Expresiones regulares para la limpieza y transformación de datos

Riva Quiroga y Stephanie Orellana [*] Diego López Tamayo [†]

Contents

[*]Latin R, R-Ladies
[†]El Colegio de México, diego.lopez@colmex.mx

```r
# Importamos las librerías necesarias.
# pacman las instala con las dependencias si es necesario.

# El paquete datos se instala desde git:
# remotes::install_github("cienciadedatos/datos")

rm(list = ls())
pacman::p_load(tidyverse,
               janitor,
               pdftools,
               datos,
               devtools)
```

Introducción

Repositorio del taller aquí

Regex + Tidyverse

Manera que tenemos de pedirle a la computadora que busque un patrón, pensar las expresiones regulares como una lengua especial.

Nos permiten ir de lo literal o abstracto a lo general. Por ejemplo encontrar un número en cualquier parte del texto.

Categorías

Podemos indicarlo con: `\d` ó `[:digit:]` ó `[0-9]`. Existen muchas más categorías, podemos revisar la hoja de referencia de stringr:

Ejemplos

- `\d` cualquier dígito
- `\w` cualquier caracter de palabra
- `\s` espacio en blanco
- `\b` límite de palabra
- `\D` cualuier NO dígito
- `\W` cualquier NO caracter de palabra
- `\S` no espacios en blanco
- `.` cualquier caractér saltos de línea

Cuantificadores

Por ejemplo `\d{4}` nos permite cuantificar y encontrar patrones de 4 dígitos.

- `?` 0 o 1 vez
- `+` 1 o más
- `*` 0 o más
- `{n}` exactamente n

Quiero encontrar una secuencia de cuatro números que estén l final de una cadena de texto: `\d{4}$` utilizando el ancla:

- `^` al inicio del string
- `$` al final del string

Por ejemplo, quiero encontrar la palabra **programación** con o sin tilde en la o:

- `programac[o|ó]n`

Agrupamiento

- `( )` para crear grupos

Supongamos que tenemos: `"Stark, Arya"` yo puedo encerrar en paréntesis y separar en grupos `"(Stark), (Arya)"` donde `\1 corresponde a Stark y \2 corresponde a Arya`.

En R debemos de usar una doble barra `\\d` para que signifique `\d`, o bien `\\?` para que signifique `\?`. No es el caso cuando utilizamos la notación: `[:digit:]`.

Aplicación en R

```
# Importamos desde el repositorio de Github de @rivaquiroga
animales <-
  read_csv("https://raw.githubusercontent.com/rivaquiroga/latinr-taller-regex/master/datos/animales.csv
camarones <-
  read_delim("https://raw.githubusercontent.com/rivaquiroga/latinr-taller-regex/master/datos/casen_cama
telefonos <-
  read_csv("https://raw.githubusercontent.com/rivaquiroga/latinr-taller-regex/master/datos/telefonos.cs
```

Ejemplo con paises

Veremos las posibilidades que existen con la función `filter()`

```
data(paises)

paises %>%
  filter(pais=="México") %>% head(3)
```

```
## # A tibble: 3 x 6
##   pais   continente  anio esperanza_de_vida poblacion pib_per_capita
##   <fct>  <fct>      <int>             <dbl>     <int>          <dbl>
## 1 México Américas    1952              50.8  30144317          3478.
## 2 México Américas    1957              55.2  35015548          4132.
## 3 México Américas    1962              58.3  41121485          4582.
```

Podemos filtrar con str_detect para buscar de manera más flexible en la base de datos, por ejemplo para encontrar el string "Corea" en nuestra columna de paises:

```
paises %>%
  filter(str_detect(pais,"Corea")) %>%  head(3)
```

```
## # A tibble: 3 x 6
##   pais             continente  anio esperanza_de_vida poblacion pib_per_capita
##   <fct>            <fct>      <int>             <dbl>     <int>          <dbl>
## 1 Corea, Rep. Dem. Asia        1952              50.1   8865488          1088.
## 2 Corea, Rep. Dem. Asia        1957              54.1   9411381          1571.
## 3 Corea, Rep. Dem. Asia        1962              56.7  10917494          1622.
```

