# Tarea19

**Yimmy Eman**

2022-07-14

# Pregunta 1

Antes de usar stringr, muchas veces habréis usado las funciones paste() y paste0(). ¿Qué diferencia ambas funciones? A qué función de stringr son equivalentes? ¿Cómo gestionan internamente los NA?

paste() y paste0() se diferencian en la opción por defecto para separar los elementos que se concatenan. paste() deja un espacio; paste0() no.

```
paste("uno", "dos", "tres")
```

```
## [1] "uno dos tres"
```

```
paste0("uno", "dos", "tres")
```

```
## [1] "unodostres"
```

Estas funciones son equivalentes a str_c(). Esta función por defecto no agrega un espacio entre los elementos concatenados, por lo que si queremos incluirlo es necesario explicitarlo con el argumento sep =:

```
str_c("uno", "dos", "tres")
```

```
## [1] "unodostres"
```

```
str_c("uno", "dos", "tres", sep = " ")
```

```
## [1] "uno dos tres"
```

# Pregunta 2

- Describe la diferencia entre sep y collapse de la función str_c()

sep define el caracter que se inserta entre los elementos a concatenar. collapse, por su parte, es el caracter para combinar entre los elementos a concatenar para generar un vector de extensión 1

- Utiliza str_length() y str_sub() para extraer el carácter central de un string. ¿Qué harías si el string tiene un número par / impar de caracteres?

```
x <- c("uno", "dos", "tres")
largo <- str_length(x)
mitad <- ceiling(largo / 2)
str_sub(x, start = mitad, end = mitad)
```

```
## [1] "n" "o" "r"
```

- ¿Qué hace la función str_wrap() y cuándo podrías usarla?

str_wrap() permite formatear párrafos indicando el ancho en cantidad de caracteres que queremos que tenga. El valor por defecto es 80.

- ¿Qué hace la función str_trim() y cuál es su función opuesta?

str_trim() permite eliminar espacios en blanco al inicio y al final de una cadena de caracteres:

- Escribe una función que convierta un vector (“a”,“b”,“c”) en un string que diga “a, b y c”. Piensa en las posiciones en el caso de vectores de longitud 0, 1, y 2 especialmente.

# Pregunta 3

Explica por qué los siguientes strings no coinciden con “":"", “\” y “\"

- ¿Cómo localizarías la secuencia “'?

```
str_view("\"'\\", "\"'\\\\", match = TRUE)
```

```
"'\
```

- ¿Qué patrón encontraría la expresión regular .....? ¿Cómo lo representarías en formato string?

```
str_view(c(".a.b.c", ".a.b", "....."), c("\\..\\..\\.."), match = TRUE)
```

```
.a.b.c
```

- Cómo buscarías el carácter string “^”?

```
str_view(c("$^$", "ab$^$sfas"), "^\\$\\^\\$$", match = TRUE)
```

```
$^$
```

# Pregunta 4

A partir de las palabras dadas en stringr::words, escribe expresiones regulares para localizar palabras que:

1. Empiecen por “y”

```
str_view(words, "^y", match = T)
```

```
year
yes
yesterday
yet
you
young
```

2. Acaben por “x”

```
str_view(words, "x$", match = T)
```

```
box
sex
six
tax
```

3. Tengan exactamente tres letras (sin usar la función str_length())

```
str_view(words, "^...$", match = T)
```

```
act
```

```
add
age
ago
air
all
and
any
arm
art
ask
bad
bag
bar
bed
bet
big
bit
box
boy
bus
but
buy
can
car
cat
cup
cut
dad
day
die
dog
dry
due
eat
egg
end
eye
far
few
fit
fly
for
fun
```

```
gas
get
god
guy
hit
hot
how
job
key
kid
lad
law
lay
leg
let
lie
lot
low
man
may
mrs
new
non
not
now
odd
off
old
one
out
own
pay
per
put
red
rid
run
say
see
set
sex
she
sir
```

sit
six
son
sun
tax
tea
ten
the
tie
too
top
try
two
use
war
way
wee
who
why
win
yes
yet
you

