

Python vs R - Data Cleaning

Maxime & Lucas

11/17/2020

1. Importation de fichier

Python

```
import pandas as pd
import numpy as np
import re

dataframe = pd.read_csv("personnes.csv")
#display(dataframe)
```

R

```
dataframe = read.csv("personnes.csv", encoding = "UTF-8")
#dataframe
```

En Python il faut utiliser pandas pour ouvrir un dataframe. Alors que l'utilisation des dataframes est incluse directement dans R

| prenom | email | date_naissance | pays | taille |
|--------|--|----------------|---------------|--------|
| Leila | leila@example.com | 23/01/1990 | France | 1.49m |
| Samuel | samuel_329@example.com | 20/09/2001 | | 1.67m |
| Radia | choupipoune@supermail.eu | 12 sept. 1984 | Côte d'Ivoire | 153cm |
| Marc | marco23@example.com, mc23@supermail.eu | 10/02/1978 | France | 1.65m |
| Heri | helloworld@supermail.eu | 05/03/2008 | Madagascar | 1.34m |
| Hanna | hanna2019@supermail.eu | 01/01/1970 | 24 | 3.45m |
| samuël | samuel_329@example.com | | Bénin | 1.45m |

2. observation des types de données

Python

```
dataframe.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 7 entries, 0 to 6
## Data columns (total 5 columns):
## #   Column      Non-Null Count  Dtype
## ---  -
## 0   prenom      7 non-null     object
## 1   email       7 non-null     object
## 2   date_naissance 6 non-null     object
## 3   pays        6 non-null     object
## 4   taille      7 non-null     object
## dtypes: object(5)
## memory usage: 408.0+ bytes
```

R

```
str(dataframe)
```

```
## 'data.frame':   7 obs. of  5 variables:
## $ prenom      : chr  "Leila" "Samuel" "Radia" "Marc"
## $ email       : chr  "leila@example.com" "samuel_329
## $ date_naissance: chr  "23/01/1990" "20/09/2001" "12 s
## $ pays        : chr  "France" "" "Côte d'Ivoire" "Fr
## $ taille      : chr  "1.49m" "1.67m" "153cm" "1.65m"
```

les fonctions str() et .info() donnent les informations générales.

3. Colonnes noms

Python

```
clean_dataframe = pd.DataFrame() #dataframe pour stocker les nouvelles valeurs et les comparer.

clean_dataframe["prenom"] = dataframe["prenom"].str.replace("ä","e").str.capitalize()
```

On donne à la colonne une propriété "str" et on applique les différentes fonctions

R

```
lower_case = function(value){
  return(tolower(value))
}
dataframe['prenom'] = apply(dataframe['prenom'],1,lower_case)
#dataframe
```

En R il faut d'abord définir les différentes fonctions que l'on va appliquer sur les colonnes du dataframe

| prenom |
|--------|
| Leila |
| Samuel |
| Radia |
| Marc |
| Heri |
| Hanna |
| Samuel |

4. Colonne email

Python

```
liste_mail = dataframe["email"].str.split(",")

clean_email = []
for mail in liste_mail :
    clean_email.append(mail[0])

clean_dataframe["email"] = clean_email
```

```
clean_dataframe["email"] = clean_email
```

On sépare chaque cellule en utilisant la fonction "split()".
 On crée une liste qui va contenir les valeurs propres.
 on boucle sur liste_mail afin de récupérer les valeurs en position 0

R

```
first = function(str){
  str = str[[1]]
  parts = strsplit(str, ',')[[1]]
  first_part = parts[1]
  return(first_part)
}
dataframe['email'] = apply(dataframe['email'], 1, first)
#data
```

En R on crée une fonction qui permet de sélectionner le mail en 1ere position.
 On utilise "apply()" pour appliquer la fonction sur la colonne mail.

| prenom | email |
|--------|--------------------------|
| Leila | leila@example.com |
| Samuel | samuel_329@example.com |
| Radia | choupipoune@supermail.eu |
| Marc | marco23@example.com |
| Heri | helloworld@supermail.eu |
| Hanna | hanna2019@supermail.eu |
| Samuel | samuel_329@example.com |

5. Colonnes date de naissance

Python

```
clean_dataframe["date_naissance"] = pd.to_datetime(dataframe["date_naissance"])
```

R

```
dataframe["date_naissance"] = as.Date(dataframe$date_naissance, "%d/%m/%Y")
```

Que ce soit en Python ou en R il existe une fonction qui traite les données de type "date".
 Pour Python, si on veut changer le format année-mois-jour, il faut utiliser le paramètre "format = \"%d/%m/%Y\"".

| prenom | email | date_naissance |
|--------|--------------------------|----------------|
| Leila | leila@example.com | 1990-01-23 |
| Samuel | samuel_329@example.com | 2001-09-20 |
| Radia | choupipoune@supermail.eu | 1984-09-12 |
| Marc | marco23@example.com | 1978-10-02 |
| Heri | helloworld@supermail.eu | 2008-05-03 |
| Hanna | hanna2019@supermail.eu | 1970-01-01 |
| Samuel | samuel_329@example.com | NA |

6. Colonne pays

Python

```
for pays in dataframe["pays"].unique() :
    print(pays)
```

```
## France
## nan
## Côte d'Ivoire
## Madagascar
## 24
## Bénin
```

```
clean_dataframe["pays"] = dataframe["pays"].replace("24", np.nan)
```

R

```
VALID_COUNTRIES = c('France', 'Côte d'Ivoire', 'Madagascar', 'Bénin', 'Allemagne', 'USA')

check_country = function(country){
  if(! country %in% VALID_COUNTRIES){
    return(NA)
  }
  return =(country)
}

dataframe['pays'] = apply(dataframe['pays'], 1, check_country)
#dataframe
```

Approche Python

On utilise ".unique()" pour retourner les valeurs de la colonne pays sans doublon. Cela nous permettra de trouver les valeurs qui posent problèmes.
 On utilise ensuite la fonction "replace()" sur les fausses valeurs.

