R Notebook

# String processing

### Introduction

Tous tâche d’un data scientist commence par la mis á propres des données á analyser. Le Data cleaning est un étape indispensable dans l’analyse de données et il prends la grande partie d’un de temps d’un scientifique de données. Il peux consister au traitement des chaînes de charactères, des valeurs maquantes, convertion de types, opérations sur les dates etc. Il est très cruciale de bien purifier les données car une mauvaise purification de données peut entrainer, Une erreurs d’analyse, une perte d’information ou encore une prise de décision erroné, etc.

Le traitement avec les chaînes de caractère est l’une opération dans la purification de données, dans ce cas d’utilisation on va voir les différents problèmes qu’on peut rencontrer quand on traite avec les caractères on verras plus précisement:

* Extractions des nombres d’une chaines de caractères
* Suppressions des charactères non désiré d’un text
* recherche d’un modèle spécifique et le remplacer par un autre dans une chaine
* extraction une partie spécifique d’un text
* Division d’une chaine de caractère en de multiples parties

Le packages nommés **base** de R fournie des fonctions pour éffectué toutes ces tâches cependant le nom de ces fonction n’est pas faciles á mémoriser. Heureusement il y a un package de R appélé **stringr** qui réalise d’une manière concise l’ensemble de ces opérations mais aussi le nom est faciles a mémoriser, toutes les fonctions opérant sur les chaines de charactères commence par le mot **str\_**. Par exemple la fonction **str\_split** de stringr est la fonction qui divise une chaine de charactère selon un séparateur qui lui est fournie.

#### Importation des bibliothèques

* **tidyverse** une groupe de packages regroupant tous les outils pour l’analyse de données
* **stringr** le package en question qui fournie les outils pour traiter avec les chaines de charactères
* **Dslabs** est un packages construit pour les data scientists, il contient les fonctions et les datasets utilisés par pour pratiquer l’ensemble des challenges qu’on peut rencontrer en analyse de données.

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.2 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(stringr)  
library(dslabs)

dslabs contient un dataset appélé **reported\_heights**, il est construits á la suite d’un formulaire web que les etudiant on remplis leur demandant de reporter leur tailles. Nous allons l’utilisé dans ce documents

data("reported\_heights")

class(reported\_heights$height)

## [1] "character"

Les tailles reportées sont des chaîne de caractères. Créons une variable x a qui on affecte la taille.

x <- reported\_heights$height

x[c(1:30, 60:100, 150:200)]

## [1] "75" "70" "68"   
## [4] "74" "61" "65"   
## [7] "66" "62" "66"   
## [10] "67" "72" "6"   
## [13] "69" "68" "69"   
## [16] "66" "75" "64"   
## [19] "60" "67" "66"   
## [22] "5' 4\"" "70" "73"   
## [25] "72" "69" "69"   
## [28] "72" "64" "72"   
## [31] "67" "69" "73"   
## [34] "74" "70" "66"   
## [37] "511" "72" "65"   
## [40] "65" "70" "73"   
## [43] "67" "72" "68"   
## [46] "68" "65" "72"   
## [49] "71" "65" "72"   
## [52] "69" "70" "72"   
## [55] "6" "62" "65"   
## [58] "70" "60" "67"   
## [61] "62" "71" "63"   
## [64] "68" "64.1732" "64"   
## [67] "71" "68.5" "62"   
## [70] "2" "70" "5'3\""   
## [73] "73" "68" "77"   
## [76] "70.5" "63" "69"   
## [79] "69" "68.89" "66.5"   
## [82] "64.173" "63" "65"   
## [85] "64" "63" "63"   
## [88] "69" "69" "64"   
## [91] "62" "70" "70"   
## [94] "59" "65" "67.7"   
## [97] "72" "74" "71.7"   
## [100] "70.87" "66" "72"   
## [103] "74" "69" "71"   
## [106] "70" "70" "64"   
## [109] "5 feet and 8.11 inches" "68" "66"   
## [112] "64" "67" "65"   
## [115] "72" "5.25" "70"   
## [118] "64.57" "51" "63"   
## [121] "70" "68"

Les données reporté ici, ne sont pas sous le même format même s’il sont correct, il serait difficile de les analysé, nous allons les ramener sous le même format. Les étudiant devraient rapporter leur tailles en inches qui est une unité de mésure utilisé dans certains pays comme le Royaume Uni, le canada et les USA .Mais il y a certains qui ont reporter en **Cm**, d’autre en **Feet** et d’autre en lettre comme **5 feet and 8.11 inches**

