COMMENT MANIPULER DU TEXTE AVEC STRINGR ?

Léonard Boisson

15/11/2020

I - INTRODUCTION

Définition

Le package stringr , développé par Hadley Wickham, a été conçu pour agir comme une simple 'enveloppe' permettant de rendre les fonctionnalités de R applicables aux chaînes de caractères plus cohérentes, simples et faciles à utiliser

Historique

Stringr a été construit à partir du package stringi, qui lui utilise la librairie C/C++ de la ICU (International Components for Unicode), fournissant des implémentations rapides et robustes couvrant pratiquement toutes les manipulations de chaînes de caractères imaginables.

Cette particularité permet au package stringr d'offrir des fonctions qui gèrent convenablement les valeurs manquantes NA ainsi que les caractères de longueur nulle, en plus d'assurer une cohérence au niveau des noms de fonction et d'argument.

Finalement, toutes les fonctionnalités de stringr retournent des structures de données en sortie qui correspondent à celles reçues en entrée par les autres fonctions du package. Cette dernière caractéristique simplifie de beaucoup l'utilisation du résultat d'une fonction comme argument en entrée d'une autre fonction.

Chargement du package

```
library('stringr')
```

II - PRINCIPALES FONCTIONS

On considère le tableau suivant pour tous les exemples qui suivront :

```
library(readxl)
```

```
d <- read_excel("Data_Stringr.xlsx")
```

1. Concaténer des chaînes de caractères avec str_c

- Par défaut, paste concatène en ajoutant un espace entre les différentes chaînes. Il est possible de spécifier un autre séparateur avec son argument sep.

```
str_c(d$`Region Code`, d$`Sector Code`, sep = " - ")
```

```
##  [1] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
##  [3] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
##  [5] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
##  [7] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
##  [9] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [11] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [13] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [15] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [17] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [19] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [21] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [23] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [25] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [27] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [29] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [31] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [33] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [35] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [37] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [39] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [41] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [43] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [45] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [47] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [49] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [51] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [53] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [55] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [57] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [59] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [61] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [63] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [65] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [67] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [69] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [71] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [73] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [75] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [77] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [79] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [81] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [83] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [85] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [87] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [89] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [91] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [93] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [95] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [97] "FTCVS11010000 - FTCVS1101S001" "FTCVS11010000 - FTCVS1101S001"
## [99] "FTCVS11010000 - FTCVS1101S001"
```

- Si on veut concaténer les différents éléments d'un vecteur entre eux, il faut ajouter l'argument collapse qui renvoi le séparateur.

```
str_c(d$`Region Code`, collapse = ", ")
```

```
## [1] "FTCVS11010000, FTCVS11010000, FTCVS11010000, FTCVS11010000, FTCVS11010000, FTCVS11010000, FTCVS
```

2. Convertir en majuscule et minuscule

a) str_to_lower : convertit en minuscule

```
str_to_lower(d$`Region Code`)
```

```
##  [1] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
##  [5] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
##  [9] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [13] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [17] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [21] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [25] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [29] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [33] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [37] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [41] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [45] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [49] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [53] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [57] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [61] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [65] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [69] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [73] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [77] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [81] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [85] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [89] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [93] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [97] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
```

***b) str_to_upper : convertit en majuscule**

```
str_to_lower(d$`Region Code`)
```

```
##  [1] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
##  [5] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
##  [9] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [13] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [17] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [21] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [25] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [29] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [33] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [37] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [41] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [45] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [49] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [53] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [57] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [61] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [65] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [69] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [73] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [77] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [81] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [85] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [89] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [93] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
## [97] "ftcvs11010000" "ftcvs11010000" "ftcvs11010000"
```

