Faciliter la manipulation de chaînes de caractères avec le package stringr

Isabelle Truchon

2017-04-19

Table des matières

Introduction

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. Le package de base de R offre tout de même de bonnes fonctions destinées à la manipulation de chaînes de caractères, mais celles-ci peuvent se montrer parfois incohérentes (étant donné le nombre grandissant de fonctions disponibles au fil des années) et un peu difficiles à utiliser.

`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.

Installation du package `stringr`

Pour utiliser le package `stringr`, il faut en premier lieu procéder à son installation puis le charger :

```
# Installation du package
install.packages("stringr")

# Chargement du package
library(stringr)
```

Jeux de données inclus avec `stringr`

Trois jeux de données sont inclus dans le package `stringr`, soit `fruit`, `words` et `sentences`.

`fruit` et `words` sont deux jeux de données qui proviennent du package `rcorpora`. Les données qu'ils contiennent sont disponibles au lien suivant : https://github.com/dariusk/corpora.

`sentences` constitue une collection de phrases utilisées dans le cadre de tests de voix standardisés effectués à l'Université Harvard.

Ces jeux de données seront utilisés dans cette fiche pour la présentation des exemples.

Fonctionnalités du package `stringr`

Le package `stringr` comprend des fonctions permettant la manipulation des chaînes de caractères pour des opérations de base de même que pour des opérations d'expressions régulières. La présente fiche ne traite cependant que des fonctions pour les manipulations de base. Les expressions régulières n'y sont pas abordées. Le lien qui suit donne tous les détails sur l'ensemble des fonctionnalités du package `stringr` : http://cran.r-project.org/web/packages/stringr/index.html.

Le tableau suivant présente les fonctions de `stringr` qui concernent les manipulations de base, une brève description de la tâche qu'elles accomplissent, ainsi que la fonction du package de base correspondante :

Fonction	Description	Similaire
`str_c()`	Concaténer des chaînes de caractères	`paste0()`
`str_dup()`	Dupliquer des chaînes de caractères	*aucune*
`str_length()`	Calculer le nombre de caractères	`nchar()`
`str_order()`	Ordonner une chaîne de caractères	`order()`
`str_sort()`	Ordonner une chaîne de caractères	`sort()`
`str_pad()`	Compléter une chaîne de caractères par les extrémités	`formatC()`
`str_sub()`	Extraire/remplacer une sous-chaîne de caractères	`substring()`
`str_to_lower()`	Convertir une chaîne de caractères en minuscules	`tolower()`
`str_to_upper()`	Convertir une chaîne de caractères en majuscules	`toupper()`
`str_trim()`	Supprimer des espaces en début et fin	*aucune*
`str_trunc()`	Tronquer une chaîne de caractères	*aucune*
`str_wrap()`	Présenter une chaîne de caractères en paragraphe	`strwrap()`
`word()`	Extraire un mot d'une phrase	*aucune*

Tel que présenté dans le tableau ci-dessus, pratiquement toutes les fonctions de manipulations de base des chaînes de caractères dans `stringr` débutent par `str_`, suivi d'un terme associé directement à leur rôle. De plus, toutes les fonctions du package `stringr` prennent en entrée un vecteur de chaînes de caractères comme premier argument.

Les descriptions détaillées de ces fonctions, accompagnées d'exemples, permettent de constater que certaines d'entre elles offrent une alternative beaucoup plus simple et efficace aux fonctions existantes du package de base.

`str_c()`

La fonction `str_c()` du package `stringr` est équivalente à la fonction `paste0()` du package de base de R. Les deux fonctions concatènent des chaînes de caractères en utilisant la chaîne vide `""` comme séparateur par défaut. L'exemple qui suit produit le même résultat avec les deux fonctions.

```
# Exemple avec la fonction paste0() de l'installation de base R
paste0("Ceci","est","un","exemple","de","concaténation")
```

```
## [1] "Ceciestunexempledeconcaténation"
```

```r
# Exemple avec la fonction str_c() du package stringr
str_c("Ceci","est","un","exemple","de","concaténation")
```

```
## [1] "Ceciestunexempledeconcaténation"
```

Les différences entre les deux fonctions concernent les valeurs manquantes `NA` ainsi que les arguments de longueur nulle.

