# WarmingUp

BWI Specialisiation Data science

MS

8 9 2021

## Contents



# 1 Overview

- IDE
- Basic data structure in R
- Advanced Data Structures in R
  - `data.frame()`
- Loops etc.

# 2 IDE

Insert Chunks

- Strg + Alt + I

Execute Chunk

- Strg + Enter (Line of code in chunk)
- Strg + Shift + Enter (Execute whole chunk)

Menu Help

- Cheatsheets (Help -> Cheat Sheets -> RStudio IDE Cheat Sheet)

Comment Out/In code or text

- Strg + Shift + C

Keyboard Shortcut Help

- Alt + Shift + K

Insert History (Tab History) into Code Chunk

- Insert to Source

```
vec_a = c(1,2,3)
vec_a
```

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

Many Tabs (lower right)

- File
- Packages
- Help

```
#?mean
#?str #Commented because its really going on my nerves!
```

Environment tab

- Shows all defined variables in the code
- available packages

# 3 Basic data structures

Some Different basic data types in R are

- integers
- doubles
- strings

data is organized in Vectors

```
vector_zahlen = c(1,2,5,7) #NOTE Chunk Option result='hold'
vector_zahlen
is.double(vector_zahlen)
is.integer(vector_zahlen)
```

```
## [1] 1 2 5 7
## [1] TRUE
## [1] FALSE
```

Get integer Values:

```
vector_zahlen = c(1L,2L,5L,7L) #NOTE Force Integer Values
vector_zahlen
is.double(vector_zahlen)
is.integer(vector_zahlen)
```

```
## [1] 1 2 5 7
## [1] FALSE
## [1] TRUE
```

Coersion of Basic data types

- Vector of basic types can only have one type -> coersion

```
vector_mixed <- c(1.0, 1L, "Data Science", TRUE)
vector_mixed
```

```
## [1] "1"            "1"            "Data Science" "TRUE"
```

- all coersed to character

Creating vectors shortcuts to c() combine function

```
c(1,2,3,4,5)
1:5
```

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

```
seq(1,10,2) # NOTE step size = 2
seq(from = 1,to = 10, by = 2)
seq.int(1,10,2)
```

```
## [1] 1 3 5 7 9
## [1] 1 3 5 7 9
## [1] 1 3 5 7 9
```

```
is.integer(seq(1,10,2))
is.integer(seq(from = 1,to = 10, by = 2))
vector_integer <- as.integer(seq.int(1,10,2))
is.integer(as.integer(seq.int(1,10,2)))
```

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

Replicate Elements of a vector or list

```
# ?rep
rep(vector_integer, 3)
rep(vector_integer, each=3)
```

```
##  [1] 1 3 5 7 9 1 3 5 7 9 1 3 5 7 9
##  [1] 1 1 1 3 3 3 5 5 5 7 7 7 9 9 9
```

## 3.1 Indexing Data

```
vector_zahlen[1:2]
vector_zahlen[c(1,2,4)]
```

```
## [1] 1 2
## [1] 1 2 7
```

## 3.2 Named Vectors

```
vector_names <- c(zahl1 = 12L, zahl2 = 1212)
vector_names
```

```
## zahl1 zahl2
##    12  1212
```

Indexing Named Vector

```
vector_names[1] # NOTE Call Vector by Position
vector_names['zahl1'] # NOTE If Vector Elements have name you can call elements with name
```

```
## zahl1
##    12
## zahl1
##    12
```

Operations on vectors are vectorized

```
vector_integer * 1000 # NOTE Operation on all elements
```

```
## [1] 1000 3000 5000 7000 9000
```

```
vector_names <- c("Ben", "David", "Joe")
vector_names
paste(vector_names, "is a friend of mine")
```

```
## [1] "Ben"   "David" "Joe"
## [1] "Ben is a friend of mine"   "David is a friend of mine"
## [3] "Joe is a friend of mine"
```

Works on INT and Char vectors

# 4 List

**A List is a generic vector:** - a list can contain different types (no coersion)

```
l <- list(1:3, "foo", 3.0, list(2, "bar"))
l
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "foo"
##
## [[3]]
## [1] 3
##
## [[4]]
## [[4]][[1]]
## [1] 2
##
## [[4]][[2]]
## [1] "bar"
```

