title : Metacharacters I description : In this chapter I will teach you about metacharacters attachments : slides\_link : <https://s3.amazonaws.com/assets.datacamp.com/course/teach/slides_example.pdf>

--- type:MultipleChoiceExercise lang:r xp:50 skills:1 key:ab57cc6e24 ##Wildcard .

The wildcard **.** represent any character in a regular expression. It means that **.** represents any character *that exists* in a given position. **Which of the following texts does not match with the regular expression */Hell./* ?**

\*\*\* =instructions - Hell - Hell is warm - Hello, how are you? - I love Hellene \*\*\* =hint Remember the wildcard match any character that *exists* \*\*\* =sct

#  
msg\_bad\_1 <- "In this instance the wildcard matches with a `space`"  
msg\_bad\_2 <- "In this instance the wildcard matches with an `o`"  
msg\_bad\_3 <- "In this instance the wildcard matches with an `e`"  
msg\_success <- "Exactly! In this intance, there is no character after Hell, therefore the regex doesn't match"  
  
  
test\_mc(correct = 1, feedback\_msgs = c(msg\_success, msg\_bad\_1, msg\_bad\_2, msg\_bad\_3 ))

--- type:NormalExercise lang:r xp:50 skills:1 key:452b676422 ##The escape I from metacharacters to literals:  
As I mentioned before the metacharacters have *special meaning*, however what if I want to give a *literal sense*, for instance a dot actually means dot, a question mark that really means a question mark. In such case you need to *escape* the metacharacter with the escape metacharacter or \. **In R and Java, you need two bashlashes or \\ to escape a metacharacter!!**

\*\*\* =instructions - I have loaded a movie titles vector from the movies database - Use the proper regex to know how many titles do have a dot in their titles? - Use the proper regex to know how many titles do have a question mark in their titles? - Use the proper regex to know how many titles do have a backslash in their titles?

\*\*\* =hint - Remember 2 things: in R you need a double backslash to escape a metacharacter!!

\*\*\* =pre\_exercise\_code

movies <- read.csv("http://s3.amazonaws.com/assets.datacamp.com/course/introduction\_to\_r/movies.csv", stringsAsFactors=F)  
titles <- movies$title  
rm(movies)

\*\*\* =sample\_code

#I have loaded a title vector of movie titles  
# Complete the \_\_\_ with the correct regex in order to answer the questions  
# Don't do any other change in the script!  
  
#How many movies contains a dot (.) in their titles  
regex.dot <- '\_\_\_\_'  
length(grep(regex.dot, titles))  
  
#How many movies contains a question mark (?) in their titles  
regex.quest <- '\_\_\_\_'  
length(grep(regex.quest, titles))  
  
#How many movies contains a backslash (\) in their titles  
regex.bl <- '\_\_\_\_'  
length(grep(regex.bl, titles))

\*\*\* =solution

#I have loaded a title vector of movie titles  
# Complete the \_\_\_ with the correct regex in order to answer the questions  
# Don't do any other change in the script!  
  
#How many movies contains a dot (.) in their titles  
regex.dot <- '\\.'  
length(grep(regex.dot, titles))  
  
#How many movies contains a question mark (?) in their titles  
regex.quest <- '\\?'  
length(grep(regex.quest, titles))  
  
#How many movies contains a backslash (\) in their titles  
regex.bl <- '\\\\'  
length(grep(regex.bl, titles))

\*\*\* =sct

test\_error()  
test\_object("regex.dot")  
test\_object("regex.quest")  
test\_object("regex.bl")  
  
success\_msg("Good work!")