***c) str_to_title : capitalise les éléments d'un vecteurde chaînes de caractères**

```
str_to_title(d$`Brand DESC`)
```

```
##  [1] "Coumadin"  "Coumadin"  "Coumadin"  "Eliquis"   "Eliquis"   "Eliquis"
##  [7] "Pradaxa"   "Pradaxa"   "Pradaxa"   "Previscan" "Previscan" "Previscan"
## [13] "Sintrom"   "Sintrom"   "Sintrom"   "Xarelto"   "Xarelto"   "Xarelto"
## [19] "Coumadin"  "Coumadin"  "Coumadin"  "Eliquis"   "Eliquis"   "Eliquis"
## [25] "Pradaxa"   "Pradaxa"   "Pradaxa"   "Previscan" "Previscan" "Previscan"
## [31] "Sintrom"   "Sintrom"   "Sintrom"   "Xarelto"   "Xarelto"   "Xarelto"
## [37] "Coumadin"  "Coumadin"  "Coumadin"  "Eliquis"   "Eliquis"   "Eliquis"
## [43] "Pradaxa"   "Pradaxa"   "Pradaxa"   "Previscan" "Previscan" "Previscan"
## [49] "Sintrom"   "Sintrom"   "Sintrom"   "Xarelto"   "Xarelto"   "Xarelto"
## [55] "Coumadin"  "Coumadin"  "Coumadin"  "Eliquis"   "Eliquis"   "Eliquis"
## [61] "Pradaxa"   "Pradaxa"   "Pradaxa"   "Previscan" "Previscan" "Previscan"
## [67] "Sintrom"   "Sintrom"   "Sintrom"   "Xarelto"   "Xarelto"   "Xarelto"
## [73] "Coumadin"  "Coumadin"  "Coumadin"  "Eliquis"   "Eliquis"   "Eliquis"
## [79] "Pradaxa"   "Pradaxa"   "Pradaxa"   "Previscan" "Previscan" "Previscan"
## [85] "Sintrom"   "Sintrom"   "Sintrom"   "Xarelto"   "Xarelto"   "Xarelto"
## [91] "Coumadin"  "Coumadin"  "Coumadin"  "Eliquis"   "Eliquis"   "Eliquis"
## [97] "Pradaxa"   "Pradaxa"   "Pradaxa"
```

3. Découper des chaînes de caractères avec str_split

- Découpe la chaîne de caractère en fonction d'un délimiteur.

```r
str_split('trois-dix-douze', '-')
```

```
## [[1]]
## [1] "trois" "dix"   "douze"
```

- Appliqué à un vecteur, str_split créé une liste

```r
nom <- c('Léonard Boisson')
```

```r
str_split(nom, ' ')
```

```
## [[1]]
## [1] "Léonard" "Boisson"
```

- Appliqué à un tableau, str_plit peut aussi créer une matrice si on ajoute l'argument simplify = TRUE

```r
str_split(d$`Sales Channel ID`, "_", simplify = TRUE)
```

```
##         [,1]   [,2] 
##    [1,] "CODE" "0"  
##    [2,] "CODE" "1"  
##    [3,] "GERS" "HOP"
##    [4,] "CODE" "0"  
##    [5,] "CODE" "1"  
##    [6,] "GERS" "HOP"
##    [7,] "CODE" "0"  
##    [8,] "CODE" "1"  
##    [9,] "GERS" "HOP"
##   [10,] "CODE" "0"  
##   [11,] "CODE" "1"  
##   [12,] "GERS" "HOP"
##   [13,] "CODE" "0"  
##   [14,] "CODE" "1"  
##   [15,] "GERS" "HOP"
##   [16,] "CODE" "0"  
##   [17,] "CODE" "1"  
##   [18,] "GERS" "HOP"
##   [19,] "CODE" "0"  
##   [20,] "CODE" "1"  
##   [21,] "GERS" "HOP"
##   [22,] "CODE" "0"  
##   [23,] "CODE" "1"  
##   [24,] "GERS" "HOP"
##   [25,] "CODE" "0"  
##   [26,] "CODE" "1"  
##   [27,] "GERS" "HOP"
##   [28,] "CODE" "0"  
##   [29,] "CODE" "1"  
##   [30,] "GERS" "HOP"
##   [31,] "CODE" "0"  
##   [32,] "CODE" "1"  
## [33,] "GERS" "HOP"
## [34,] "CODE" "0"
## [35,] "CODE" "1"
## [36,] "GERS" "HOP"
## [37,] "CODE" "0"
## [38,] "CODE" "1"
## [39,] "GERS" "HOP"
## [40,] "CODE" "0"
## [41,] "CODE" "1"
## [42,] "GERS" "HOP"
## [43,] "CODE" "0"
## [44,] "CODE" "1"
## [45,] "GERS" "HOP"
## [46,] "CODE" "0"
## [47,] "CODE" "1"
## [48,] "GERS" "HOP"
## [49,] "CODE" "0"
## [50,] "CODE" "1"
## [51,] "GERS" "HOP"
## [52,] "CODE" "0"
## [53,] "CODE" "1"
## [54,] "GERS" "HOP"
## [55,] "CODE" "0"
## [56,] "CODE" "1"
## [57,] "GERS" "HOP"
## [58,] "CODE" "0"
## [59,] "CODE" "1"
## [60,] "GERS" "HOP"
## [61,] "CODE" "0"
## [62,] "CODE" "1"
## [63,] "GERS" "HOP"
## [64,] "CODE" "0"
## [65,] "CODE" "1"
## [66,] "GERS" "HOP"
## [67,] "CODE" "0"
## [68,] "CODE" "1"
## [69,] "GERS" "HOP"
## [70,] "CODE" "0"
## [71,] "CODE" "1"
## [72,] "GERS" "HOP"
## [73,] "CODE" "0"
## [74,] "CODE" "1"
## [75,] "GERS" "HOP"
## [76,] "CODE" "0"
## [77,] "CODE" "1"
## [78,] "GERS" "HOP"
## [79,] "CODE" "0"
## [80,] "CODE" "1"
## [81,] "GERS" "HOP"
## [82,] "CODE" "0"
## [83,] "CODE" "1"
## [84,] "GERS" "HOP"
## [85,] "CODE" "0"
## [86,] "CODE" "1"
##  [87,] "GERS" "HOP"
##  [88,] "CODE" "0"
##  [89,] "CODE" "1"
##  [90,] "GERS" "HOP"
##  [91,] "CODE" "0"
##  [92,] "CODE" "1"
##  [93,] "GERS" "HOP"
##  [94,] "CODE" "0"
##  [95,] "CODE" "1"
##  [96,] "GERS" "HOP"
##  [97,] "CODE" "0"
##  [98,] "CODE" "1"
##  [99,] "GERS" "HOP"
```

