Data Test : BIFORA

ROBERT Wendy

18 août 2025

Table des matières

Introduction

L'objectif de ce travail est d'analyser et de préparer un jeu de données issu d'un fichier Excel fourni. Ce fichier comporte deux feuilles principales (`Data_1` et `Data_2`) décrivant des informations sur une base de clients, ainsi qu'une troisième feuille (`TYPES VARIABLES`) servant de dictionnaire de données.

1 Importation des données

1.1 Python

Lanalyse débute par limportation du fichier Excel `data_test_BIFORA.xlsx` contenant trois feuilles de données : `Data_1`, `Data_2` et `TYPES VARIABLES`. Cette opération est réalisée en Python grâce à la bibliothèque `pandas`, qui permet de lire directement des fichiers Excel.

1.1.1 Chargement des bibliothèques

Nous commençons par importer la bibliothèque `pandas` :

```
import pandas as pd
```

Listing 1 – Importation bibliothèque Pandas

1.1.2 Lecture du fichier Excel

Ensuite, nous spécifions le chemin du fichier et chargeons les trois feuilles de données :

```
# Importation des 3 feuilles
path = r"/Users/wendyr/Downloads/data_test_BIFORA (2) (1).xlsx"
data1 = pd.read_excel(path, sheet_name="Data_1")
data2 = pd.read_excel(path, sheet_name="Data_2")
dict_vars = pd.read_excel(path, sheet_name="TYPES VARIABLES ")
```

Listing 2 – Importation du fichier Excel

Une fois les données importées, nous pouvons examiner leurs dimensions, leurs colonnes et les premières lignes afin de comprendre leur structure :

Cette étape constitue la base de toute l'analyse, car elle permet de vérifier que les données sont bien accessibles et lisibles par Python avant de passer au diagnostic et au nettoyage.

1.1.3 Préparation et nettoyage des données

Une fois les données importées, il est nécessaire d'effectuer plusieurs opérations de nettoyage afin d'obtenir un jeu de données cohérent et exploitable.

1.1.4 Renommage et nettoyage des colonnes

Certaines variables contiennent des erreurs de frappe ou des espaces parasites dans leur nom. Nous commençons donc par corriger ces problèmes.

```
# Renommer correctement la variable identifiant
data1 = data1.rename(columns={"identiffiant ": "identifiant"})

# Nettoyer les noms des variables (suppression des espaces)
data1.columns = data1.columns.str.strip()
data2.columns = data2.columns.str.strip()
dict_vars["VARIABLES"] = dict_vars["VARIABLES"].str.strip()
```

Listing 3 – Correction des noms de variables

1.1.5 Conversion automatique des variables

À laide du dictionnaire de données (`TYPES VARIABLES`), nous convertissons les variables quantitatives en numérique. Lorsque la conversion échoue (exemple : valeur texte dans une colonne numérique), la valeur est transformée en `NaN` (valeur manquante).

```
for _, row in dict_vars.iterrows():
    var = row["VARIABLES"]
    if var in data1.columns:
        if row["Quali / Quanti "] == "Quanti":
            data1[var] = pd.to_numeric(data1[var], errors="coerce")
    if var in data2.columns:
        if row["Quali / Quanti "] == "Quanti":
            data2[var] = pd.to_numeric(data2[var], errors="coerce")
```

Listing 4 – Conversion des variables selon le dictionnaire

1.1.6 Diagnostic des données

Nous définissons une fonction `diagnostic` qui permet d'afficher rapidement des informations essentielles : types des variables, valeurs manquantes et doublons.

```
def diagnostic(df, name):
    print("\n--- Diagnostic "+name+" ---")
    print("Types de variables :\n", df.dtypes)
    print("\nValeurs manquantes :\n", df.isna().sum())
    print("\nDoublons :", df.duplicated().sum())

diagnostic(data1, "Data_1")
diagnostic(data2, "Data_2")
```

Listing 5 – Diagnostic des bases

1.1.7 Suppression des doublons et fusion des bases

Enfin, nous supprimons les doublons éventuels puis fusionnons les deux jeux de données sur la variable `identifiant`, qui n'aurait pas été possible sans la correction du nom de la variable dans `Data_1`.

```
# Suppression des lignes identiques
data1 = data1.drop_duplicates()
data2 = data2.drop_duplicates()
```

```python
# Fusion des deux jeux de données
merged = data1.merge(data2, on="identifiant", how="inner")

print("\nDimensions Data_1 :", data1.shape)
print("\nDimensions Data_2 :", data2.shape)
print("\nDimensions après fusion :", merged.shape)
```

Listing 6 – Nettoyage et fusion des données

1.2 SAS

En parallèle de Python, les données ont également été importées dans SAS afin de permettre une exploration complémentaire et des traitements statistiques.

1.2.1 Définition de la librairie et importation

Tout d'abord, une librairie pointant vers le fichier Excel a été définie. L'instruction `proc import` a ensuite été utilisée pour lire directement les feuilles `Data_1` et `Data_2` :

```
libname x xlsx '/home/u64300920/data_test_BIFORA (2) (1).xlsx';

/* Import feuille Data_1 */
proc import datafile='/home/u64300920/data_test_BIFORA (2) (1).xlsx'
    out=x.data1
    dbms=xlsx
    replace;
    sheet="Data_1";
    getnames=yes;
run;

/* Import feuille Data_2 */
proc import datafile='/home/u64300920/data_test_BIFORA (2) (1).xlsx'
    out=x.data2
    dbms=xlsx
    replace;
    sheet="Data_2";
    getnames=yes;
run;
```

Listing 7 – Importation des données dans SAS

1.2.2 Conversion des variables mal typées

Lors de l'importation, certaines variables continues (par exemple `Epargne_financiere`) ont été interprétées comme du texte. Afin de pouvoir effectuer des analyses statistiques correctes, elles ont été converties en variables numériques à l'aide de la fonction `input()` :

```
/* Création d'une nouvelle version de Data_1 avec conversion */
data x.data1_num;
    set x.data1;
    Epargne_financiere_num = input(Epargne_financiere, best32.);
    drop Epargne_financiere;
```

```
    rename Epargne_financiere_num = Epargne_financiere;
run;
```

Listing 8 – Conversion de variables texte en numérique

Cette étape garantit que les montants financiers pourront être utilisés dans des calculs (sommes, moyennes, médianes, etc.).

1.2.3 Contrôle de l'importation

Enfin, une inspection des structures des tables a été réalisée avec la procédure `proc contents`, afin de vérifier la nature des variables importées (caractère ou numérique) :

```
proc contents data=x.data1;
    title "Structure du dataset DATA_1";
run;

proc contents data=x.data2;
    title "Structure du dataset DATA_2";
run;
```

Listing 9 – Inspection des tables SAS

Cette première phase dans SAS permet donc d'assurer la bonne lecture des données et la cohérence des formats avant de passer à l'analyse descriptive et au rapprochement des deux jeux de données.