--- type:NormalExercise lang:r xp:150 skills:1 key:452b67642c ##The escape II literals with special meaning:  
As I mentioned before the escape character \ eliminate the *special meaning* of the metacharacter it modifies. However in the other hand, it gives a special meaning to some literal characters. **In R and Java, you need two bashlashes or \\ to escape a metacharacter!!**

\*\*\* =instructions - I have loaded a movie titles vector from the movies database - You might need to answer some questions - Use the following list of literal characters modified by \ to solve the questions in the excercise: - \d means "any number" - \D means "any character but a number" - \w means "any printable character" - \W means "any non-printable character" - \s means "any white space character" - \S means "any non-white space character" - \b means "word boundary" - \t means "tab" - \n means "next line" - Use the proper regex to know how many titles do have a backslash in their titles?

\*\*\* =hint - Remember 2 things: in R you need a double backslash to escape a metacharacter!!

\*\*\* =pre\_exercise\_code

movies <- read.csv("http://s3.amazonaws.com/assets.datacamp.com/course/introduction\_to\_r/movies.csv", stringsAsFactors=F)  
titles <- movies$title  
rm(movies)

\*\*\* =sample\_code

#I have loaded a title vector of movie titles  
#It will be needed to answer some questions  
# Complete the \_\_\_ with the correct character in order to answer the questions  
# Never use w, D, s or S in any case!  
# Don't do any other change in the script!  
  
#Complete the following regex with the correct characters!  
regex.1 <- 'Your score is \\\_\_\\\_\_\\\_\_\\\_\_'  
grepl(regex.1, 'Your score is 67.8')  
  
#Complete the following regex with the correct characters!  
regex.2 <- "My name is Gill\\\_\_\\\_\_I am a very proficient computer scientist, don't you think so\\\_\_"  
grepl(regex.2, "My name is Gill.  
I am a very proficient computer scientist, don't you think so?")  
  
#How many movies contains a tab in their titles  
regex.tab <- '\_\_\_\_'  
length(grep(regex.tab, titles))  
  
#How many movies contains at least two consecutive numbers in their titles  
regex.numbers <- '\_\_\_\_'  
length(grep(regex.numbers, titles))

\*\*\* =solution

#I have loaded a title vector of movie titles  
#It will be needed to answer some questions  
# Complete the \_\_\_ with the correct character in order to answer the questions  
# Never use w, D, s or S in any case!  
# Don't do any other change in the script!  
  
#Complete the following regex with the correct characters!  
regex.1 <- 'Your score is \\d\\d\\.\\d'  
grepl(regex.1, 'Your score is 67.8')  
  
#Complete the following regex with the correct characters!  
regex.2 <- "My name is Gill\\.\\nI am a very proficient computer scientist, don't you think so\\?"  
grepl(regex.2, "My name is Gill.  
I am a very proficient computer scientist, don't you think so?")  
  
#How many movies contains a tab in their titles  
regex.tab <- '\\t'  
length(grep(regex.tab, titles))  
  
#How many movies contains at least two consecutive numbers in their titles  
regex.numbers <- '\\d\\d'  
length(grep(regex.numbers, titles))

\*\*\* =sct

test\_error()  
test\_object("regex.1")  
test\_object("regex.2")  
test\_object("regex.tab")  
test\_object("regex.numbers")  
  
success\_msg("Good work!")

--- type:MultipleChoiceExercise lang:r xp:50 skills:1 key:ab57cd6e24 ##Analyzing a Python file

You can read any kind of code or program using R. I have loaded a Python file in the working directory. Use the R function **list.files()** to show the python file in the working directory. Read the file using the R function **python <- readLines("python file.py")**. **Which of the following regular expressions (in R) does determine the attributes of a Python class?** \*\*\* =pre\_exercise\_code

python.file <- c( "import math",  
 "class gameObject:",  
 " def \_\_init\_\_(self, pos, radius):",  
 " self.pos=pos",  
 " self.radius=radius",  
  
 "def distance(Coord\_1,Coord\_2):",  
 ' if len(Coord\_1) != len(Coord\_2)',  
 ' return "The length of the vectors is not equal"',  
 " else:",  
 " l=[(Coord\_1[i]-Coord\_2[i])\*\*2 for i in range(len(Coord\_1))]",  
   