Approche R

Approche R

On declare d'abord les noms des pays justes ou valables. On declare la fonction qui permet de remplacer les noms non reconnus comme des pays par un NA.

| prenom | email | date_naissance | pays |
|--------|--------------------------|----------------|---------------|
| Leila | leila@example.com | 1990-01-23 | France |
| Samuel | samuel_329@example.com | 2001-09-20 | NaN |
| Radia | choupipoune@supermail.eu | 1984-09-12 | Côte d'Ivoire |
| Marc | marco23@example.com | 1978-10-02 | France |
| Heri | helloworld@supermail.eu | 2008-05-03 | Madagascar |
| Hanna | hanna2019@supermail.eu | 1970-01-01 | NaN |
| Samuel | samuel_329@example.com | NA | Bénin |

6. Colonne taille

Python

```
taille_cm = []
for taille in dataframe["taille"] :
    clean_height = re.sub("[.,m,c]", "",taille) #re.sub on enleve Les caractères ".",m,c"
    taille_cm.append(clean_height)

clean_dataframe["taille"] = taille_cm

clean_dataframe["taille"] = clean_dataframe["taille"].astype(int)

clean_dataframe.describe()
```

```
##          taille
## count    7.000000
## mean   179.714286
## std    73.767008
## min    134.000000
## 25%    147.000000
## 50%    153.000000
## 75%    166.000000
## max    345.000000
```

```
#remplacement de la valeur aberrante.
clean_dataframe["taille"] = clean_dataframe["taille"].replace(345,np.nan)
```

R

```
#une fonction pour ignorer Les valeurs en Cm et permettant de retirer aussi Les unités 'm'
convert_height = function(height){
  found = regmatches(height, regexpr("[[:digit:]]\\.\\.[:digit:]]{2}m", height))
  if(length(found)==0){
    return(NA)
  }else{
    value = substring(height,1,nchar(height)-1) # on enleve Le dernier caractere, qui est 'm'
    return(as.numeric(value))
  }
}

#Une fonction permettant de remplacer une valeur anormale par la moyenne de la taille de tous Les individus (Hanna qui mesur
ait 3.45m aura une taille qui sera la moyenne des tailles)
fill_height = function(height, replacement){
  if(is.na(height)){
    return(replacement)
  }
  return(height)
}

dataframe["taille"] = apply(dataframe["taille"],1,convert_height)
dataframe["taille"] = apply(dataframe["taille"], 1, function(t) if(!is.na(t) & t<3){t}else{NA}))

#Calcul de la moyenne et application de la fonction fill_height
mean_height = mean(as.numeric(dataframe$taille), na.rm=TRUE)

#Une boucle qui permet de remplacer toutes Les valeurs aberrantes par la moyenne calculée ci-dessus
for(i in 1:nrow(dataframe))
  dataframe[i,"taille"] = fill_height(dataframe[i,"taille"], mean_height)
```

En python on choisi de se débarrasser de la valeur maquante. En R on décide de la remplacer par la moyenne des autres valeurs.

Résultat

| prenom | email | date_naissance | pays | taille |
|--------|--|----------------|---------------|--------|
| Leila | leila@example.com | 23/01/1990 | France | 1.49m |
| Samuel | samuel_329@example.com | 20/09/2001 | NaN | 1.67m |
| Radia | choupipoune@supermail.eu | 12 sept. 1984 | Côte d'Ivoire | 153cm |
| Marc | marco23@example.com, mc23@supermail.eu | 10/02/1978 | France | 1.65m |
| Heri | helloworld@supermail.eu | 05/03/2008 | Madagascar | 1.34m |
| Hanna | hanna2019@supermail.eu | 01/01/1970 | 24 | 3.45m |
| samuël | samuel_329@example.com | NaN | Bénin | 1.45m |

| prenom | email | date_naissance | pays | taille |
|--------|--------------------------|----------------|---------------|--------|
| Leila | leila@example.com | 1990-01-23 | France | 149 |
| Samuel | samuel_329@example.com | 2001-09-20 | NaN | 167 |
| Radia | choupipoune@supermail.eu | 1984-09-12 | Côte d'Ivoire | 153 |
| Marc | marco23@example.com | 1978-10-02 | France | 165 |

| | | | | |
|--------|--|------------|------------|-----|
| Heri | helloworld@supermail.eu | 2008-05-03 | Madagascar | 134 |
| Hanna | hanna2019@supermail.eu | 1970-01-01 | NaN | NaN |
| Samuel | samuel_329@example.com | NA | Bénin | 145 |

Liens utiles

python

jeu de données : <https://openclassrooms.com/fr/courses/4525266-decrivez-et-nettoyez-votre-jeu-de-donnees/4928126-tp-nettoyez-votre-jeu-de-donnees>.

Pandas : <https://pandas.pydata.org/docs/>

Numpy : <https://numpy.org/doc/>

regex : <https://docs.python.org/3/library/re.html>

Matplotlib : <https://matplotlib.org/>

R

jeu de données : <https://openclassrooms.com/fr/courses/4525266-decrivez-et-nettoyez-votre-jeu-de-donnees/4928126-tp-nettoyez-votre-jeu-de-donnees>

Functions : https://www.tutorialspoint.com/r/r_functions.htm

Learning data preparation and cleaning in R : <https://bookdown.org/aschml11/RESMHandbook/data-preparation-and-cleaning-in-r.html>