vector

list(employ.df, y, z)
```

```
## [[1]]
##   employee salary
## 1     John  21000
## 2    Peter  23400
## 3   Sylvie  26800
##
## [[2]]
## [1] "a" "b" "c" "d" "e"
##
## [[3]]
## [1] 1 2 3
```

5.2 Named list

The elements of a list can have a name, i.e., named lists have the attributes `names`.

```
mixlist <- list(logica = c(T, T, T, F),
                plant = c("tree", "bush", "grass"),
                comment = "these components are unrelated")
mixlist
```

```
## $logica
## [1]  TRUE  TRUE  TRUE FALSE
##
## $plant
## [1] "tree"  "bush"  "grass"
##
## $comment
## [1] "these components are unrelated"
```

```
attributes(mixlist)
```

```
## $names
## [1] "logica"  "plant"   "comment"
```

5.3 Accessing elements of a list

To access a list component: use component number or component name.

```
mixlist$plant # Returns a character vector
```

```
## [1] "tree"  "bush"  "grass"
```

```
mixlist[[2]] # Returns a character vector
```

```
## [1] "tree"  "bush"  "grass"
```

```
mixlist[2] # Returns a list
```

```
## $plant
## [1] "tree"  "bush"  "grass"
```

6 Exercises

6.1 Vectors

a. Create the object `test1` with numbers 1.5, 0.7, 45.6.
b. Create a vector `y1` with the numbers from 1 to 10.
c. Create a logical vector `y2` from `y1`. An element of `y2` should be `TRUE` if the corresponding element of `y1` is larger than 5.
d. How many elements from `y1` has a value larger than 5?

6.2 Creating sequences

a. Create a vector `x` with elements (1, 2, 3, . . . , 100)
b. Create a vector `y` with elements (0, 5, 10, 15, . . . , 500)
c. Create a vector `z1` with elements (1, 1, 1, 2, 2, 2, . . . , 50, 50, 50)
d. Create a vector `z2` with elements (1, 2, 2, 3, 3, 3, . . . , 10)
e. Create a vector `z3` with elements (1, 2, 2, 3, 4, 4, 5, 6, 6, . . . , 50, 50)

6.3 Matrix creation

a. Create a vector with 100 random normal numbers and use that to generate a 10 by 10 matrix. Call this matrix `mat1`.
Hint: to generate a random normal vector, use the function `rnorm()`.
b. Add an extra row to `mat1` with the numbers 1 to 10 which will be the new first row. Also add the row with numbers 10 to 1 which will be the last row.
c. Add an extra column to the matrix obtained in step b with the number 1 to 12 (as first column).

6.4 Working with data frames

`tips` (reshape package) One waiter recorded information about each tip he received over a period of a few months working in one restaurant. He collected several variables:

- tip in dollars
- bill in dollars
- sex of the bill payer
- whether there were smokers in the party
- day of the week
- time of the day
- size of the party

In all he recorded 244 tips.

a. Install and load the package `reshape`.
b. Check the data description of this data frame.
c. Ask for the names of the variables in this data frame.
d. Take a subset of data tips which contains the observations from 1 until 20 and only the variables `tip`, `sex` and `day`.