 " return math.sqrt(sum(l))",  
 "def collisions(x,y):",  
 " Coord\_x = x.pos",  
 " Coord\_y = y.pos",  
 " impact = x.radius + y.radius",  
 ' if distance(Coord\_x, Coord\_y) <= impact:',  
 ' return "boom"' )   
write.table(python.file, "python file.py",quote=F, col.names=F, row.names=F, sep="\n")  
rm(python.file)

\*\*\* =instructions - class - self. - self\\. - self\. \*\*\* =hint Remember the wildcard match any character that *exists* \*\*\* =sct

#  
msg\_bad\_1 <- "`class` makes no sense in this context"  
msg\_bad\_2 <- "`.` is a wildcard, it matches with anything"  
msg\_bad\_3 <- "`\\` in R is not the metacharacter for escape"  
msg\_success <- "Exactly! This is correct!"  
  
  
test\_mc(correct = 3, feedback\_msgs = c(msg\_bad\_1, msg\_bad\_2, msg\_success, msg\_bad\_3 ))

--- type:MultipleChoiceExercise lang:r xp:50 skills:1 key:0d507f6f0e ##Parentheses ()

The pair of parenthesis has **no meaning** in isolation. They have a special meaning, but we discuss that in another exercise. **Which of the following regex contains match the string "(a)"?**

\*\*\* =instructions - /(a/ - /\(a\)/ - (a) - /a)/ \*\*\* =hint Remember what you have learned about the escape metacharacter? \*\*\* =sct

# The sct section defines the Submission Correctness Tests (SCTs) used to  
# evaluate the student's response. All functions used here are defined in the   
# testwhat R package  
  
msg\_bad\_1 <- "This is an invalid regex! it lead to a runtime error"  
msg\_bad\_2 <- "That regex matches the string "a", not (a)'  
msg\_success <- "Exactly! the pair of paranthesis need to be escaped!"  
  
# Use test\_mc() to grade multiple choice exercises.   
# Pass the correct option (Action, option 2 in the instructions) to correct.  
# Pass the feedback messages, both positive and negative, to feedback\_msgs in the appropriate order.  
test\_mc(correct = 2, feedback\_msgs = c(msg\_bad\_1, msg\_success, msg\_bad\_2,msg\_bad\_1))

--- type:NormalExercise lang:r xp:50 skills:1 key:452b6764ff ##The alternation (|)

In a previous exercise we learned the *a set of parenthesis in isolation are meaningless*. The **alternation** represent a list of posibilities separated |

\*\*\* =instructions - The environment is populated with a character vector titles - Fill in the blank with the proper *regex*

\*\*\* =hint - Remember the literal values represent themselves

\*\*\* =pre\_exercise\_code

library(gtools)  
movies <- read.csv("http://s3.amazonaws.com/assets.datacamp.com/course/introduction\_to\_r/movies.csv", stringsAsFactors=F)  
titles<-movies$title  
rm(movies)

\*\*\* =sample\_code

#The environment is populated with the vector `titles`  
  
#Replace \_\_\_\_ below with the proper regular expression   
#in order to find out how many movies has the word "Time" in their title  
#Don't change anything else!  
regex <- "\_\_\_\_"  
length(grep(regex, titles,value=T))

\*\*\* =solution

#The environment is populated with the vector `titles`  
  
#Replace \_\_\_\_ below with the proper regular expression  
#in order to find out how many movies are the word "Time" in their title  
#Don't change anything else!  
regex <- "Time"  
length(grep(regex, titles,value=T))

\*\*\* =sct

regex.sln<-function(r){  
if (regex==r) {  
success\_msg("Good work!")  
}else{  
err\_msg <- paste('The regex `', regex, '` is incorrect', sep="")  
test\_an\_object("regex", undefined\_msg = err\_msg)  
}}  
regex.sln("Time")