Premièrement on récupérer les données qui ne sont pas en **pouces**( inches) ni en **Cm** pour cela on écirt une fonction qui récupère ces données. La taille minimal en **inche** est 50 et le maximal est 80. La fonction prends en paramètre minimun de inches, le maximum de inches et le données en question

la fonction récupere les valeurs non disponible après être transformé en numerique, les numérique qui sont inférieur au minimum et supérieur au maximum ce qui veux dire qu’il ne sont pas en inches, et le dernier c’est ceux qui ne sont pas noté en centimètre. Pour les autres ils sont correctement écrit en inches

not\_inches\_or\_cm <- function(x, smallest = 50, tallest = 84){   
inches <- suppressWarnings(as.numeric(x))  
ind <- !is.na(inches) & ((inches >= smallest & inches <= tallest) | (inches/2.54 >= smallest & inches/2.54 <= tallest))  
!ind }  
  
problematiques <- reported\_heights %>% filter(not\_inches\_or\_cm(height)) %>% pull(height)

c(length(problematiques), length(x))

## [1] 200 1095

as\_tibble(problematiques)

## # A tibble: 200 × 1  
## value   
## <chr>   
## 1 "6"   
## 2 "5' 4\""  
## 3 "5.3"   
## 4 "165cm"   
## 5 "511"   
## 6 "6"   
## 7 "2"   
## 8 "5'7"   
## 9 ">9000"   
## 10 "5'7\""   
## # ℹ 190 more rows

Après analyse, on distingue les problème suivant:

1. x’y ou x’ y’ ’ ou x’y’’ ou x reprentes le foot et y le inches
2. x.y. ou x,y de la même manière x represente le foot et y le inches
3. ceux qui sont representé en cm
4. et bien evidement d’autre entre

Tout d’abord on va commencer par ces trois premier ensuite attacker le reste. On va utilisé les expression régulier pour effectuer ces text.

#### Les Expressions régulière

Une expresion regulière est une description d’un modèle d’un ensemble chaine de caractère utilisée pour effectuer des recherches dans un text. Si on demande á un utilisateur d’entrer son adresse mail, on peux définir un modèl d’adress mail avec qui on verifie si l’adress rentré correspond bien au norme. Voic quelques exemples d’expression régulière en r:

* \\d signifie un chiffre quelconques
* [] ceci represente un classe de charactère ei: [1-5] peux correspondre a 1,2,3,4 ou 5
* Anchors: permet de definir la debut et la fin du modèle. ^ debut $ fin
* quantificateurs: permet de definir un nombre de charactère {1,3,4} 1, 3 ou c4 charactère
* represente les espaces
* +, \*, ? represente respectivement 1 ou plus, 0 ou plus, 1 et 1 seul charactère
* ^ á l’interieur d’un chrochet signifie non ex : [^a-zA-Z] pas d’alphabet majuscule ou miniscule
* Pour echapper les charactères spéciaux on utilisie \ ex: “^\\d\”$”

On a une idée des expression régulier en r, on va parcourir les différent modèle de problème existant

model <- "^[4-7]'\\d{1,2}\"$" # commence par un chiffre entre 4 et 7 suivie de ' et d'un ou deux chiffre et "  
 str\_subset(problematiques, model)

## [1] "5'7\"" "5'3\"" "5'5\"" "5'2\"" "5'3\"" "5'2\"" "5'8\"" "5'11\""  
## [9] "5'7\"" "6'1\"" "5'8\"" "6'3\"" "5'7\"" "6'4\""

class(problematiques)

## [1] "character"

str\_subset(problematiques, "inches")

## [1] "5 feet and 8.11 inches" "Five foot eight inches" "5 feet 7inches"   
## [4] "5ft 9 inches" "5 ft 9 inches" "5 feet 6 inches"

str\_subset(problematiques, "''")

## [1] "5'9''" "5'10''" "5'10''" "5'3''" "5'7''" "5'6''" "5'7.5''"  
## [8] "5'7.5''" "5'10''" "5'11''" "5'10''" "5'5''"

str\_subset(problematiques, "\\d{1}\\.\\d{1,2}")

