Applied Time Series

Oxana Malakhovskaya, NRU HSE

September 6, 2019

Naming a vector

- To give names to the elements of a vector we use the `names()` function.

```
food_expenses <- c(1400, 510, 280, 315, 640)
days_vector <- c("Monday", "Tuesday", "Wednesday",
"Thursday", "Friday")
names(food_expenses) <- days_vector
```

- If the elements of a vector are named, we can select or remove elements using their names and not only their indices.

```
  food_expenses["Tuesday"]
```

```
## Tuesday
##     510
```

Matrices: general information

- Matrices are rectangular datasets that can contain elements of the same data type (numeric, character, logical) but not a mixture of them.
- A matrix is two-dimensional (contrary to an array) and has a fixed number of rows and columns.
- To create a matrix we use the `matrix()` function .

Matrices: examples

```
A <- matrix(1:9, nrow = 3)
A
```

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

```
B <- matrix(1:9, nrow = 3, byrow = TRUE)
B
```

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

Matrices: names of the rows and columns

- Rows and columns of a matrix can be named.

```
# Survived adult passengers of Titanic
first_class <- c(57, 140)
second_class <- c(14, 80)
third_class <- c(75, 76)
crew <- c(192, 20)

# Construct the matrix
adults_survived <- matrix(c(first_class, second_class,
            third_class, crew), nrow = 4, byrow = TRUE)

# Vectors gender and class, used for naming
gender <- c("male", "female")
class <- c("1st class", "2nd class", "3rd class", "crew" )
```

Matrices: names of the rows and columns(2)

```
# Name the columns with gender
colnames(adults_survived) <- gender
# Name the rows with class
rownames(adults_survived) <- class

# Print out adults_survived matrix
adults_survived
```

```
##              male female
## 1st class      57    140
## 2nd class      14     80
## 3rd class      75     76
## crew          192     20
```

Matrices: names of the rows and columns(3)

- We can go an alternative way to create a matrix with row and columns names using an option `dimnames`.

```
victims <- c(118, 4, 154, 13, 387, 89, 670, 3)
adults_dead <- matrix(victims, nrow = 4, byrow = TRUE,
            dimnames = list(c("1st class", "2nd class",
            "3rd class","crew" ), c("male", "female")))
adults_dead
```

```
##             male female
## 1st class    118      4
## 2nd class    154     13
## 3rd class    387     89
## crew         670      3
```

Matrices: element summation

- To calculate the totals for each row of a matrix we can use the `rowSums()` function . The sum for each column of a matrix can be found with the `colSums` function .

```
both_genders_survived <- rowSums(adults_survived)
both_genders_survived
```

```
## 1st class 2nd class 3rd class      crew
##       197        94       151       212
```

```
all_classes_survived <- colSums(adults_survived)
all_classes_survived
```

```
##   male female
##    338    316
```

Matrices: element selection

- Similar to vectors, we use square brackets [] to select elements of a matrix. But contrary to vectors which are one-dimensional, matrices are two-dimensional and require double indices for element(s) selection.

```
 # Select the element in the 3rd row and 2nd column
adults_survived[3,2]
```

```
## [1] 76
```

```
# Select the elements in the first two rows and 1st column
adults_survived[1:2,1]
```

```
## 1st class 2nd class
##        57        14
```

Matrices: element selection(2)

```
# Select all elements in the first column
adults_survived[ ,1]
```

```
## 1st class 2nd class 3rd class      crew
##        57        14        75       192
```

Matrices: arithmetic operations

- Similar to vectors, the standard operators `+`, `-`, `*`, `/` work on matrices on element-by-element basis.

```
all_passangers <- adults_survived + adults_dead
all_passangers
```

```
##             male female
## 1st class   175    144
## 2nd class   168     93
## 3rd class   462    165
## crew        862     23
```

Matrices: horizontal concatenation

- To concatenate matrices horizontally (or column-wise) we use the `cbind()` function.

```
cbind(adults_survived, adults_dead)
```

```
##             male female male female
## 1st class     57    140  118      4
## 2nd class     14     80  154     13
## 3rd class     75     76  387     89
## crew         192     20  670      3
```

Matrices: vertical concatenation

- To concatenate matrices vertically (or row-wise) we use the `rbind()` function.

```
rbind(adults_survived, adults_dead)
```

```
##              male female
## 1st class      57    140
## 2nd class      14     80
## 3rd class      75     76
## crew          192     20
## 1st class     118      4
## 2nd class     154     13
## 3rd class     387     89
## crew          670      3
```