Les arguments de longueur nulle sont enlevés lors de la concaténation avec la fonction `str_c()`. Dans le cas de la fonction `paste0()`, les arguments de longueur nulle sont plutôt recyclés en chaîne vide `""`.

La fonction `str_c()` du package `stringr` respecte l'absence de valeur en présence de `NA`, tandis que la fonction `paste0()` traite les valeurs manquantes comme des chaînes de caractères `"NA"`. Cependant, le fait que `str_c()` traite convenablement les valeurs manquantes amène le problème commun à la plupart des fonctions de R : une valeur manquante combinée avec une autre chaîne de caractères résulte toujours en une valeur manquante.

```r
fruits <- fruit[1:4]
FRUITS <- str_to_upper(fruits)

fruits[c(1,4)] <- NA
print(fruits)
```

```
## [1] NA        "apricot" "avocado" NA
```

```r
FRUITS[c(1,3)] <- NA
print(FRUITS)
```

```
## [1] NA        "APRICOT" NA        "BANANA"
```

```r
# Traitement convenable des valeurs manquantes NA par la fonction str_c()
# Si une valeur est manquante dans une seule des chaînes de caractères,
# str_c() retourne une valeur manquante
str_c(fruits, FRUITS)
```

```
## [1] NA               "apricotAPRICOT" NA               NA
```

```r
# Traitement des valeurs manquantes NA comme des chaînes de caractères "NA" par la fonction paste0()
paste0(fruits, FRUITS)
```

```
## [1] "NANA"           "apricotAPRICOT" "avocadoNA"      "NABANANA"
```

Contrairement à la fonction `paste0()`, `str_c()` affiche l'avertissement usuel concernant la règle de recyclage lorsque les arguments en entrée ont des longueurs différentes et ne sont pas des multiples exacts l'un de l'autre.

```r
deuxMots <- words[c(1,2)]
print(deuxMots)
```

```
## [1] "a"    "able"
```

```r
cinqMots <- words[100:104]
print(cinqMots)
```

```
## [1] "board" "boat"  "body"  "book"  "both"
```

```
# Concaténation de deux objets de longueur différente avec paste0(): aucun avertissement
paste0(deuxMots, cinqMots)
```

```
## [1] "aboard"   "ableboat" "abody"    "ablebook" "aboth"
```

```
# Concaténation de deux objets de longueur différente avec str_c(): affichage de l'avertissement
str_c(deuxMots, cinqMots)
```

```
## Warning in stri_c(..., sep = sep, collapse = collapse, ignore_null = TRUE):
## longer object length is not a multiple of shorter object length
```

```
## [1] "aboard"   "ableboat" "abody"    "ablebook" "aboth"
```

str_dup()

Le package de base de R n'offre pas de fonction permettant de dupliquer des chaînes de caractères ; ce que le package `stringr` offre avec la fonction `str_dup()`. Cette fonction prend en entrée deux arguments, soit une chaîne de caractères et le nombre de fois qu'on souhaite la dupliquer. Une fois dupliquées, les chaînes de caractères sont concaténées (sans espace). Seul bémol, cette fonction ne permet pas de modifier le séparateur qui est appliqué lors de l'étape de la concaténation.

```
fruitsC <- fruit[17:20]
print(fruitsC)
```

```
## [1] "chili pepper" "clementine"   "cloudberry"   "coconut"
```

```
# Dupliquer 2 fois le vecteur contenant 4 noms de fruit
str_dup(fruitsC,2)
```

```
## [1] "chili pepperchili pepper" "clementineclementine"
## [3] "cloudberrycloudberry"     "coconutcoconut"
```

```
# Dupliquer de 1 à 4 fois le vecteur contenant 4 noms de fruit
str_dup(fruitsC, 1:4)
```

```
## [1] "chili pepper"                    "clementineclementine"
## [3] "cloudberrycloudberrycloudberry" "coconutcoconutcoconutcoconut"
```

str_length()