## [1] "5.3" "5 feet and 8.11 inches" "5.25"   
## [4] "5.5" "6.5" "103.2"   
## [7] "5.8" "5.6" "5.9"   
## [10] "5.5" "6.2" "6.2"   
## [13] "5.8" "5.1" "5.11"   
## [16] "5.75" "5.4" "6.1"   
## [19] "5.6" "5.6" "0.7"   
## [22] "5.4" "5.9" "5.6"   
## [25] "5.6" "5.5" "5.2"   
## [28] "5.5" "5.5" "6.5"   
## [31] "5.11" "5.5" "5'7.5''"   
## [34] "5'7.5''" "5' 7.78\"" "6.7"   
## [37] "5.1" "5.6" "5.5"   
## [40] "5.2" "5.6" "5.7"   
## [43] "5.9" "6.5" "5.11"   
## [46] "1.6" "5.7" "5.5"   
## [49] "1.7" "5.8" "5.8"   
## [52] "5.1" "5.11" "5.7"   
## [55] "5.9" "5.2" "5.5"   
## [58] "5.51" "5.8" "5.7"   
## [61] "6.1" "5.69" "5.7"   
## [64] "5.25" "5.5" "5.1"   
## [67] "6.3" "5.5" "5.7"   
## [70] "5.57" "5.7"

pattern <- "^[4-7]\\s\*'\\s\*\\d{1,2}$"  
str\_subset(problematiques, pattern)

## [1] "5'7" "5'11" "5' 10" "5'3" "5'12" "5'11" "5'4" "5'5" "5'8"   
## [10] "5'6" "5'4" "5'5" "5'7" "5'6" "5'7" "5'8" "5'10" "5'10"   
## [19] "5'2" "5'11" "5'8" "5'9" "5'4" "5'6" "5'6"

model <- "^[4-7]\\s\*'\\s\*\\d{1,2}$" # par exemple 5'10 correspond

problematiques %>% str\_replace("feet|ft|foot", "'") %>%   
 str\_replace("inches|in|''|\"", "") %>%   
 str\_detect(pattern = model) %>% sum()

## [1] 53

On ecris deux fonction:

* Une fonction qui reconvertie le format des données
* Une fonction qui convertis les nombres qui sont écris en lettres en chiffres

convert\_format <- function(s){  
 s %>%   
 str\_replace("feet|foot|ft", "'") %>%   
 str\_replace\_all("inches|in|''|\"|cm|and", "") %>%   
 str\_replace("^([4-7])\\s\*[,\\.\\s+]\\s\*(\\d\*)$", "\\1'\\2") %>%   
 str\_replace("^([56])'?$", "\\1'0") %>%   
 str\_replace("^([12])\\s\*,\\s\*(\\d\*)$", "\\1\\.\\2") %>%   
 str\_trim()  
}  
  
# convert word to number  
words\_to\_numbers <- function(s){ str\_to\_lower(s) %>%  
 str\_replace\_all("zero", "0") %>%  
 str\_replace\_all("one", "1") %>%   
 str\_replace\_all("two", "2") %>%  
 str\_replace\_all("three", "3") %>%  
 str\_replace\_all("four", "4") %>%   
 str\_replace\_all("five", "5") %>%   
 str\_replace\_all("six", "6") %>%   
 str\_replace\_all("seven", "7") %>%   
 str\_replace\_all("eight", "8") %>%  
 str\_replace\_all("nine", "9") %>%  
 str\_replace\_all("ten", "10") %>%   
 str\_replace\_all("eleven", "11")  
}

Maintenant on fait tous en une seul bloques

pattern <- "^([4-7])\\s\*'\\s\*(\\d+\\.?\\d\*)$"  
smallest <- 50  
tallest <- 84  
new\_heights <- reported\_heights %>%  
mutate(original = height, height = words\_to\_numbers(height) %>% convert\_format()) %>% extract(height, c("feet", "inches"), regex = pattern, remove = FALSE) %>% mutate\_at(c("height", "feet", "inches"), as.numeric) %>%  
mutate(guess = 12\*feet + inches) %>%  
mutate(height = case\_when(  
!is.na(height) & between(height, smallest, tallest) ~ height,   
!is.na(height) & between(height/2.54, smallest, tallest) ~ height/2.54,   
!is.na(height) & between(height\*100/2.54, smallest, tallest) ~ height\*100/2.54,  
!is.na(guess) & inches < 12 & between(guess, smallest, tallest) ~ guess,   
TRUE ~ as.numeric(NA))) %>%  
select(-guess)