4. Extraire des sous-chaînes par position avec str_sub

- Extrait des sous-chaînes par position en indiquant les positions des premier et dernier caractères

```
str_sub(d$‘Brand DESC‘, 1, 3)
```

```
##  [1] "COU" "COU" "COU" "ELI" "ELI" "ELI" "PRA" "PRA" "PRA" "PRE" "PRE" "PRE"
## [13] "SIN" "SIN" "SIN" "XAR" "XAR" "XAR" "COU" "COU" "COU" "ELI" "ELI" "ELI"
## [25] "PRA" "PRA" "PRA" "PRE" "PRE" "PRE" "SIN" "SIN" "SIN" "XAR" "XAR" "XAR"
## [37] "COU" "COU" "COU" "ELI" "ELI" "ELI" "PRA" "PRA" "PRA" "PRE" "PRE" "PRE"
## [49] "SIN" "SIN" "SIN" "XAR" "XAR" "XAR" "COU" "COU" "COU" "ELI" "ELI" "ELI"
## [61] "PRA" "PRA" "PRA" "PRE" "PRE" "PRE" "SIN" "SIN" "SIN" "XAR" "XAR" "XAR"
## [73] "COU" "COU" "COU" "ELI" "ELI" "ELI" "PRA" "PRA" "PRA" "PRE" "PRE" "PRE"
## [85] "SIN" "SIN" "SIN" "XAR" "XAR" "XAR" "COU" "COU" "COU" "ELI" "ELI" "ELI"
## [97] "PRA" "PRA" "PRA"
```

5. Détecter des motifs

a) str_detect

- Détecte la présence d’un motif parmi les éléments d’un vecteur, en renvoyant un vecteur de valeurs logiques

```
str_detect(d$‘Brand DESC‘, ’ELIQUIS’)
```

```
##  [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [73] FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
## [97] FALSE FALSE FALSE
```

b. str_count

- Renvoie le nombre de motif

```
str_count(d$`Brick Code`, 'B')
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [77] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
```

c. str_subset

- Renvoie seulement les chaînes de caractères où le motif choisit est présent.

```
str_subset(d$`Sales Channel ID`, 'CODE')
```

```
##  [1] "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1"
##  [9] "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1"
## [17] "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1"
## [25] "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1"
## [33] "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1"
## [41] "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1"
## [49] "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1"
## [57] "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1" "CODE_0" "CODE_1"
## [65] "CODE_0" "CODE_1"
```