Para buscar México con y sin acento en nuestra base de datos podemos usar `[ | ]` con `str_detect` dentro de `filter`.

```
paises %>%
  filter(str_detect(pais,"M[e|é]xico")) %>% head(3)
```

```
## # A tibble: 3 x 6
##   pais   continente  anio esperanza_de_vida poblacion pib_per_capita
##   <fct>  <fct>      <int>             <dbl>     <int>          <dbl>
## 1 México Américas    1952              50.8  30144317          3478.
## 2 México Américas    1957              55.2  35015548          4132.
## 3 México Américas    1962              58.3  41121485          4582.
```

Ejemplo con datos de teléfonos

Observemos nuestra base:

```
head(telefonos)
```

```
## # A tibble: 6 x 3
##   nombre             ciudad     numero_telefonico
##   <chr>              <chr>      <chr>
## 1 Pérez, María       Valparaíso +56 981497134
## 2 Juan González      Olmué      928374591
## 3 Hernández, Lautaro quilpué    56 964379583
## 4 Lobos, Antonia     La Serena  987654321
## 5 Martínez, José     La Serena  +56-983726153
## 6 Manuel Garrido     valparaiso 914230795
```

Notar que no hay homogeneidad en la forma en que las personas escribieron sus datos, algunos utilizaron coma para los nombres, la ciudad con letra mayúscula, el número de teléfono con clave internacional etc.

Esto lo podemos resolver con las herramientas de regex.

Corregir la ciudad

```
telefonos %>%
  count(ciudad)
```

```
## # A tibble: 9 x 2
##   ciudad         n
##   <chr>      <int>
## 1 La Serena      2
## 2 Olmué          2
## 3 quilpue        1
## 4 quilpué        3
## 5 Serena         2
## 6 valparaiso     1
## 7 Valparaiso     1
## 8 valparaíso     2
## 9 Valparaíso     5
```

Nos muestra las observaciones distintas para cada categoría. Nos permite observar las diferencias en las categorías.

A continuación queremos identificar las observaciones de Quilpué con y sin acento:

```
telefonos %>%
  filter(str_detect(ciudad,"quilpu[e|é]"))
```

```
## # A tibble: 4 x 3
##   nombre             ciudad  numero_telefonico
##   <chr>              <chr>   <chr>
## 1 Hernández, Lautaro quilpué 56 964379583
```

```
## 2 Sanhueza, Pedro    quilpué 56 981408973
## 3 Aníbal García      quilpue 982733747
## 4 Gómez, Lirayén     quilpué 914257483
```

Ahora buscamos Valparaíso en todas las formas posibles:

```
telefonos %>%
  filter(str_detect(ciudad,"[V|v]alpara[i|í]so"))
```

```
## # A tibble: 9 x 3
##   nombre               ciudad     numero_telefonico
##   <chr>                <chr>      <chr>
## 1 Pérez, María         Valparaíso +56 981497134
## 2 Manuel Garrido       valparaiso 914230795
## 3 Soto, Loreto         valparaíso 925479238
## 4 Pérez, Pedro         valparaíso 985740293
## 5 González, Camila     Valparaiso 56 973629584
## 6 Cayuqueo, Marina     Valparaíso 973629853
## 7 Jaime Espinoza       Valparaíso 56981359284
## 8 Hernández, Francisca Valparaíso 56993418209
## 9 González, Olivia     Valparaíso no tengo
```

`str_detect` nos permite hacer las modificaciones necesarias para encontrar los distintos elementos.

¿Cómo podemos arreglar? Con la función `case_when` de Tidyverse que nos permite modificar una variable basado en ciertas condiciones:

- Del lado izquierdo de `case_when` va lo que queremos encontrar (la condición) y del lado derecho a lo que queremos convertir. Con la función `mutate()` creamos la nueva columna ciudad_limpia para observar nuestras modificaciones.

```
telefonos %>%
  mutate(ciudad_limpia=case_when(
    str_detect(ciudad,"[V|v]alpara[i|í]so") ~ "Valparaíso",
    TRUE ~ as.character(ciudad)
  ))
```

```
## # A tibble: 19 x 4
##    nombre               ciudad     numero_telefonico ciudad_limpia
##    <chr>                <chr>      <chr>             <chr>
##  1 Pérez, María         Valparaíso +56 981497134     Valparaíso
##  2 Juan González        Olmué      928374591         Olmué
##  3 Hernández, Lautaro   quilpué    56 964379583      quilpué
##  4 Lobos, Antonia       La Serena  987654321         La Serena
##  5 Martínez, José       La Serena  +56-983726153     La Serena
##  6 Manuel Garrido       valparaiso 914230795         Valparaíso
##  7 Soto, Loreto         valparaíso 925479238         Valparaíso
##  8 Pérez, Pedro         valparaíso 985740293         Valparaíso
##  9 González, Camila     Valparaiso 56 973629584      Valparaíso
## 10 Jiménez, Luis        Olmué      925403948         Olmué
## 11 Sanhueza, Pedro      quilpué    56 981408973      quilpué
## 12 Aníbal García        quilpue    982733747         quilpue
## 13 Gómez, Lirayén       quilpué    914257483         quilpué
## 14 Cayuqueo, Marina     Valparaíso 973629853         Valparaíso
## 15 Jaime Espinoza       Valparaíso 56981359284       Valparaíso
## 16 Hernández, Francisca Valparaíso 56993418209       Valparaíso
## 17 López, Carla         Serena     56912338475       Serena
```

```
## 18 Soto, Martina        Serena     925653948         Serena
## 19 González, Olivia     Valparaíso no tengo          Valparaíso
```

```
# Le estoy diciendo que para todas las observaciones que cumplan la condición las convierta en Valparaí
```