## Warning: There was 1 warning in `mutate()`.  
## ℹ In argument: `height = .Primitive("as.double")(height)`.  
## Caused by warning:  
## ! NAs introduced by coercion

new\_heights %>% filter(not\_inches\_or\_cm(original)) %>%  
 select(original, height) %>%   
 arrange(height)

## original height  
## 1 5 60.00000  
## 2 5 60.00000  
## 3 5 60.00000  
## 4 5 60.00000  
## 5 5.1 61.00000  
## 6 5.1 61.00000  
## 7 5.1 61.00000  
## 8 5.1 61.00000  
## 9 5'2" 62.00000  
## 10 5.2 62.00000  
## 11 5'2" 62.00000  
## 12 5.2 62.00000  
## 13 5.2 62.00000  
## 14 5'2 62.00000  
## 15 1.6 62.99213  
## 16 5.3 63.00000  
## 17 5'3" 63.00000  
## 18 5,3 63.00000  
## 19 5'3 63.00000  
## 20 5'3'' 63.00000  
## 21 5'3" 63.00000  
## 22 5' 4" 64.00000  
## 23 5,4 64.00000  
## 24 5.4 64.00000  
## 25 5.4 64.00000  
## 26 5'4 64.00000  
## 27 5'4 64.00000  
## 28 5'4 64.00000  
## 29 165cm 64.96063  
## 30 5.5 65.00000  
## 31 5.5 65.00000  
## 32 5'5" 65.00000  
## 33 5.5 65.00000  
## 34 5.5 65.00000  
## 35 5.5 65.00000  
## 36 5.5 65.00000  
## 37 5.5 65.00000  
## 38 5'5 65.00000  
## 39 5.5 65.00000  
## 40 5'5 65.00000  
## 41 5.5 65.00000  
## 42 5.5 65.00000  
## 43 5'5'' 65.00000  
## 44 5.5 65.00000  
## 45 5.6 66.00000  
## 46 5.6 66.00000  
## 47 5.6 66.00000  
## 48 5.6 66.00000  
## 49 5.6 66.00000  
## 50 5'6'' 66.00000  
## 51 5.6 66.00000  
## 52 5'6 66.00000  
## 53 5.6 66.00000  
## 54 5'6 66.00000  
## 55 5 feet 6 inches 66.00000  
## 56 5'6 66.00000  
## 57 5'6 66.00000  
## 58 1,70 66.92913  
## 59 1.7 66.92913  
## 60 170 cm 66.92913  
## 61 5'7 67.00000  
## 62 5'7" 67.00000  
## 63 5'7'' 67.00000  
## 64 5 feet 7inches 67.00000  
## 65 5.7 67.00000  
## 66 5.7 67.00000  
## 67 5.7 67.00000  
## 68 5'7 67.00000  
## 69 5'7" 67.00000  
## 70 5'7 67.00000  
## 71 5' 7" 67.00000  
## 72 5.7 67.00000  
## 73 5.7 67.00000  
## 74 5'7" 67.00000  
## 75 5.7 67.00000  
## 76 5.7 67.00000  
## 77 5'7.5'' 67.50000  
## 78 5'7.5'' 67.50000  
## 79 5' 7.78" 67.78000  
## 80 5.8 68.00000  
## 81 Five foot eight inches 68.00000  
## 82 5.8 68.00000  
## 83 5,8 68.00000  
## 84 5'8 68.00000  
## 85 5.8 68.00000  
## 86 5'8" 68.00000  
## 87 5.8 68.00000  
## 88 5'8 68.00000  
## 89 5.8 