- `[[]]` indication for a list

## 4.1 Indexing a list

- is a generic vector
- can be indexed like a vector

```
l[1:2]
l[c(1,2)]
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "foo"
##
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] "foo"
```

Get the content of a list entry

```
l[1] # NOTE a list
l[[1]] # NOTE a vector
is.list(l[1])
is.list(l[[1]])
```

```
## [[1]]
## [1] 1 2 3
##
## [1] 1 2 3
## [1] TRUE
## [1] FALSE
```

- function `mean()` requires a numeric vector

```
mean(l[[1]]) # NOTE Works
try(
mean(l[1]) # NOTE Leads to error
)
```

```
## Warning in mean.default(l[1]): Argument ist weder numerisch noch boolesch: gebe
## NA zurück
```

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

## 4.2 Named List

```r
l_named <- list(a = 1:10, b = letters[1:10])
```

Indexing Named List:

```r
# Extract element a with the numbers 1:10
l_named[[1]]
l_named$a # NOTE $ means a list -> doesn't work for a vector
l_named[["a"]]
```

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

```r
#Returns a list
l_named["a"]
l_named[1]
```

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

# 5 Attributes

- Attributes add additional informations to an R object

```r
vector_with_attribute <- 1:5
attr(vector_with_attribute, "creator") = "MyNameIsSlimShady"
attr(vector_with_attribute, "date") = "08.09.2021"
vector_with_attribute
```

```
## [1] 1 2 3 4 5
## attr(,"creator")
## [1] "MyNameIsSlimShady"
## attr(,"date")
## [1] "08.09.2021"
```

Show all Attributes of an Object with:

```r
attributes(vector_with_attribute)
attributes(vector_with_attribute)$creator
```

```
## $creator
## [1] "MyNameIsSlimShady"
##
## $date
## [1] "08.09.2021"
##
## [1] "MyNameIsSlimShady"
```

## 5.1 Zwei Spezielle Attribute

- dim
- class

```
x = 1:8
x
```

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

Set dim attribute

```
dim(x) <- c(2,4)
x
```

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

Create 3 Dimensional Array

```
dim(x) <- c(2,2,2)
x
```

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

Set Class Attribute

```
class(x) = "myclass"
attributes(x)
```

```
## $dim
## [1] 2 2 2
##
## $class
## [1] "myclass"
```

- Dim and class are special attributes because R works differently when these are set, they are frequently employed attributes

# 6 generic functions

- Work with the class attribute
- Use “myclass” from above

```
print.myclass <- function(x){
  cat("array dimension", dim(x), "\n")
}
```

Print an Object of class type “myclass”

```
x
```

```
## array dimension 2 2 2
```

# 7 Data Frame

- a list of named vectors
- all vectors have the same length
- an attribute for rows is set `row.names`

```
df <- data.frame(Name = c("Andi", "Sepp", "Horst"), Alter = c(23L,25L,38L))
df
```

```
##    Name Alter
## 1  Andi    23
## 2  Sepp    25
## 3 Horst    38
```

Inspect df element

```
attributes(df)
```

```
## $names
## [1] "Name"  "Alter"
##
## $class
## [1] "data.frame"
##
## $row.names
## [1] 1 2 3
```

Unclass df

```
df
unclass(df)
```

```
##    Name Alter
## 1  Andi    23
## 2  Sepp    25
## 3 Horst    38
## $Name
## [1] "Andi"  "Sepp"  "Horst"
##
## $Alter
## [1] 23 25 38
##
## attr(,"row.names")
## [1] 1 2 3
```

# 8 Loops and [ls}apply

Instead of iterationg through a loop use `lapply` to apply a function to each element of the list.

```
lapply(df, summary)
```

```
## $Name
##    Length     Class      Mode
##         3 character character
##
## $Alter
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   23.00   24.00   25.00   28.67   31.50   38.00
```

```
lapply(df, length)
```

```
## $Name
## [1] 3
##
## $Alter
## [1] 3
```

Simplify output (instead of a list a vector)

```
sapply(df, length)
```

```
##  Name Alter
##     3     3
```