La fonction `str_length()` qu'offre le package `stringr` est équivalente à la fonction `nchar()` du package de base. Les deux fonctions retournent le nombre de caractères dans une chaîne. Cependant, contrairement à `str_length()`, la fonction `nchar()` du package de base n'accepte pas en entrée des facteurs. La fonction `str_length()` converti les facteurs en caractères et peut ainsi retourner le nombre de caractères que contient une chaîne, tel que souhaité.

```
motsFacteurs <- factor(words[500:504])
print(motsFacteurs)
```

```
## [1] make   man    manage many   mark
## Levels: make man manage many mark
```

```r
# La fonction str_length() accepte les facteurs en entrée et les convertis en caractères
str_length(motsFacteurs)
```

```
## [1] 4 3 6 4 4
```

```r
# La fonction nchar() ne prend pas les facteurs en arguments
nchar(motsFacteurs)
```

```
## Error in nchar(motsFacteurs) : 'nchar()' requires a character vector
```

str_order() / str_sort()

Les fonctions `str_order()` / `str_sort()` permettent d'ordonner des chaînes de caractères, tout comme le font les fonctions `order()` / `sort()` du package de base de R.

Par contre, les fonctions de base `order()` et `sort()` ordonnent les chaînes en utilisant uniquement la langue locale courante. Ainsi, ces fonctions du package de base de R ne permettent pas de modifier la langue et cela peut résulter en un ordonnancement erroné.

Pour éviter ce problème et construire du code plus robuste, `stringr` offre les fonctions `str_order()` et `str_sort()` qui prennent toutes deux un argument additionnel appelé `locale` permettant de modifier la langue.

La langue locale est représentée par une abréviation de deux ou trois lettres, tel que spécifié dans le code des langues ISO 639 https://fr.wikipedia.org/wiki/Liste_des_codes_ISO_639-1.

Par exemple, les éléments d'un vecteur de chaînes de caractères seront ordonnés différemment en langue hawaïenne versus en langue anglaise.

```r
fruitsSort <- fruit[c(1,4,28)]
print(fruitsSort)
```

```
## [1] "apple"    "banana"   "eggplant"
```

```r
# Ordonner avec la fonction de base selon la langue locale
sort(fruitsSort)
```

```
## [1] "apple"    "banana"   "eggplant"
```

```r
# Ordonner avec la fonction du package 'stringr' en langue anglaise
str_sort(fruitsSort, locale = "en")
```

```
## [1] "apple"    "banana"   "eggplant"
```

```r
# Ordonner avec la fonction str_sort() en langue hawaiienne
str_sort(fruitsSort, locale = "haw")
```

```
## [1] "apple"    "eggplant" "banana"
```

str_pad()

La fonction `str_pad()` du package `stringr` prend en entrée une chaîne de caractères et ajoute des caractères aux extrémités de la chaîne, pour atteindre la longueur spécifiée par l'argument `width`. Les caractères à ajouter sont définis par l'argument `pad` qui utilise par défaut l'espace `" "`. Les caractères peuvent être ajoutés

à gauche, à droite ou des deux côtés de la chaîne. Par défaut, cette fonction utilise `side = "left"`, ce qui retourne la chaîne de caractères en donnant l'impression d'être alignée à droite.

Advenant le cas que la chaîne de caractères fournit en entrée est plus longue que la longueur `width` demandée, la chaîne est retournée inchangée.

```
unMot <- words[350]
print(unMot)
```

```
## [1] "future"
```

```
# Utilisation avec les arguments par défaut pour 'side = "left"' et 'pad = " "'
# 'unMot' possède 6 caractères, auxquels sont ajoutés 3 espaces à gauche pour obtenir la largeur 9
str_pad(unMot, width = 9)
```

```
## [1] "   future"
```

```
# On souhaite ajouter un + de chaque côté du mot
str_pad(unMot, width = 8, side = "both", pad = "+")
```

```
## [1] "+future+"
```

Le package de base de R offre une fonction similaire à `str_pad()`, soit `formatC()`, mais cette dernière est nettement moins performante en terme de vitesse. Voici un petit test de comparaison des performances de ces deux fonctions :
(voir le lien http://stackoverflow.com/questions/14409084/pad-with-leading-zeros-to-common-width pour plus de détails)

```
# FORMATC <- formatC(string, width = 2,flag = 0)
# STR_PAD <- str_pad(string, width = 2, side = "left", pad = "0")
```

Unités : millisecondes

Fonction	min	q1	médiane	q3	max
FORMATC	623.53785	629.69005	638.78667	671.22769	679.8790
STR_PAD	116.54969	118.41944	118.97363	120.05729	163.9664