68.00000  
## 90 5'8" 68.00000  
## 91 5'8 68.00000  
## 92 5 feet and 8.11 inches 68.11000  
## 93 5'9'' 69.00000  
## 94 5.9 69.00000  
## 95 5.9 69.00000  
## 96 5.9 69.00000  
## 97 5.9 69.00000  
## 98 69" 69.00000  
## 99 5ft 9 inches 69.00000  
## 100 5 ft 9 inches 69.00000  
## 101 5'9 69.00000  
## 102 5'10'' 70.00000  
## 103 5' 10 70.00000  
## 104 5'10'' 70.00000  
## 105 5'10'' 70.00000  
## 106 5'10 70.00000  
## 107 5'10 70.00000  
## 108 5'10'' 70.00000  
## 109 5'11 71.00000  
## 110 5.11 71.00000  
## 111 5'11 71.00000  
## 112 5.11 71.00000  
## 113 5.11 71.00000  
## 114 5 .11 71.00000  
## 115 5 11 71.00000  
## 116 5.11 71.00000  
## 117 5'11" 71.00000  
## 118 5' 11" 71.00000  
## 119 5'11 71.00000  
## 120 5'11'' 71.00000  
## 121 6 72.00000  
## 122 6 72.00000  
## 123 6 72.00000  
## 124 6' 72.00000  
## 125 6 72.00000  
## 126 6 72.00000  
## 127 6 72.00000  
## 128 6 72.00000  
## 129 6 72.00000  
## 130 6 72.00000  
## 131 6 72.00000  
## 132 6 72.00000  
## 133 6 72.00000  
## 134 6 72.00000  
## 135 6 72.00000  
## 136 6 72.00000  
## 137 6 72.00000  
## 138 6 72.00000  
## 139 6 72.00000  
## 140 6 72.00000  
## 141 6.1 73.00000  
## 142 6'1" 73.00000  
## 143 6.1 73.00000  
## 144 6.2 74.00000  
## 145 6.2 74.00000  
## 146 6'3" 75.00000  
## 147 6.3 75.00000  
## 148 6 04 76.00000  
## 149 6'4" 76.00000  
## 150 6.5 77.00000  
## 151 6.5 77.00000  
## 152 6.5 77.00000  
## 153 2 78.74016  
## 154 6.7 79.00000  
## 155 6,8 80.00000  
## 156 511 NA  
## 157 >9000 NA  
## 158 5.25 NA  
## 159 11111 NA  
## 160 103.2 NA  
## 161 19 NA  
## 162 300 NA  
## 163 5.75 NA  
## 164 7 NA  
## 165 214 NA  
## 166 0.7 NA  
## 167 5'12 NA  
## 168 2'33 NA  
## 169 612 NA  
## 170 87 NA  
## 171 111 NA  
## 172 12 NA  
## 173 yyy NA  
## 174 89 NA  
## 175 34 NA  
## 176 25 NA  
## 177 22 NA  
## 178 684 NA  
## 179 1 NA  
## 180 1 NA  
## 181 6\*12 NA  
## 182 87 NA  
## 183 120 NA  
## 184 120 NA  
## 185 23 NA  
## 186 5.51 NA  
## 187 5.69 NA  
## 188 86 NA  
## 189 708,661 NA  
## 190 5.25 NA  
## 191 649,606 NA  
## 192 10000 NA  
## 193 1 NA  
## 194 728,346 NA  
## 195 0 NA  
## 196 100 NA  
## 197 5.57 NA  
## 198 88 NA  
## 199 7,283,465 NA  
## 200 34 NA