Notar que ahora todas las observaciones de Valparaíso están uniformes en mayúsculas y con tilde.

Agregemos más condiciones para corregir el resto de ciudades y sobreescribimos nuestra base:

```
telefonos <- telefonos %>%
  mutate(ciudad_limpia=case_when(
    str_detect(ciudad,"[V|v]alpara[i|í]so") ~ "Valparaíso",
    str_detect(ciudad,"quilpu[e|é]") ~ "Quilpué",
    str_detect(ciudad,"Serena") ~ "La Serena",
    TRUE ~ as.character(ciudad)
  ))
```

```
telefonos %>%
  count(ciudad_limpia)
```

```
## # A tibble: 4 x 2
##   ciudad_limpia     n
##   <chr>         <int>
## 1 La Serena         4
## 2 Olmué             2
## 3 Quilpué           4
## 4 Valparaíso        9
```

Separar columnas agrupando

```
peliculas <- read_csv("https://raw.githubusercontent.com/cienciadedatos/datos-de-miercoles/master/datos/

head(peliculas,4)
```

```
## # A tibble: 4 x 9
##   ranking titulo     anio puntaje genero     votos direccion  duracion ganancias
##     <dbl> <chr>     <dbl>   <dbl> <chr>      <dbl> <chr>         <dbl>     <dbl>
## 1       1 The Shaw~  1994     9.3 Drama     2.01e6 Frank Dar~      142      28.3
## 2       2 The Dark~  2008     9   Acción, ~ 1.98e6 Christoph~      152     535.
## 3       3 Inception  2010     8.8 Acción, ~ 1.76e6 Christoph~      148     293.
## 4       4 Fight Cl~  1999     8.8 Drama     1.61e6 David Fin~      139      37.0
```

Notar que en la columna género tenemos un género principal y los géneros secundarios después de la coma, entonces podemos agrupar.

Utilizamos la función `separate` que requiere la columna principal y las columnas en las que queremos separar.

```
pelis_sep <- peliculas %>%
  separate(genero, into=c("genero_principal","genero_secundario"),sep = ", ")
```

```
head(pelis_sep,4)
```

```
## # A tibble: 4 x 10
##   ranking titulo  anio puntaje genero_principal genero_secundar~  votos
##     <dbl> <chr>  <dbl>   <dbl> <chr>            <chr>             <dbl>
## 1       1 The S~  1994     9.3 Drama            <NA>             2.01e6
## 2       2 The D~  2008     9   Acción           Crimen           1.98e6
## 3       3 Incep~  2010     8.8 Acción           Aventura         1.76e6
```

```
## 4       4 Fight~  1999     8.8 Drama            <NA>             1.61e6
## # ... with 3 more variables: direccion <chr>, duracion <dbl>, ganancias <dbl>
```

Notar que estamos perdiendo información para las películas que tienen más de dos géneros secundarios. Podemos ayudarnos del argumento `remove = T,F` que nos permite quitar la columna objetivo, la fijamos en Falso para darnos cuenta de la pérdida de datos.

```
pelis_sep <- peliculas %>%
  separate(genero, into=c("genero_principal","genero_secundario"),sep = ", ", remove = F)
```

```
head(pelis_sep,4)
```

```
## # A tibble: 4 x 11
##   ranking titulo  anio puntaje genero genero_principal genero_secundar~  votos
##     <dbl> <chr>  <dbl>   <dbl> <chr>  <chr>            <chr>             <dbl>
## 1       1 The S~  1994     9.3 Drama  Drama            <NA>             2.01e6
## 2       2 The D~  2008     9   Acció~ Acción           Crimen           1.98e6
## 3       3 Incep~  2010     8.8 Acció~ Acción           Aventura         1.76e6
## 4       4 Fight~  1999     8.8 Drama  Drama            <NA>             1.61e6
## # ... with 3 more variables: direccion <chr>, duracion <dbl>, ganancias <dbl>
```