`str_sub()`

Pour extraire ou remplacer une sous-chaîne d'un vecteur de caractères, le package `stringr` offre la fonction `str_sub()` équivalente à la fonction de base `substring()`. Les trois arguments que prend en entrée `str_sub()` sont un vecteur de caractères, une valeur de départ `start` indiquant la position du premier caractère à inclure dans la sous-chaîne, de même qu'une valeur `end` indiquant la position du dernier caractère.

```
phrase <- sentences[6]
print(phrase)
```

```
## [1] "The juice of lemons makes fine punch."
```

```
# Extraction d'une sous-chaîne avec la fonction str_sub()
str_sub(phrase, start = 5, end = 9)
```

```
## [1] "juice"
```

```r
# Extraction d'une sous-chaîne avec la fonction de base substring():
# le résultat est le même avec les deux fonctions
substring(phrase, first = 5, last = 9)
```

```
## [1] "juice"
```

```r
# Les valeurs par défaut de 'start' et 'end' sont respectivement le premier et dernier caractère
# L'exemple suivant utilise l'argument 'end' par défaut
str_sub("juice", start = 1:3)
```

```
## [1] "juice" "uice"  "ice"
```

```r
# Remplacements multiples d'une sous-chaîne de caractères
str_sub(phrase, start = c(14,27), end = c(19,30)) <- c("oranges", "good")
print(phrase)
```

```
## [1] "The juice of oranges makes fine punch."
## [2] "The juice of lemons makes good punch."
```

Une des fonctionnalités intéressantes de `str_sub()` est sa capacité à travailler avec des indices négatifs donnés en argument à `start` et `end`. Lorsqu'on fournit une position négative, `str_sub()` compte à rebours, à partir du dernier caractère de la chaîne. En comparaison, la fonction `substring()` du package de base ignore les indices négatifs et ne retourne que des chaînes vides.

```r
desMots <- words[587:592]
print(desMots)
```

```
## [1] "other"     "otherwise" "ought"     "out"       "over"      "own"
```

```r
# str_sub() utilisée avec des positions négatives: les deux derniers caractères sont extraits
str_sub(desMots, start = -2, end = -1)
```

```
## [1] "er" "se" "ht" "ut" "er" "wn"
```

```r
# Utilisation des positions négatives avec substring(): retourne des chaînes vides
substring(desMots, first = -2, last = -1)
```

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

```r
phrase2 <- sentences[16]
print(phrase2)
```

```
## [1] "A pot of tea helps to pass the evening."
```

```r
# Remplacement d'une sous-chaîne de caractères en utilisant des positions négatives
str_sub(phrase2, start = -8, end = -2) <- "night"
print(phrase2)
```

```
## [1] "A pot of tea helps to pass the night."
```

`str_to_lower()` / `str_to_upper()`

Les fonctions `str_to_lower()` et `str_to_upper()` du package `stringr` permettent de convertir la casse d'une chaîne de caractères, comme le font les fonctions du package de base `tolower()` et `toupper()`. Cependant, les

fonctions de `stringr` offrent la possibilité de modifier la langue à utiliser pour la conversion avec l'argument `locale`. En effet, certaines langues utilisent des règles différentes de conversion de casse.

C'est le cas du Turc qui possède deux 'i' différents, un sans le point et un autre avec le point. La règle de conversion de la casse sera donc différente de celle utilisée pour l'Anglais par exemple.

```
# Convertir la lettre 'i' en majuscule dans la langue courante par défaut avec str_to_upper()
str_to_upper("i")
```

```
## [1] "I"
```

```
# Convertir la lettre 'i' en majuscule dans la langue courante par défaut avec la fonction toupper()
# Le résultat est le même avec les deux fonctions
toupper("i")
```

```
## [1] "I"
```

```
# Convertir la lettre 'i' en majuscule dans la langue turque
# Dans cet exemple, la lettre 'i' majuscule en Turc s'écrit aussi avec un point
str_to_upper("i", locale = "tr")
```

```
## [1] "İ"
```

str_trim()