new\_heights[c(1:10, 30:80, 100:150),]

## time\_stamp sex height feet inches original  
## 1 2014-09-02 13:40:36 Male 75.00000 NA NA 75  
## 2 2014-09-02 13:46:59 Male 70.00000 NA NA 70  
## 3 2014-09-02 13:59:20 Male 68.00000 NA NA 68  
## 4 2014-09-02 14:51:53 Male 74.00000 NA NA 74  
## 5 2014-09-02 15:16:15 Male 61.00000 NA NA 61  
## 6 2014-09-02 15:16:16 Female 65.00000 NA NA 65  
## 7 2014-09-02 15:16:19 Female 66.00000 NA NA 66  
## 8 2014-09-02 15:16:21 Female 62.00000 NA NA 62  
## 9 2014-09-02 15:16:21 Female 66.00000 NA NA 66  
## 10 2014-09-02 15:16:22 Male 67.00000 NA NA 67  
## 30 2014-09-02 15:16:30 Male 72.00000 NA NA 72  
## 31 2014-09-02 15:16:30 Male 75.00000 NA NA 75  
## 32 2014-09-02 15:16:30 Male 71.00000 NA NA 71  
## 33 2014-09-02 15:16:31 Female 67.00000 NA NA 67  
## 34 2014-09-02 15:16:31 Female 66.00000 NA NA 66  
## 35 2014-09-02 15:16:31 Female 67.00000 NA NA 67  
## 36 2014-09-02 15:16:31 Male 69.00000 NA NA 69  
## 37 2014-09-02 15:16:31 Male 68.00000 NA NA 68  
## 38 2014-09-02 15:16:31 Female 66.75000 NA NA 66.75  
## 39 2014-09-02 15:16:31 Male 72.00000 NA NA 72  
## 40 2014-09-02 15:16:32 Female 63.00000 5 3 5.3  
## 41 2014-09-02 15:16:33 Male 69.00000 NA NA 69  
## 42 2014-09-02 15:16:33 Male 68.00000 NA NA 68  
## 43 2014-09-02 15:16:33 Female 63.00000 NA NA 63  
## 44 2014-09-02 15:16:33 Male 60.00000 NA NA 60  
## 45 2014-09-02 15:16:33 Male 73.00000 NA NA 73  
## 46 2014-09-02 15:16:33 Male 74.00000 NA NA 74  
## 47 2014-09-02 15:16:33 Male 74.00000 NA NA 74  
## 48 2014-09-02 15:16:34 Male 66.00000 NA NA 66  
## 49 2014-09-02 15:16:34 Male 68.00000 NA NA 68  
## 50 2014-09-02 15:16:34 Male 73.00000 NA NA 73  
## 51 2014-09-02 15:16:35 Male 70.00000 NA NA 70  
## 52 2014-09-02 15:16:35 Male 68.00000 NA NA 68  
## 53 2014-09-02 15:16:36 Male 73.00000 NA NA 73  
## 54 2014-09-02 15:16:37 Male 70.50000 NA NA 70.5  
## 55 2014-09-02 15:16:37 Female 64.96063 NA NA 165cm  
## 56 2014-09-02 15:16:37 Male 71.00000 NA NA 71  
## 57 2014-09-02 15:16:37 Male 70.00000 NA NA 70  
## 58 2014-09-02 15:16:38 Male 67.00000 NA NA 67  
## 59 2014-09-02 15:16:38 Male 69.00000 NA NA 69  
## 60 2014-09-02 15:16:38 Male 67.00000 NA NA 67  
## 61 2014-09-02 15:16:39 Male 69.00000 NA NA 69  
## 62 2014-09-02 15:16:39 Male 73.00000 NA NA 73  
## 63 2014-09-02 15:16:40 Male 74.00000 NA NA 74  
## 64 2014-09-02 15:16:41 Male 70.00000 NA NA 70  
## 65 2014-09-02 15:16:41 Male 66.00000 NA NA 66  
## 66 2014-09-02 15:16:41 Male NA NA NA 511  
## 67 2014-09-02 15:16:41 Male 72.00000 NA NA 72  
## 68 2014-09-02 15:16:41 Female 65.00000 NA NA 65  
## 69 2014-09-02 15:16:42 Male 65.00000 NA NA 65  
## 70 2014-09-02 15:16:42 Male 70.00000 NA NA 70  
## 71 2014-09-02 15:16:42 Male 73.00000 NA NA 73  
## 72 2014-09-02 15:16:42 Male 67.00000 NA NA 67  
## 73 2014-09-02 15:16:43 Male 72.00000 NA NA 72  
## 74 2014-09-02 15:16:43 Male 68.00000 NA NA 68  
## 75 2014-09-02 15:16:43 Male 68.00000 NA NA 68  
## 76 2014-09-02 15:16:43 Female 65.00000 NA NA 65  
## 77 2014-09-02 15:16:43 Male 72.00000 NA NA 72  
## 78 2014-09-02 15:16:44 Male 71.00000 NA NA 71  
## 79 2014-09-02 15:16:44 Female 65.00000 NA NA 65  
## 80 2014-09-02 15:16:45 Female 72.00000 NA NA 72  
## 100 2014-09-02 15:16:50 Male 70.00000 NA NA 70  
## 101 