4. Tengan siete o más letras. Como la lista es algo larga, intenta usar el parámetro match del str_view() para acotar y ver solo los resultados positivos

```
str_view(stringr::words, ".......", match = TRUE)
```

absolute
account
achieve
address
advertise
afternoon
against
already
alright
although
america
another
apparent
appoint
approach

```
appropriate
arrange
associate
authority
available
balance
because
believe
benefit
between
brilliant
britain
brother
business
certain
chairman
character
Christmas
colleague
collect
college
comment
committee
community
company
compare
complete
compute
concern
condition
consider
consult
contact
continue
contract
control
converse
correct
council
country
current
decision
definite
```

```
department
describe
develop
difference
difficult
discuss
district
document
economy
educate
electric
encourage
english
environment
especial
evening
evidence
example
exercise
expense
experience
explain
express
finance
fortune
forward
function
further
general
germany
goodbye
history
holiday
hospital
however
hundred
husband
identify
imagine
important
improve
include
increase
```

```
individual
industry
instead
interest
introduce
involve
kitchen
language
machine
meaning
measure
mention
million
minister
morning
necessary
obvious
occasion
operate
opportunity
organize
original
otherwise
paragraph
particular
pension
percent
perfect
perhaps
photograph
picture
politic
position
positive
possible
practise
prepare
present
pressure
presume
previous
private
probable
```

```
problem
proceed
process
produce
product
programme
project
propose
protect
provide
purpose
quality
quarter
question
realise
receive
recognize
recommend
relation
remember
represent
require
research
resource
respect
responsible
saturday
science
scotland
secretary
section
separate
serious
service
similar
situate
society
special
specific
standard
station
straight
strategy
```

`structure`

`student`

`subject`

`succeed`

`suggest`

`support`

`suppose`

`surprise`

`telephone`

`television`

`terrible`

`therefore`

`thirteen`

`thousand`

`through`

`thursday`

`together`

`tomorrow`

`tonight`

`traffic`

`transport`

`trouble`

`tuesday`

`understand`

`university`

`various`

`village`

`wednesday`

`welcome`

`whether`

`without`

`yesterday`

# Pregunta 5

Con el mismo dataset de stringr::words - Crea una expresión regular que se quede con las palabras que: 1. empiezan por vocal

```
head(str_subset(stringr::words, "^[aeiou]"),10)
```

```
##  [1] "a"        "able"     "about"    "absolute" "accept"   "account"
##  [7] "achieve"  "across"   "act"      "active"
```

```
tail(str_subset(stringr::words, "^[aeiou]"),10)
```

```
##  [1] "union"      "unit"       "unite"      "university" "unless"
##  [6] "until"      "up"         "upon"       "use"        "usual"
```

2. contengan solo consonantes

```
str_view(words, "^[^aeiou]+$", match = T)
```

by

dry

fly

mrs

try

why

3. Acaben con -ed (verbos en pasado) pero no en -eed 4. Acaben con -ing o -ise

```
str_view(words, "[^e]ed$", match = TRUE)
```

bed

hundred

red

```
str_view(words, "i(ng|se)$", match = TRUE)
```

advertise

bring

during

evening

exercise

king

meaning

morning

otherwise

practise

raise

realise

ring

rise

sing

surprise

thing

- ¿Verifíca la regla de gramática “i antes de e excepto si va después de c” de la gramática inglesa.

```
length(str_subset(stringr::words, "(cei|[^c]ie)"))
```

```
## [1] 14
```

```
length(str_subset(stringr::words, "(cie|[^c]ei)"))
```

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

- ¿Todas las palabras que tienen una q les sigue después una u?

```
str_view(words, "q[^u]", match = TRUE)
```

- Adapta mi código para escribir expresiones regulares que coincidan con teléfonos de tu región.
- Crea una expresión regular que se quede con las palabras que cumplan :

