# Factors, Data Frames, Data Tables and Tibbles

## 1. Factors

As we have noted, factors are variables in R which take on a limited number of different values; such variables are often referred to as categorical variables.

In a dataset, we can distinguish two types of variables: categorical and continuous.

In a categorical variable, the value is limited and usually based on a particular finite group. For example, a categorical variable can be countries, year, gender, occupation. A continuous variable, however, can take any values, from integer to decimal. For example, we can have the revenue, price of a share, etc.

#### 1.1 Factors Code Example

## Example   
# ---  
# Question: Lets create a vector v  
# ---  
# OUR CODE GOES BELOW  
#   
v <- c(1,3,5,8,2,1,3,5,3,5)  
  
# Then determine whether this vector is a factor  
# ---  
#  
is.factor(v)

## [1] FALSE

## Challenge   
# ---  
# Question: Calculate the categorical distribution as shown and figure out why the given output  
# ---  
# OUR CODE GOES BELOW   
#   
factor(v)

## [1] 1 3 5 8 2 1 3 5 3 5  
## Levels: 1 2 3 5 8

## Example   
# ---  
# Question: Assign factor v tox and print out x   
# ---  
# OUR CODE GOES BELOW   
#   
x <- factor(v)  
  
x

## [1] 1 3 5 8 2 1 3 5 3 5  
## Levels: 1 2 3 5 8

## Challenge   
# ---  
# Question: Determine whether x is a factor below.   
# Hint: Just like the way you did when you were finding out whether vector v is a factor  
# ---  
# OUR CODE GOES BELOW  
#   
is.factor(x)

## [1] TRUE

#### 1.2 Factors Code Example

## Example   
# ---  
# Question: First we create a vector as input, check whether its a factor,   
# apply the factor function to create a factor from the vector  
# ---  
# OUR CODE GOES BELOW  
#   
data <- c("East","West","East","North","North","East","West","West","West","East","North")  
  
# Then print out this vector  
data

## [1] "East" "West" "East" "North" "North" "East" "West" "West" "West"   
## [10] "East" "North"

# Now, check whether this is a factor  
is.factor(data)

## [1] FALSE

# Then, apply the factor function to create a factor from the vector  
factor\_data <- factor(data)   
  
# Then see our newly created factor  
factor\_data

## [1] East West East North North East West West West East North  
## Levels: East North West

# Check whether this is a factor  
is.factor(factor\_data)

## [1] TRUE

#### 1.3 Factors Code Example

# Example   
# ---  
# Creating a factor, determine and check the levels   
# ---  
# OUR CODE GOES BELOW  
#   
sex <- factor(c("male", "female", "female", "male"))  
  
# Determining the levels  
levels(sex)

## [1] "female" "male"

# Then checking the number of levels using nlevels()  
nlevels(sex)

## [1] 2

# Sometimes, the order of the factors does not matter, other times you might want to specify the order   
# because it is meaningful (e.g., “low”, “medium”, “high”) or it is required by particular type of analysis.   
# Additionally, specifying the order of the levels allows us to compare levels:  
  
food <- factor(c("low", "high", "medium", "high", "low", "medium", "high"))  
  
# then print out levels of food  
levels(food)

## [1] "high" "low" "medium"

## 2. Data Frames

A data frame is used for storing data tables. Unlike a matrix in data frame each column can contain different modes of data.

### Creating a Dataframe

#### 2.1 Creating a Dataframe Code Example

## Example   
# ---  
# Question: Lets create a data frame BMI  
# ---  
# OUR CODE GOES BELOW  
#   
BMI <- data.frame(   
 gender = c("Male", "Male","Female"),   
 height = c(152, 171.5, 165),   
 weight = c(81,93, 78),   
 Age = c(42,38,26)  
)   
  
# Then print it out below  
BMI

## gender height weight Age  
## 1 Male 152.0 81 42  
## 2 Male 171.5 93 38  
## 3 Female 165.0 78 26

## Challenge   
# ---  
# Question: Create a data frame family with column names Name, Age, Gender and Occupation.   
# Populate it with 5 your own family members.  
# ---  
# OUR CODE GOES BELOW  
#   
FAM <- data.frame(   
 Name = c("CLaire","Mr.Bean","The Rock", "Angelina Jolie","Serena Williams"),   
 Age = c("19","60","40","50","40"),   
 Gender = c("Female","Male","Male","Female","Female"),   
 Occupation = c("Student","Comedian","Weight lifter","Actress","Athlete")  
   
)   
  
FAM

## Name Age Gender Occupation  
## 1 CLaire 19 Female Student  
## 2 Mr.Bean 60 Male Comedian  
## 3 The Rock 40 Male Weight lifter  
## 4 Angelina Jolie 50 Female Actress  
## 5 Serena Williams 40 Female Athlete