2014-09-02 15:16:50 Male 72.00000 NA NA 72  
## 102 2014-09-02 15:16:50 Female 71.00000 NA NA 71  
## 103 2014-09-02 15:16:51 Male 71.00000 NA NA 71  
## 104 2014-09-02 15:16:51 Female 69.60000 NA NA 69.6  
## 105 2014-09-02 15:16:51 Female 65.00000 NA NA 65  
## 106 2014-09-02 15:16:51 Male 70.00000 NA NA 70  
## 107 2014-09-02 15:16:51 Female 61.00000 NA NA 61  
## 108 2014-09-02 15:16:52 Female 63.00000 NA NA 63  
## 109 2014-09-02 15:16:52 Male 70.00000 NA NA 70  
## 110 2014-09-02 15:16:52 Male 67.00000 5 7 5'7  
## 111 2014-09-02 15:16:52 Female 62.00000 NA NA 62  
## 112 2014-09-02 15:16:53 Male 68.00000 NA NA 68  
## 113 2014-09-02 15:16:53 Male 73.00000 NA NA 73  
## 114 2014-09-02 15:16:53 Female 66.50000 NA NA 66.5  
## 115 2014-09-02 15:16:53 Male 69.00000 NA NA 69  
## 116 2014-09-02 15:16:53 Male 74.00000 NA NA 74  
## 117 2014-09-02 15:16:54 Male 71.50000 NA NA 71.5  
## 118 2014-09-02 15:16:54 Male 76.00000 NA NA 76  
## 119 2014-09-02 15:16:54 Male 69.00000 NA NA 69  
## 120 2014-09-02 15:16:54 Male 74.00000 NA NA 74  
## 121 2014-09-02 15:16:55 Male 74.50000 NA NA 74.5  
## 122 2014-09-02 15:16:55 Male 69.00000 NA NA 69  
## 123 2014-09-02 15:16:55 Female 66.00000 NA NA 66  
## 124 2014-09-02 15:16:55 Female 64.00000 NA NA 64  
## 125 2014-09-02 15:16:55 Male 78.00000 NA NA 78  
## 126 2014-09-02 15:16:56 Male NA NA NA >9000  
## 127 2014-09-02 15:16:56 Male 67.00000 5 7 5'7"  
## 128 2014-09-02 15:16:56 Female 69.00000 NA NA 69  
## 129 2014-09-02 15:16:57 Female 67.00000 NA NA 67  
## 130 2014-09-02 15:16:58 Female 63.00000 NA NA 63  
## 131 2014-09-02 15:16:58 Male 74.00000 NA NA 74  
## 132 2014-09-02 15:16:59 Female 62.00000 NA NA 62  
## 133 2014-09-02 15:16:59 Female 69.00000 NA NA 69  
## 134 2014-09-02 15:16:59 Female 64.00000 NA NA 64  
## 135 2014-09-02 15:17:01 Male 71.00000 NA NA 71  
## 136 2014-09-02 15:17:02 Female 62.50000 NA NA 62.5  
## 137 2014-09-02 15:17:02 Male 68.00000 NA NA 68  
## 138 2014-09-02 15:17:02 Female 67.00000 NA NA 67  
## 139 2014-09-02 15:17:03 Male 71.00000 NA NA 71  
## 140 2014-09-02 15:17:03 Male 74.00000 NA NA 74  
## 141 2014-09-02 15:17:05 Male 75.00000 NA NA 75  
## 142 2014-09-02 15:17:06 Female 65.00000 NA NA 65  
## 143 2014-09-02 15:17:06 Male 68.00000 NA NA 68  
## 144 2014-09-02 15:17:07 Female 65.00000 NA NA 65  
## 145 2014-09-02 15:17:07 Female 66.00000 NA NA 66  
## 146 2014-09-02 15:17:07 Male 72.00000 NA NA 72  
## 147 2014-09-02 15:17:08 Male 73.00000 NA NA 73  
## 148 2014-09-02 15:17:08 Male 71.00000 NA NA 71  
## 149 2014-09-02 15:17:08 Male 74.00000 NA NA 74  
## 150 2014-09-02 15:17:09 Female 63.00000 5 3 5'3"

Les données ont été bien nétoyé

### Conclusion

Ce cas d’utilisation a mis montre quelques problèmes qu’on peux rencontrer quand on procéde les chaînes de caractère en r et comment y remedier. Pour purifier les chaînes de charactères, il est très important de maitriser les expression régulier en r qui ne sont qu’un modèle de recherche dans un text. Ces dernier donne un moyen efficace et concise de netoyer les chaine de caractère. Le traitement de chaîne de caractère ne sont pas que les seul aspect de la purification de données, un autre aspect est le traitement des dates, suppression des doublons, traitement des valeurs manquantes etc.