Advanced indexing techniques

Lukáš Lehotský and Petr Ocelík

Selecting: operators

Selection type	**Applicable to**	**Operator**
integer/logical	vector, matrix, df, list	**[** index **]**
integer/logical	vector, matrix, df, list	**[[** index **]]**
variable name	vector, matrix, df, list	**"**name**"**
variable name	df, list	**$**name
variable name	df, list	**@**name
special operator	vector, matrix, df, list	**%in%**

Selecting: indexes

- **[ ]** accesses an object's internal structure
  - Accesses whole **"data container"**
  - Particular data elements in case of vectors, matrices and data frames, accesses
  - Particular object with its wrapper in case of lists
- **[[ ]]** accesses a nested **"single item"** in the internal data structure
  - Access to one item in the object's internal structure
  - Useful to access objects within **lists**

Selecting: indexes

```
vect <- c(2, 6, 9)
str(vect)
num [1:3] 2 6 9
```

Selecting: indexes

```
vect <- c(2, 6, 9)
str(vect)
num [1:3] 2 6 9

vect[3]
[1] 9

vect[[3]]
[1] 9
```

Selecting: indexes

```
vect <- c(2, 6, 9)
str(vect)
num [1:3] 2 6 9

vect[3]
[1] 9

vect[[3]]
[1] 9

vect[1:3]
[1] 2 6 9

vect[[1:3]]
Error in v[[1:3]] ...
```

Selecting: indexes of nested objects – a list

```
ls <- list(c(0,1,2,3),
           c("car","bike"),
           "single object")
```

Selecting: indexes of nested objects – a list

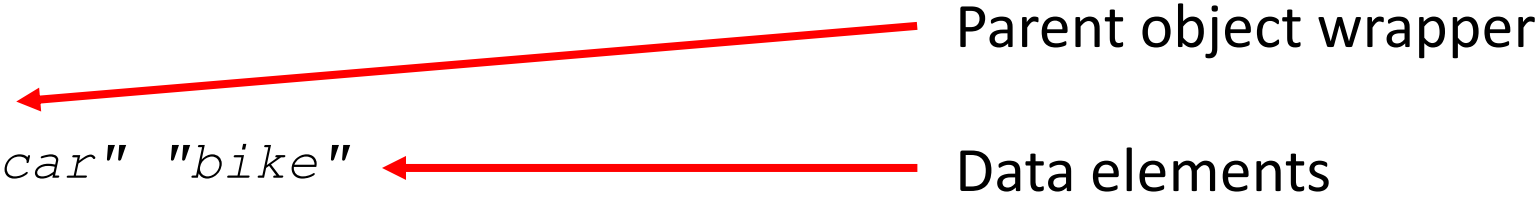
```
ls <- list(c(0,1,2,3),
           c("car","bike"),
           "single object")

ls[2]
[[1]]
[1] "car" "bike"
```

Selecting: indexes of nested objects – a list

```
ls <- list(c(0,1,2,3),
           c("car","bike"),
           "single object")

ls[2]
[[1]]
[1] "car" "bike"
```



Selecting: indexes of nested objects – a list

```
ls <- list(c(0,1,2,3),
           c("car","bike"),
           "single object")

ls[2]
[[1]]
[1] "car" "bike"

ls[[2]]
[1] "car" "bike"
```

Data elements only

Selecting: indexes of nested objects – a list

```
ls <- list(c(0,1,2,3),
           c("car","bike"),
           "single object")

ls[2]
[[1]]
[1] "car" "bike"

ls[[2]]
[1] "car" "bike"

ls[[2]][2]
[1] "bike"
```

Single element

Selecting: indexes of nested objects – a list

```
ls[2:3]
[[1]]
[1] "car" "bike"

[[2]]
[1] "single object"

ls[[2:3]]
Error in ls[[2:3]] : subscript out of bounds
```

Advanced use of indexes: vectors

- Indexing accepts **any result** that provides either **numeric indexes** or **logical values**
  - Existing objects containing index information
  - Vectors of TRUE/FALSE values from logical evaluations
  - Functions generating index information/providing logical evaluations
- Useful to **subset** data

Advanced use of indexes: logical statement

- A logical test applied to a vector will **create a vector of logical values** (TRUE/FALSE)
- Such vector may serve as an index

```
vect <- c(2, 6, 9)

vect == 9
[1] FALSE FALSE TRUE

index <- vect == 9

vect[index]
[1] 9
```

Advanced use of indexes: logical operators

Operator	Description
<	Left is smaller than right
>	Left is larger than right
<=	Left is smaller or equal than right
>=	Left is larger or equal than right
==	Left is equal than right
!=	Left is not equal than right
!	Negation
&	AND – allows test combinations, **all** logical statements must be true
|	OR – allows test combinations, **at least one** statement must be true

Source: Adler, J. (2012). R in a nutshell. Pp. 86.