### Selecting Elements From a DataFrame

#### 2.2 Selecting Elements From a DataFrame Code Example

## Example   
# ---  
# Question: Selecting elements from the BMI dataframe  
# ---  
# OUR CODE GOES BELOW  
#   
  
# selecting row 1   
BMI[1,]

## gender height weight Age  
## 1 Male 152 81 42

# selecting rows 1 to 2  
BMI[1:2, ]

## gender height weight Age  
## 1 Male 152.0 81 42  
## 2 Male 171.5 93 38

# selecting column 1   
BMI[,1]

## [1] "Male" "Male" "Female"

# selecting column 1 to 2  
BMI[,1:2 ]

## gender height  
## 1 Male 152.0  
## 2 Male 171.5  
## 3 Female 165.0

# selecting row 1 in column 2  
BMI[1,2]

## [1] 152

## Challenge   
# ---  
# Question: Select the column 2 from the BMI dataframe  
# ---  
# OUR CODE GOES BELOW  
#   
BMI[,2]

## [1] 152.0 171.5 165.0

## Challenge   
# ---  
# Question: Select the second and third members of your family   
# ---  
# OUR CODE GOES BELOW  
#   
FAM[2:3,]

## Name Age Gender Occupation  
## 2 Mr.Bean 60 Male Comedian  
## 3 The Rock 40 Male Weight lifter

### Sorting

#### 2.3 Sorting Code Example

## Example   
# ---  
# Question: Sort the BMI dataframe by using the order() function   
# ---  
#   
  
# Sort in ascending order by gender  
# ---  
#  
sorted\_by\_gender <- BMI[order(BMI$gender),]  
  
# Print out sorted\_by\_gender below  
# ---  
#  
sorted\_by\_gender

## gender height weight Age  
## 3 Female 165.0 78 26  
## 1 Male 152.0 81 42  
## 2 Male 171.5 93 38

# Sort in descending order by weight  
# ---  
#   
sorted\_by\_weight <- BMI[order(-BMI$weight),]  
  
# Print out sorted\_by\_weight below  
# ---  
#   
sorted\_by\_weight

## gender height weight Age  
## 2 Male 171.5 93 38  
## 1 Male 152.0 81 42  
## 3 Female 165.0 78 26

# And sort in descending order by gender below  
# ---  
# OUR CODE GOES BELOW  
#   
  
sorted\_by\_gender1 <- BMI[order(BMI$gender),]  
sorted\_by\_gender1

## gender height weight Age  
## 3 Female 165.0 78 26  
## 1 Male 152.0 81 42  
## 2 Male 171.5 93 38

## 3. Data Tables

As we have mentioned, a data table provides an enhanced version of data.frames.

### Creating a Data Table

The data.table R package is considered as the fastest package for data manipulation.

#### 3.2 Creating a Data Table Code Example

library(data.table)  
## Example   
# ---  
# Question: Create a data table DT  
# ---  
#   
DT = data.table(  
 ID = c("b","b","b","a","a","c"),  
 a = 1:6,  
 b = 7:12,  
 c = 13:18  
)  
  
# Load the data.table package  
# ---  
#   
  
  
# Then print it out to see what happens  
# ---  
# OUR CODE GOES BELOW  
#   
DT

## ID a b c  
## 1: b 1 7 13  
## 2: b 2 8 14  
## 3: b 3 9 15  
## 4: a 4 10 16  
## 5: a 5 11 17  
## 6: c 6 12 18

### Selecting Elements From a Data Table

#### 3.3 Selecting Elements From a Data Table Code Example

## Example   
# ---  
# Question: Select elements from the given datatable DT  
# ---  
# OUR CODE GOES BELOW  
#   
  
# Selecting Row 1   
DT[1,]

## ID a b c  
## 1: b 1 7 13

# Selecting Rows 1 to 2  
DT[1:2,]

## ID a b c  
## 1: b 1 7 13  
## 2: b 2 8 14

# Find out what happens when we print out the following statement   
DT[,1]

## ID  
## 1: b  
## 2: b  
## 3: b  
## 4: a  
## 5: a  
## 6: c

# Find out what happens when we print out the following statement  
DT[,1:2]

## ID a  
## 1: b 1  
## 2: b 2  
## 3: b 3  
## 4: a 4  
## 5: a 5  
## 6: c 6

# And lastly find out what happens when we print out the following statement  
DT[1,2]

## a  
## 1: 1

# Select the fourth and third rows from the data table  
# ---  
# OUR CODE GOES BELOW  
#

### Sorting a Data Table

#### 3.4 Sorting a Data Table Code Example

## Example   
# ---  
# Question: Sorting the datatable in ascending order by c   
# ---  
# OUR CODE GOES BELOW  
#   
  
# Performing the sort  
#   
sorted\_by\_c <- DT[order(DT$c),]  
  
# Printing out sorted\_by\_c  
# ---  
# OUR CODE GOES BELOW  
#   
  
# Sort in descending order by b, uncommenting the line below  
# ---  
#   
# sorted\_by\_b <- DT[order(-DT$b),]  
  
# Finally printing out sorted\_by\_b below  
# ---  
# OUR CODE GOES BELOW  
#

## 4. Tibbles

Tibbles are data frames, but they tweak some older behaviours to make life a little easier. They also have an enhanced print() method which makes them easier to use with large datasets containing complex objects

### Creating a Tibble

You can create a new tibble from individual vectors with tibble(). tibble() will automatically recycle inputs of length 1, and allows you to refer to variables that you just created, as shown below.