La fonction `str_trim()` de `stringr` permet d'enlever les espaces aux extrémités d'une chaîne de caractères. Deux arguments sont à fournir en entrée, soit le vecteur de caractères et l'extrémité de la chaîne dont on veut enlever des espaces. Par défaut, l'argument `side` prend la valeur `"both"`.

```
desFruits <- c("  lychee ", "mandarine   ", "  mango ")

# Enlever les espaces à gauche
str_trim(desFruits, side = "left")
```

```
## [1] "lychee "      "mandarine   " "mango "
```

```
# Enlever les espaces des deux côtés (par défaut)
str_trim(desFruits)
```

```
## [1] "lychee"    "mandarine" "mango"
```

str_trunc()

Cette fonction du package `stringr` permet de tronquer des chaînes de caractères en spécifiant en entrée la longueur maximale à conserver, à partir de quel côté la troncature doit être faite, de même que les caractères à utiliser pour indiquer à partir de quelle position la chaîne a été tronquée.

```
phrase3 <- sentences[29]
print(phrase3)
```

```
## [1] "It snowed, rained, and hailed the same morning."
```

```
# Utilisation de str_trunc() avec l'argument 'ellipsis' par défaut (...)
str_trunc(phrase3, 20, side = "right")
```

```
## [1] "It snowed, rained..."
```

str_wrap()

La fonction `str_wrap()` du package `stringr` est similaire à la fonction de base `strwrap()`, utilisée pour formater en paragraphe une longue chaîne de caractères. La fonction `str_wrap()` permet tout d'abord de séparer la longue chaîne de caractères selon une largeur spécifiée de paragraphe (`width =`). Ensuite, cette fonction permet de définir un espacement pour la première ligne du paragraphe (`indent =`) et pour les lignes suivantes (`extend =`). Les valeurs par défaut sont de 80 pour l'argument `width` et 0 pour les arguments `indent` et `exdent`.

La différence entre les deux fonctions `str_wrap()` et `strwrap()` se situe au niveau de la performance en terme de vitesse. En effet, étant donné que la fonction `str_wrap()` appelle `stri_wrap` du package `stringi` (construit en C/C++), sa performance sera meilleure que celle de `strwrap()`.

```
texte <- str_c(sentences[603],sentences[609],sentences[64], sep = " ")
print(texte)
```

```
## [1] "There is a strong chance it will happen once more. A six comes up more often than a ten. The cl
```

```
# Affichage du texte sous forme d'un paragraphe avec largeur de colonne de 30
cat(str_wrap(texte, width = 30))
```

```
## There is a strong chance it
## will happen once more. A six
## comes up more often than a
## ten. The clock struck to mark
## the third period.
```

```
# Affichage du texte avec espacement de la première ligne
cat(str_wrap(texte, width = 30, indent = 3))
```

```
##    There is a strong chance it
## will happen once more. A six
## comes up more often than a
## ten. The clock struck to mark
## the third period.
```

```
# Affichage du texte avec espacement des lignes suivantes
cat(str_wrap(texte, width = 30, exdent = 3))
```

```
## There is a strong chance it
##    will happen once more. A six
##    comes up more often than a
##    ten. The clock struck to mark
##    the third period.
```

word()

La fonction `word()` de `stringr` a été conçue pour extraire des mots d'une phrase. Cette fonction prend en arguments une chaîne de caractères, puis une position de départ `start` et de fin `end` spécifiant le premier et le dernier mot à extraire de la chaîne de caractères. Les arguments par défaut pour `start` et `end` sont dans les deux cas le premier mot et le séparateur `sep` est un espace simple `" "`. Les indices des positions peuvent aussi être négatifs.

```r
phrases <- c(sentences[680], sentences[679])
print(phrases)
```

```
## [1] "Dig deep in the earth for pirate's gold."
## [2] "No doubt about the way the wind blows."
```

```r
# Extraction du premier mot de chaque phrase avec la valeur par défaut pour l'argument 'end'
word(phrases, start = 1)
```

```
## [1] "Dig" "No"
```

```r
# Extraction de l'avant-dernier mot de chaque phrase
word(phrases, start = -2)
```

```
## [1] "pirate's" "wind"
```

```r
# Extraction de tous les mots sauf les trois premiers
word(phrases, start = 4, end = -1)
```

```
## [1] "the earth for pirate's gold." "the way the wind blows."
```