6. Extraire des motifs avec str_extract

- Permet d'extraire les valeurs correspondant à un motif en utilisant des expressions régulières en arguments.
- Str_extract extrait la première occurrence

```
str_extract(d$`Brick Code`, '^\\d+')
```

```
##  [1] "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93"
## [16] "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93"
## [31] "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93"
## [46] "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93"
## [61] "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93"
## [76] "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93" "93"
## [91] "93" "93" "93" "93" "93" "93" "93" "93" "93"
```

```
# Isole les numéros
```

- Str_extract_all extrait l'ensemble des nombres présents

```
str_extract_all(d$`Sector Code`, "\\d+")
```

```
## [[1]]
## [1] "1101" "001"
##
## [[2]]
## [1] "1101" "001"
##
## [[3]]
## [1] "1101" "001"
##
## [[4]]
## [1] "1101" "001"
##
## [[5]]
## [1] "1101" "001"
##
## [[6]]
## [1] "1101" "001"
##
## [[7]]
## [1] "1101" "001"
##
## [[8]]
## [1] "1101" "001"
##
## [[9]]
## [1] "1101" "001"
##
## [[10]]
## [1] "1101" "001"
##
## [[11]]
## [1] "1101" "001"
##
## [[12]]
## [1] "1101" "001"
##
## [[13]]
## [1] "1101" "001"
##
## [[14]]
## [1] "1101" "001"
##
## [[15]]
## [1] "1101" "001"
##
## [[16]]
## [1] "1101" "001"
##
## [[17]]
## [1] "1101" "001"
##
## [[18]]
## [1] "1101" "001"
##
```

```
## [[19]]
## [1] "1101" "001"
##
## [[20]]
## [1] "1101" "001"
##
## [[21]]
## [1] "1101" "001"
##
## [[22]]
## [1] "1101" "001"
##
## [[23]]
## [1] "1101" "001"
##
## [[24]]
## [1] "1101" "001"
##
## [[25]]
## [1] "1101" "001"
##
## [[26]]
## [1] "1101" "001"
##
## [[27]]
## [1] "1101" "001"
##
## [[28]]
## [1] "1101" "001"
##
## [[29]]
## [1] "1101" "001"
##
## [[30]]
## [1] "1101" "001"
##
## [[31]]
## [1] "1101" "001"
##
## [[32]]
## [1] "1101" "001"
##
## [[33]]
## [1] "1101" "001"
##
## [[34]]
## [1] "1101" "001"
##
## [[35]]
## [1] "1101" "001"
##
## [[36]]
## [1] "1101" "001"
##
```

```
## [[37]]
## [1] "1101" "001"
##
## [[38]]
## [1] "1101" "001"
##
## [[39]]
## [1] "1101" "001"
##
## [[40]]
## [1] "1101" "001"
##
## [[41]]
## [1] "1101" "001"
##
## [[42]]
## [1] "1101" "001"
##
## [[43]]
## [1] "1101" "001"
##
## [[44]]
## [1] "1101" "001"
##
## [[45]]
## [1] "1101" "001"
##
## [[46]]
## [1] "1101" "001"
##
## [[47]]
## [1] "1101" "001"
##
## [[48]]
## [1] "1101" "001"
##
## [[49]]
## [1] "1101" "001"
##
## [[50]]
## [1] "1101" "001"
##
## [[51]]
## [1] "1101" "001"
##
## [[52]]
## [1] "1101" "001"
##
## [[53]]
## [1] "1101" "001"
##
## [[54]]
## [1] "1101" "001"
##
```

```
## [[55]]
## [1] "1101" "001"
##
## [[56]]
## [1] "1101" "001"
##
## [[57]]
## [1] "1101" "001"
##
## [[58]]
## [1] "1101" "001"
##
## [[59]]
## [1] "1101" "001"
##
## [[60]]
## [1] "1101" "001"
##
## [[61]]
## [1] "1101" "001"
##
## [[62]]
## [1] "1101" "001"
##
## [[63]]
## [1] "1101" "001"
##
## [[64]]
## [1] "1101" "001"
##
## [[65]]
## [1] "1101" "001"
##
## [[66]]
## [1] "1101" "001"
##
## [[67]]
## [1] "1101" "001"
##
## [[68]]
## [1] "1101" "001"
##
## [[69]]
## [1] "1101" "001"
##
## [[70]]
## [1] "1101" "001"
##
## [[71]]
## [1] "1101" "001"
##
## [[72]]
## [1] "1101" "001"
##
```

```
## [[73]]
## [1] "1101" "001"
##
## [[74]]
## [1] "1101" "001"
##
## [[75]]
## [1] "1101" "001"
##
## [[76]]
## [1] "1101" "001"
##
## [[77]]
## [1] "1101" "001"
##
## [[78]]
## [1] "1101" "001"
##
## [[79]]
## [1] "1101" "001"
##
## [[80]]
## [1] "1101" "001"
##
## [[81]]
## [1] "1101" "001"
##
## [[82]]
## [1] "1101" "001"
##
## [[83]]
## [1] "1101" "001"
##
## [[84]]
## [1] "1101" "001"
##
## [[85]]
## [1] "1101" "001"
##
## [[86]]
## [1] "1101" "001"
##
## [[87]]
## [1] "1101" "001"
##
## [[88]]
## [1] "1101" "001"
##
## [[89]]
## [1] "1101" "001"
##
## [[90]]
## [1] "1101" "001"
##
```

```
## [[91]]
## [1] "1101" "001"
##
## [[92]]
## [1] "1101" "001"
##
## [[93]]
## [1] "1101" "001"
##
## [[94]]
## [1] "1101" "001"
##
## [[95]]
## [1] "1101" "001"
##
## [[96]]
## [1] "1101" "001"
##
## [[97]]
## [1] "1101" "001"
##
## [[98]]
## [1] "1101" "001"
##
## [[99]]
## [1] "1101" "001"
```