Advanced use of indexes: function results

- Most basic use case – locating **missing values** on variables
- We can get either rows with missing data or contrary, get rid of them
- Function `is.na()`
  - Logical test on **presence/absence** of “NA” value
  - Returns **vector of logical values** TRUE/FALSE

```
vect.na <- c(1,0,1,2,2,NA,NA,2,1)
```

Advanced use of indexes: function results

```
vect.na <- c(1,0,1,2,2,NA,NA,2,1)

is.na(vect.na)
[1] FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE
```

Advanced use of indexes: function results

```
vect.na <- c(1,0,1,2,2,NA,NA,2,1)

is.na(vect.na)
[1] FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE

index <- is.na(vect.na)

vect.na[index]
[1] NA NA
```

Advanced use of indexes: function results

```
vect.na <- c(1,0,1,2,2,NA,NA,2,1)

is.na(vect.na)
[1] FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE

index <- is.na(vect.na)

vect.na[index]
[1] NA NA
```

?

Advanced use of indexes: function results

```
vect.na <- c(1,0,1,2,2,NA,NA,2,1)

is.na(vect.na)
[1] FALSE FALSE FALSE FALSE FALSE TRUE TRUE FALSE FALSE

index <- is.na(vect.na)

vect.na[index]
[1] NA NA

Index.o1 <- !is.na(vect.na) # option 1

Index.o2 <- is.na(vect.na) == FALSE # option 2

vect.na[index.o1]
[1] 1 0 1 2 2 2 1
```

Advanced use of indexes: combining tests

```
index <- !is.na(vect.na) & vect.na >= 1.5

vect.na[index]
[1] 2 2 2
```

Advanced use of indexes: combining tests

```
index <- !is.na(vect.na) & vect.na >= 1.5

vect.na[index]
[1] 2 2 2

index <- is.na(vect.na) | vect.na >= 1.5

vect.na[index]
[1] 2 2 NA NA 2
```

Advanced use of indexes: logical tests on data frames

- Logical tests may be **used to filter** data frames
- `TRUE`/`FALSE` statements index **row dimension** (unless specifically intended to subset columns)

	cars	*type*	*price*	*consumption*
1	*BMW*	*3*	*1200000*	*6.2*
2	*Audi*	*A4*	*1164000*	*5.9*
3	*VW*	*Passat*	*950500*	*5.9*

Advanced use of indexes: logical tests on data frames

- Problem at hand – **filter the data** set by cars costing **more than 1 000 000 units**
  - Select column containing price
  - Find values over 1 000 000
  - Use the result of logical test to filter the data frame

	cars	type	price	consumption
1	BMW	3	1200000	6.2
2	Audi	A4	1164000	5.9
3	VW	Passat	950500	5.9

Advanced use of indexes: logical tests on data frames

- Select column containing price
  - We know it is the **third** column
  - We may use **index** to **extract the third column**
  - We get **vector** of the column price (downgrade as default behavior)

```
df[ ,3]
[1] 1200000 1164000 950500
```

Advanced use of indexes: logical tests on data frames

- Use logical function to evaluate the car price
  - We use the indexed column and **add logical evaluation**
  - The result of evaluation is a **vector of logical TRUE/FALSE values**

```
df[ ,3]
[1] 1200000 1164000 950500

df[ ,3] > 1000000
[1]   TRUE TRUE FALSE
```

Advanced use of indexes: logical tests on data frames

- The vector **provides information over rows**

```
df[ ,3] > 1000000
[1]   TRUE TRUE FALSE

df
      cars   type        price       consumption
1     BMW    3           1200000     6.2
2     Audi   A4          1164000     5.9
3     VW     Passat      950500      5.9

df[ ,3,drop = FALSE] > 1000000 # just a demonstration
      price
[1,]  TRUE
[2,]  TRUE
[3,]  FALSE
```

Advanced use of indexes: logical tests on data frames

- The vector **provides information over rows**

```
df[ ,3] > 1000000
[1]    TRUE TRUE FALSE

df
      cars   type          price        consumption
1     BMW    3             1200000      6.2
2     Audi   A4            1164000      5.9
3     VW     Passat        950500       5.9

df[ ,3,drop = FALSE] > 1000000 # just a demonstration
      price
[1,]  TRUE
[2,]  TRUE
[3,]  FALSE
```

Advanced use of indexes: logical tests on data frames

- The vector **provides information over rows**

```
df[ ,3] > 1000000
[1]    TRUE  TRUE  FALSE

df
      cars   type          price        consumption
1     BMW    3             1200000      6.2
2     Audi   A4            1164000      5.9
3     VW     Passat        950500       5.9

df[ ,3,drop = FALSE] > 1000000 # just a demonstration
      price
[1,]  TRUE
[2,]  TRUE
[3,]  FALSE
```