#### 4.1 Creating a Tibble Code Example

## Example   
# ---  
# Question: Create a tible tb  
# ---  
# OUR CODE GOES BELOW  
#   
  
# First, we load the tibble package  
library(tibble)  
  
# Then create our tibble tb  
tb <- tibble(  
 x = 1:5,   
 y = 1,   
 z = x ^ 2 + y  
)  
  
# And finally print the created tibble   
# ---  
# OUR CODE GOES BELOW  
#   
tb

## Warning: `...` is not empty.  
##   
## We detected these problematic arguments:  
## \* `needs\_dots`  
##   
## These dots only exist to allow future extensions and should be empty.  
## Did you misspecify an argument?

## # A tibble: 5 x 3  
## x y z  
## <int> <dbl> <dbl>  
## 1 1 1 2  
## 2 2 1 5  
## 3 3 1 10  
## 4 4 1 17  
## 5 5 1 26

### Selecting a Tibble Code Example

#### 4.1 Selecting a Tibble Code Example

## Example   
# ---  
# Question: Find out what happens when we print the following   
# ---  
# OUR CODE GOES BELOW  
#   
tb[1,]

## Warning: `...` is not empty.  
##   
## We detected these problematic arguments:  
## \* `needs\_dots`  
##   
## These dots only exist to allow future extensions and should be empty.  
## Did you misspecify an argument?

## # A tibble: 1 x 3  
## x y z  
## <int> <dbl> <dbl>  
## 1 1 1 2

tb[1:2, ]

## Warning: `...` is not empty.  
##   
## We detected these problematic arguments:  
## \* `needs\_dots`  
##   
## These dots only exist to allow future extensions and should be empty.  
## Did you misspecify an argument?

## # A tibble: 2 x 3  
## x y z  
## <int> <dbl> <dbl>  
## 1 1 1 2  
## 2 2 1 5

tb[,1]

## Warning: `...` is not empty.  
##   
## We detected these problematic arguments:  
## \* `needs\_dots`  
##   
## These dots only exist to allow future extensions and should be empty.  
## Did you misspecify an argument?

## # A tibble: 5 x 1  
## x  
## <int>  
## 1 1  
## 2 2  
## 3 3  
## 4 4  
## 5 5

tb[,1:2 ]

## Warning: `...` is not empty.  
##   
## We detected these problematic arguments:  
## \* `needs\_dots`  
##   
## These dots only exist to allow future extensions and should be empty.  
## Did you misspecify an argument?

## # A tibble: 5 x 2  
## x y  
## <int> <dbl>  
## 1 1 1  
## 2 2 1  
## 3 3 1  
## 4 4 1  
## 5 5 1

# Select the second and third rows  
# ---  
# OUR CODE GOES BELOW  
#   
tb[2:3,]

## Warning: `...` is not empty.  
##   
## We detected these problematic arguments:  
## \* `needs\_dots`  
##   
## These dots only exist to allow future extensions and should be empty.  
## Did you misspecify an argument?

## # A tibble: 2 x 3  
## x y z  
## <int> <dbl> <dbl>  
## 1 2 1 5  
## 2 3 1 10

### Sorting a Tibble

#### 4.1 Sorting a Tibble Code Example

## Example  
# ---  
# Question: Find out what happens when we sort by doing the following   
# ---  
#   
sorted\_by\_1 <- tb[order(tb$z),]  
sorted\_by\_1

## Warning: `...` is not empty.  
##   
## We detected these problematic arguments:  
## \* `needs\_dots`  
##   
## These dots only exist to allow future extensions and should be empty.  
## Did you misspecify an argument?

## # A tibble: 5 x 3  
## x y z  
## <int> <dbl> <dbl>  
## 1 1 1 2  
## 2 2 1 5  
## 3 3 1 10  
## 4 4 1 17  
## 5 5 1 26

sorted\_by\_2 <- tb[order(-tb$x),]  
sorted\_by\_2

## Warning: `...` is not empty.  
##   
## We detected these problematic arguments:  
## \* `needs\_dots`  
##   
## These dots only exist to allow future extensions and should be empty.  
## Did you misspecify an argument?

## # A tibble: 5 x 3  
## x y z  
## <int> <dbl> <dbl>  
## 1 5 1 26  
## 2 4 1 17  
## 3 3 1 10  
## 4 2 1 5  
## 5 1 1 2

# Sort tb in ascending order by x below  
# ---  
# OUR CODE GOES BELOW  
#   
sorted\_by\_3 <- tb[order(-tb$x),]

R Basics - Factors, Data Frames, Data Tables and Tibbles.md Displaying R Basics - Factors, Data Frames, Data Tables and Tibbles.md.