7. Remplacer des motifs avec str_replace

- Remplace une chaîne de caractère ou un motif par un autre

```
str_replace(d$`Sector Code`, 'FTCVS', 'X')
```

```
##  [1] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
##  [7] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [13] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [19] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [25] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [31] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [37] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [43] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [49] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [55] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [61] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [67] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [73] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [79] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [85] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [91] "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001" "X1101S001"
## [97] "X1101S001" "X1101S001" "X1101S001"
```

- str_replace_all permet de spécifier plusieurs remplacements d'un coup

```
str_remove_all(d$`Sector Code`, c('FTCVS'='X', '00'='AA', '11'='N°'))
```

```
##  [1] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
##  [7] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [13] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [19] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [25] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [31] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [37] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [43] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [49] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [55] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [61] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [67] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [73] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [79] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [85] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [91] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
## [97] "XN°01SAA1" "XN°01SAA1" "XN°01SAA1"
```

8. Modificateur de motifs avec fixed

- On peut spécifier qu'un motif n'est pas une expression régulière mais une chapine de caractère en lui appliquant la fonction fixed.
- Par exemple, si on veut compter le nombre de point, le paramétrage par défaut ne fonctionnera pas car dans une expression régulière, le point signifie n'importe quel caractère.

```
x = 'Bonjour.à.tous.je.m\'appelle.léo.'
```

```
# Paramétrage par défaut
str_count(x, '.')
```

```
## [1] 32
```

```
# Paramétrage avec fixed()
str_count(x, fixed('.'))
```

```
## [1] 6
```

II. EXPRESSIONS REGULIERES

Les ER sont utiles car les chaînes contiennent généralement des données non structurées ou semi-structurées, et les ER sont un langage concis pour décrire les motifs des chaînes. Lorsque vous regardez une ER pour la première fois, vous pensez qu'un chat a marché sur votre clavier, mais au fur et à mesure que votre compréhension s'améliore, les ER commencent à avoir un sens.

Très simplement, l'utilisation d'une expression régulière (ou regex) vous permet de parcourir une séquence de texte, afin d'en faire ressortir les motifs compatibles avec le pattern d'entrée. L'objectif ? Visualiser, modifier, ou encore supprimer…

Quelques liens pour mieux comprendre les RegEx :

- (https://data.hypotheses.org/959)
- (https://informatique-mia.inrae.fr/r4ciam/node/131)
- (https://thinkr.fr/r-les-expressions-regulieres/)

III. RESSOURCES

- (https://stringr.tidyverse.org/) Le site officiel de stringr
- (https://stringr.tidyverse.org/reference/index.html) Liste des fonctions et pages d'aide associées
- (https://stringr.tidyverse.org/articles/regular-expressions.html) Article dédié aux expressions régulières, en anglais
- (http://perso.ens-lyon.fr/lise.vaudor/Descriptoire/_book/intro.html#import-dans-r-de-donnees-textuelles) Cours plus avancé sur stringr
- (https://www.programmersought.com/article/7567693645/) Tutoriel détaillé sur le package stringr