Avantages et inconvénients du package `stringr`

Selon la description officielle du package `stringr` (https://CRAN.R-project.org/package=stringr), ses avantages majeurs sont les suivants :

- Assure une cohérence au niveau des noms de fonction et d'argument
- Offre des fonctions qui gèrent convenablement les valeurs manquantes `NA` ainsi que les caractères de longueur nulle
- Toutes les fonctionnalités de `stringr` permettent l'utilisation du résultat d'une fonction comme argument en entrée d'une autre fonction

À la lumière de ce qui a été testé dans cette fiche sur les fonctions de `stringr`, il appert que ce package présente des avantages notables par rapport aux fonctions de base pour les opérations sur des chaînes de caractères. Non seulement il offre des fonctionnalités inexistantes dans le package de base (par exemple `str_trunc()` ou `word()`), mais en plus, il offre des possibilités additionnelles pour certaines fonctions de base. C'est le cas notamment de la fonction `str_sub()` qui permet de donner en argument des indices de position négatifs, contrairement à `substring()` du package de base qui ne les reconnaît pas.

Il semble difficile d'identifier un inconvénient à utiliser le package `stringr` pour effectuer des opérations sur des chaînes de caractères. En effet, `stringr` ne présente que des avantages par rapport au package de base de R. Bien sûr, certaines de ses fonctions utilisées dans des cas simples peuvent ne présenter aucune différence significative en comparaison avec la fonction de base équivalente. Mais il y aura tout de même toujours un léger avantage en terme de performance, bien que pas nécessairement visible pour l'utilisateur. Ainsi, outre le fait de devoir l'installer et le charger dans notre programme, il n'y a aucun inconvénient à utiliser les fonctionnalités du package `stringr`.

Conclusion

En comparaison avec les fonctions du package de base de R pour la manipulation des chaînes de caractères, `stringr` constitue un package recommandé pour sa cohérence et la simplicité d'utilisation de ses fonctionnalités. De plus, la performance qu'il offre en terme de vitesse peut s'avérer très avantageuse dans certaines situations, particulièrement dans le cadre de travaux effectués sur une grande quantité de données.

Selon les données fournies par CRAN (Comprehensive R Archive Network) et compilées sur le blogue R-Statistics (https://www.r-statistics.com/2013/06/top-100-r-packages-for-2013-jan-may/), le package `stringr`, développé par Hadley Wickham, a été désigné comme étant le cinquième package de R le plus téléchargé en 2013. Avec quelques milliers de packages maintenant disponibles, `stringr` constitue encore aujourd'hui un des quatre packages destinés à la manipulation de données les plus recommandés par l'équipe de support de RStudio (https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages).

Références et hyperliens

1. Wickham, Hadley (2010). *The R Journal Vol.2/2 : stringr : modern, consistent string processing*, p.38-40. Référence web : https://journal.r-project.org/archive/2010-2/RJournal_2010-2_Wickham.pdf
2. Wickham H., Grolemund G. (2016). *R for Data Science : Import, Tidy, Transform, Visualize, and Model Data*, O'Reilly Media, Inc., 520 pages. Référence web : http://r4ds.had.co.nz/
3. Sanchez, G. (2013). *Handling and Processing Strings in R*, Trowchez Editions, Berkeley. Référence web : http://www.gastonsanchez.com/Handling_and_Processing_Strings_in_R.pdf
4. http://stringr.tidyverse.org
5. https://www.rdocumentation.org/packages/stringr/versions/1.1.0/topics/str_c
6. http://stackoverflow.com/questions/14409084/pad-with-leading-zeros-to-common-width
7. Galili, Tal (2013). *R-statistics blog, Statistics with R, and open source stuff (software, data, community) : Top 100 R packages for 2013 (Jan-May) !*. Référence web : https://www.r-statistics.com/2013/06/top-100-r-packages-for-2013-jan-may/
8. Grolemund G. (2017). *Quick list of useful R packages : Recommended Packages : To manipulate data*, RStudio Support. Référence web : https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages
9. Wickham, Hadley (2017). *stringr : Simple, Consistent Wrappers for Common String Operations*. Référence web : https://CRAN.R-project.org/package=stringr
10. https://fr.wikipedia.org/wiki/Liste_des_codes_ISO_639-1