Advanced use of indexes: logical tests on data frames

- The vector **provides information over rows**

```
df[ ,3] > 1000000
[1]    TRUE TRUE FALSE

df
      cars   type          price        consumption
1     BMW    3             1200000      6.2
2     Audi   A4            1164000      5.9
3     VW     Passat        950500       5.9

df[ ,3,drop = FALSE] > 1000000 # just a demonstration
      price
[1,]  TRUE
[2,]  TRUE
[3,]  FALSE
```

Advanced use of indexes: logical tests on data frames

- Use the result of logical test to create the condition for data-frame filtering
- **The condition thus applies to rows**

```
df[ ,3] > 1000000
[1] TRUE TRUE FALSE

condition <- df[ ,3] > 1000000

df[ condition ,  ]
      cars  type        price       consumption
1     BMW   3           1200000     6.2
2     Audi  A4          1164000     5.9
```

Advanced use of indexes: logical tests on data frames

- Use the result of logical test to create the condition for data-frame filtering
- **The condition applies to rows**

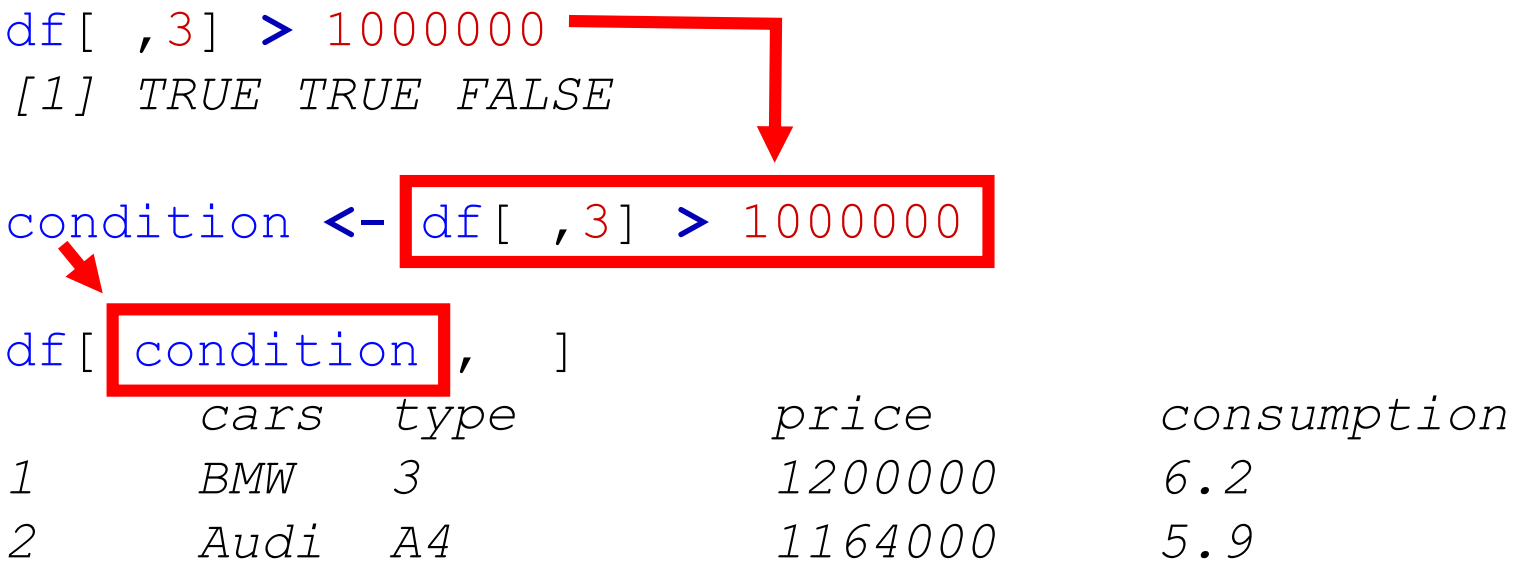
```
df[ ,3] > 1000000
[1] TRUE TRUE FALSE

condition <- df[ ,3] > 1000000

df[ condition ,  ]
      cars  type        price       consumption
1     BMW   3           1200000     6.2
2     Audi  A4          1164000     5.9
```

Advanced use of indexes: logical tests on data frames

- Use the result of logical test to create the condition for data-frame filtering
- **The condition applies to rows**