El argumento `extra` nos permite controlar cuando hay más de dos "piezas", por defecto viene en "warn" que nos avisa cuando elimina todos los valores extra, con "merge" junta todo lo que sobra.

```
pelis_sep <- peliculas %>%
  separate(genero, into=c("genero_principal","genero_secundario"),sep = ", ", remove = F, extra = "merg
```

```
head(pelis_sep,4)
```

```
## # A tibble: 4 x 11
##   ranking titulo  anio puntaje genero genero_principal genero_secundar~  votos
##     <dbl> <chr>  <dbl>   <dbl> <chr>  <chr>            <chr>             <dbl>
## 1       1 The S~  1994     9.3 Drama  Drama            <NA>             2.01e6
## 2       2 The D~  2008     9   Acció~ Acción           Crimen, Drama    1.98e6
## 3       3 Incep~  2010     8.8 Acció~ Acción           Aventura, Cienc~ 1.76e6
## 4       4 Fight~  1999     8.8 Drama  Drama            <NA>             1.61e6
## # ... with 3 more variables: direccion <chr>, duracion <dbl>, ganancias <dbl>
```

Ahora tenemos todos los géneros secundarios en nuestra columna de genero_secundario.

Separar filas

Podemos separar nuestras observaciones de acuerdo a una característica repitiendo todas las demás variables.

```
pelis_row <- peliculas %>%
  separate_rows(genero, sep=", ")
```

```
head(pelis_row,5)
```

```
## # A tibble: 5 x 9
##   ranking titulo       anio puntaje genero  votos direccion   duracion ganancias
##     <dbl> <chr>       <dbl>   <dbl> <chr>   <dbl> <chr>          <dbl>     <dbl>
## 1       1 The Shawsh~  1994     9.3 Drama  2.01e6 Frank Dara~      142      28.3
## 2       2 The Dark K~  2008     9   Acción 1.98e6 Christophe~      152     535.
## 3       2 The Dark K~  2008     9   Crimen 1.98e6 Christophe~      152     535.
## 4       2 The Dark K~  2008     9   Drama  1.98e6 Christophe~      152     535.
## 5       3 Inception    2010     8.8 Acción 1.76e6 Christophe~      148     293.
```

Observamos que género se transformó y ahora tenemos una observación para cada género que tiene una película.

Debemos tener cuidado con la codificación de nuestra base de datos (ASCII, UTF-8,)

Transformar dataset

```
head(animales,4)
```

```
## # A tibble: 4 x 9
##   especie s1_monitoreo201~ s1_monitoreo202~ s1_monitoreo201~ s1_monitoreo202~
##   <chr>   <chr>            <chr>            <chr>            <chr>
## 1 puma    3                2                3                2
## 2 guanaco 22, 3, 45        35; 10;21, 1     13, 11, 2        8, 2 /19
## 3 ñandú   3                4                4;5              6, 4
## 4 zorro   1                3                1                4
## # ... with 4 more variables: s1_monitoreo2019_agosto <chr>,
## #   s1_monitoreo2020_agosto <chr>, s1_monitoreo2019_diciembre <chr>,
## #   s1_monitoreo2020_diciembre <chr>
```

Notar el problema que tenemos para algunas observaciones como guanaco donde la variable s1 para año y mes tiene distintos datos:

```
animales_limpio <- animales %>%
  pivot_longer(starts_with("s"),
               names_pattern="(.*)_monitoreo(.*)_(.*)",
               names_to=c("sitio","año","mes")) %>%
  separate_rows(value)
```

```
head(animales_limpio,4)
```

```
## # A tibble: 4 x 5
##   especie sitio año   mes   value
##   <chr>   <chr> <chr> <chr> <chr>
## 1 puma    s1    2019  enero 3
## 2 puma    s1    2020  enero 2
## 3 puma    s1    2019  abril 3
## 4 puma    s1    2020  abril 2
```