Some frequent matrix operations

Some the most frequent matrix operations are:

```
t(A) # returns a transpose of a matrix A
solve(A) # returns an inverse of
          # a squared nonsingular matrix A
A%*%B # returns the result of a matrix multiplication
diag(n) # returns a n-by-n identity matrix
         #(if n is scalar)
diag(A) # returns the main  diagonal of A
         #(if A is a square matrix)
```

Arrays: general information

- Data structures similar to matrices but of any dimension are called arrays and can be created with the `array()` function.

```
A <- array(1:12, c(2,3,2))
A
A[1,1,1]
```

Arrays creation

```
A
```

```
## , , 1
##
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
##
## , , 2
##
##      [,1] [,2] [,3]
## [1,]    7    9   11
## [2,]    8   10   12
```

```
A[1,1,1]
```

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

Arrays: a vector transformation

- We can make an array from a vector by changing the dimensions with the `dim()` function.

```
a <- seq(from = 1, to = 50, length = 60)
dim(a) <- c(3,4,5)
str(a)
```

```
##  num [1:3, 1:4, 1:5] 1 1.83 2.66 3.49 4.32 ...
```

```
a[2,3,4]
```

```
## [1] 36.71186
```

Data frames: general information

- Data frame is a table of observations. Each row contains one observation. The columns of the data frame show the variables being observed and must have names.
- Contrary to matrices, data frames can contain elements of different types. Columns can contain numbers, strings or factors but not a mixture of them.
- To see several first rows of a data frame we use the `head()` function. To see several last rows we use the `tail()` function. In both cases the header of the data frame is shown as well.

Data frames: initialization

- We can create a data frame with the `data.frame()` function with the arguments being the variabes of the data frame.

```
# Define the variables
name <- c("Ivanov", "Petrova", "Vasechkin")
gender <- c("male", "female", "male")
height <- c(175, 164, 182)
weight <- c(83.5, 58.5, 92)
single <-c(TRUE, TRUE, FALSE)
# Create a data frame from the variables
people <- data.frame(name, gender, height, weight, single)
people
```

```
##         name gender height weight single
## 1     Ivanov   male    175   83.5   TRUE
## 2    Petrova female    164   58.5   TRUE
## 3 Vasechkin   male    182   92.0  FALSE
```

Data frames: the structure

To see the structure of the data frame we use the `str()` function. The output of this function contains:

- The total number of observations
- The total number of variables
- A full list of variables mames
- The data type of each variable
- The first observations of each variable

```
str(people)
```

```
## 'data.frame':    3 obs. of  5 variables:
##  $ name  : Factor w/ 3 levels "Ivanov","Petrova",..: 1 2
##  $ gender: Factor w/ 2 levels "female","male": 2 1 2
##  $ height: num  175 164 182
##  $ weight: num  83.5 58.5 92
##  $ single: logi  TRUE TRUE FALSE
```

Data frames: elements selection

- To select elements from a data frame we use [], similarly to what we do with vectors and matrices.

```
# Print out the height of Petrova (row 2, column 3)
people[2,3]
```

```
## [1] 164
```

```
# Print out all data for Ivanov (entire first row)
people[1,]
```

```
##     name gender height weight single
## 1 Ivanov   male    175   83.5   TRUE
```

Data frames: using names for columns

- An alternative to select columns (or some elements of the columns) is to use the columns' names.

```
# Print out the height of Ivanov and Petrova
people[1:2,"height"]
```

```
## [1] 175 164
```

```
# Print out the weight of all participants
people$weight
```

```
## [1] 83.5 58.5 92.0
```

Data frames: conditional selection

- A selection according to a certain criteria can be done in a convenient way

```
#To choose single people from the list
sing <- people$single
sing # is a logical vector
```

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

```
people_single <- people[sing,]
people_single
```

```
##      name gender height weight single
## 1  Ivanov   male    175   83.5   TRUE
## 2 Petrova female    164   58.5   TRUE
```

Data frames: conditional selection(2)

Here is another example of a conditional selection.

```
# To choose people taller than 170 cm
people_tall <- people[people$height > 170,]
people_tall
```

```
##         name gender height weight single
## 1     Ivanov   male    175   83.5   TRUE
## 3 Vasechkin   male    182   92.0  FALSE
```

Data frame: conditional selection(3)

An alternative way makes use the `subset()` function. The arguments indicate the name of the dataframe and the criteria.

```
subset(people, single == FALSE)
```

```
##         name gender height weight single
## 3 Vasechkin   male    182     92  FALSE
```

```
subset(people, weight > 90)
```

```
##         name gender height weight single
## 3 Vasechkin   male    182     92  FALSE
```

```
subset(people, gender == "female")
```

```
##      name gender height weight single
## 2 Petrova female    164   58.5   TRUE
```