Advanced use of indexes: logical tests on data frames

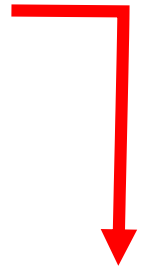
- The filtered data frame needs to be **saved to environment**

```
df[ ,3] > 1000000
[1] TRUE TRUE FALSE

condition <- df[ ,3] > 1000000

df[ condition ,  ]
      cars  type       price      consumption
1     BMW   3          1200000    6.2
2     Audi  A4         1164000    5.9

df.sub <- df[ condition ,  ]
```



Advanced use of indexes: logical tests on data frames

- **Alternative** – use $ to call a variable
  - Works only when **variables have names**

```
df$price > 1000000
[1] TRUE TRUE FALSE

condition <- df$price > 1000000

df[ condition ,  ]
      cars  type        price       consumption
1     BMW   3           1200000     6.2
2     Audi  A4          1164000     5.9

df.sub <- df[ condition ,  ]
```

Advanced use of indexes: logical tests on data frames

- **Alternative** – use the filter directly in the square brackets without a dedicated object
- There's a high **risk of getting it wrong**

```
df[ df$price > 1000000 ,  ]
      cars  type        price       consumption
1     BMW   3           1200000     6.2
2     Audi  A4          1164000     5.9

df[ df[ ,3] > 1000000 ,  ]
      cars  type        price       consumption
1     BMW   3           1200000     6.2
2     Audi  A4          1164000     5.9
```

Advanced use of indexes: logical tests on data frames

- **Combinations** of filters are possible

```
df[ ,3] > 1000000
[1] TRUE TRUE FALSE

condition <- df[ ,3] > 1000000

df[ condition , 1 ]
[1] BMW Audi

df[ condition , "consumption" ]
[1] 6.2 6.2
```

Advanced use of indexes: logical tests on data frames

- Problem at hand – **select data with two or more conditions**
  - Select cars which are BMW or Audi
- Straightforward approach does not work

```
df$cars == c( "BMW" , "Audi" )

Warning message:
In df$cars == c("BMW", "Audi") :  longer object
length is not a multiple of shorter object length
```

Advanced use of indexes: logical tests on data frames

- Problem at hand – **select data with two or more conditions**
  - Select cars which are BMW or Audi
- Approach using **logical operators**
  - Operator **AND** (“**&**”) – **all** conditions must be true at once
  - Operator **OR** (“**|**”) – **at least one** condition must be true

```
df$cars == "BMW" | df$cars == "Audi"
[1] TRUE TRUE FALSE

condition <- df$cars == "BMW" | df$cars == "Audi"

df[condition,  ]
      cars  type        price       consumption
1     BMW   3           1200000     6.2
2     Audi  A4          1164000     5.9
```

Advanced use of indexes: logical tests on data frames

- Problem at hand – **select data with two or more conditions**
  - Select cars which are BMW or Audi
- **Alternative** – use **special operator "%in%"**
  - **Counterintuitive** syntax - left %in% right means **left contains right**

```
df$cars %in% c( "BMW" , "Audi" )
[1] TRUE TRUE FALSE

condition <- df$cars %in% c( "BMW" , "Audi" )

df[condition,  ]
      cars  type        price       consumption
1     BMW   3           1200000     6.2
2     Audi  A4          1164000     5.9
```

Advanced use of indexes: ordering data frame

- Ordering a data frame is the most common use case of this logic
- Function `order()` provides an **ordered indexes of rows**
- Problem at hand
  - **Order the data frame** alphabetically by car make

	cars	*type*	*price*	*consumption*
1	*BMW*	*3*	*1200000*	*6.2*
2	*Audi*	*A4*	*1164000*	*5.9*
3	*VW*	*Passat*	*950500*	*5.9*

Advanced use of indexes: ordering data frame

```
order(df$cars)
[1] 2 1 3

condition <- order(df$cars)

df[ condition ,  ]
      cars  type        price       consumption
2     Audi  A4          1164000     5.9
1     BMW   3           1200000     6.2
3     VW    Passat      950500      5.9

df.ord <- df[ condition ,  ]
```

If statistics programs/languages were cars...

Practice 1

- Install and load the package “poliscidata” (or download it from the IS)
- Create a new object containing the dataset “world”
- Extract countries into separate vector
- Extract the Czech Republic row from the data frame
- Extract all V4 countries (CZ, SK, HU, PL) as a subset of the world dataset
- Extract only freedom indicators for V4 countries (all column names starting with “free_” – **manual index numbers**)

Practice 2

- Load dataset “states” from the “poliscidata” package into your environment (or download it from the IS)
- Order the dataset according to Obama 2012 election results (highest to lowest)
- Subset states on variable “gay_policy” – extract only states which are deemed as “liberal”
- Subset states on variable “gay_policy” and “secularism3” – extract only states which are deemed as “liberal”, but are neither deemed “secular” nor “religious”
- Extract only names of the states from the previous step
- Find the country which has missing (“NA”) value in the “secularism_3” variable