1. empiezan con tres consonantes

```
str_view(words, "^[^aeiou]{3}", match = TRUE)
```

Chr ist
Chr istmas
dry
fly
mrs
sch eme
sch ool
str aight
str ategy
str eet
str ike
str ong
str ucture
sys tem
thr ee
thr ough
thr ow
try
typ e
why

2. tienen tres o más vocales consecutivas

```
str_view(words, "[aeiou]{3,}", match = TRUE)
```

b eau ty
obv iou s
prev iou s
qu iet
ser iou s
var iou s

3. tienen dos o más pares de consonante-vocal seguidas

```
str_view(words, "([aeiou][^aeiou]){2,}", match = TRUE)
```

absolute
agent
along
america
another
apart
apparent
authority
available
aware
away
balance
basis
become
before
begin
behind
benefit
business
character
closes
community
consider
cover
debate
decide
decision
definite
department
depend
design
develop
difference
difficult
direct
divide
document
during
economy
educate

```
elect
electric
eleven
encourage
environment
europe
even
evening
ever
every
evidence
exact
example
exercise
exist
family
figure
final
finance
finish
friday
future
general
govern
holiday
honest
hospital
however
identify
imagine
individual
interest
introduce
item
jesus
level
likely
limit
local
major
manage
meaning
measure
```

minister
minus
minute
moment
money
music
nature
necessary
never
notice
okay
open
operate
opportunity
organize
original
over
paper
paragraph
parent
particular
photograph
police
policy
politic
position
positive
power
prepare
present
presume
private
probable
process
produce
product
project
proper
propose
protect
provide
quality
realise

```
reason
recent
recognize
recommend
record
reduce
refer
regard
relation
remember
report
represent
result
return
saturday
second
secretary
secure
separate
seven
similar
specific
strategy
student
stupid
telephone
television
therefore
thousand
today
together
tomorrow
tonight
total
toward
travel
unit
unite
university
upon
visit
water
woman
```

# Pregunta 6

Resuelve el crucigrama fácil de la web: https://regexcrossword.com/challenges/beginner/puzzles/1 (https://regexcrossword.com/challenges/beginner/puzzles/1)

# Pregunta 7

Describe las palabras que devolverá la expresión regular:

1. “(.)(.)(.).*\3\2\1”
2. “(.).\1.\1”
3. (..)\1
4. “(.)(.)\2\1”
5. (.)\1\1
6. “\\{4}”
7. --
8. “\{.+\}”
9. ^.*$

# Pregunta 8

Crea una expresión regular que se quede con las palabras que cumplan: - Empiezan y acaban con el mismo carácter

```
str_subset(words, "^(.)((.*\\1$)|\\1?$)")
```

```
##  [1] "a"         "america"    "area"       "dad"        "dead"
##  [6] "depend"    "educate"    "else"       "encourage"  "engine"
## [11] "europe"    "evidence"   "example"    "excuse"     "exercise"
## [16] "expense"   "experience" "eye"        "health"     "high"
## [21] "knock"     "level"      "local"      "nation"     "non"
## [26] "rather"    "refer"      "remember"   "serious"    "stairs"
## [31] "test"      "tonight"    "transport"  "treat"      "trust"
## [36] "window"    "yesterday"
```

- Contienen pares de letras repetidas (church por ejemplo)

```
str_subset("church", "([A-Za-z][A-Za-z]).*\\1")
```

```
## [1] "church"
```

- Contienen una letra repetida en al menos tres lugares (las tres a de manzana)

```
str_subset(words, "([A-Za-z][A-Za-z]).*\\1")
```

```
##  [1] "appropriate" "church"      "condition"   "decide"      "environment"
##  [6] "london"      "paragraph"   "particular"  "photograph"  "prepare"
## [11] "pressure"    "remember"    "represent"   "require"     "sense"
## [16] "therefore"   "understand"  "whether"
```