Calyxis
PÔLE D'EXPERTISE DU RISQUE

Le 07/06/2024

Document technique :

AcciStat

Réalisé par :

- Leslie PLANET
- Xavier BARBEAU
- Hélène VISOZO
- Rémi PIERRON
- Chloé DECOUST

Encadré par :

- M. BUREAU
- M. GARNIER
- Mme. CANONNE

Table des matières

I. Installation 2
II. Revue de l'interface utilisateur 2
1. Visualisation 2
2. Ajouter/Modifier 3
3. Datavisualisation et tutoriel 6
III. Fonctionnalité 6
1. Collecte et traitement des données 6
2. Analyse Statistique 8
Tutoriel d'utilisation 10

I. Installation

Pour télécharger et utiliser le logiciel, nous vous recommandons d'être sous Windows et d'avoir une version récente de Python.

Récupérez le fichier compressé fourni avec ce manuel et décompressez-le où vous souhaitez utiliser l'application.

Ensuite, assurez-vous d'avoir déjà d'installé les bibliothèques python. S'il vous en manque, vous pouvez les télécharger simplement avec la commande console « pip install bibliothèque » (pensez à remplacer la « bibliothèque » par le nom de cette dernière que vous souhaitez installer).

Voici les différentes bibliothèques utilisées :

- CSV
- Pandas
- Os
- Openpyxl
- Scipy.stats
- Chardet
- Tkinter
- PIL
- Webbrowser

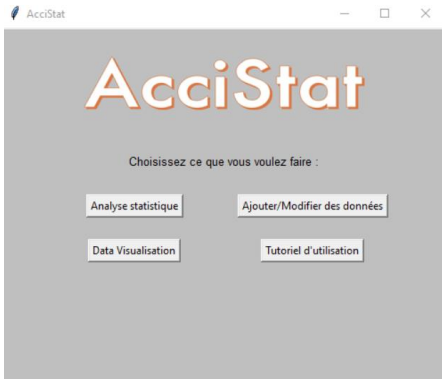
Une fois toutes les bibliothèques installées, penser à redémarrer le noyau de par votre application d'exécution de python.

Vous pouvez maintenant exécuter le programme et utiliser l'application.

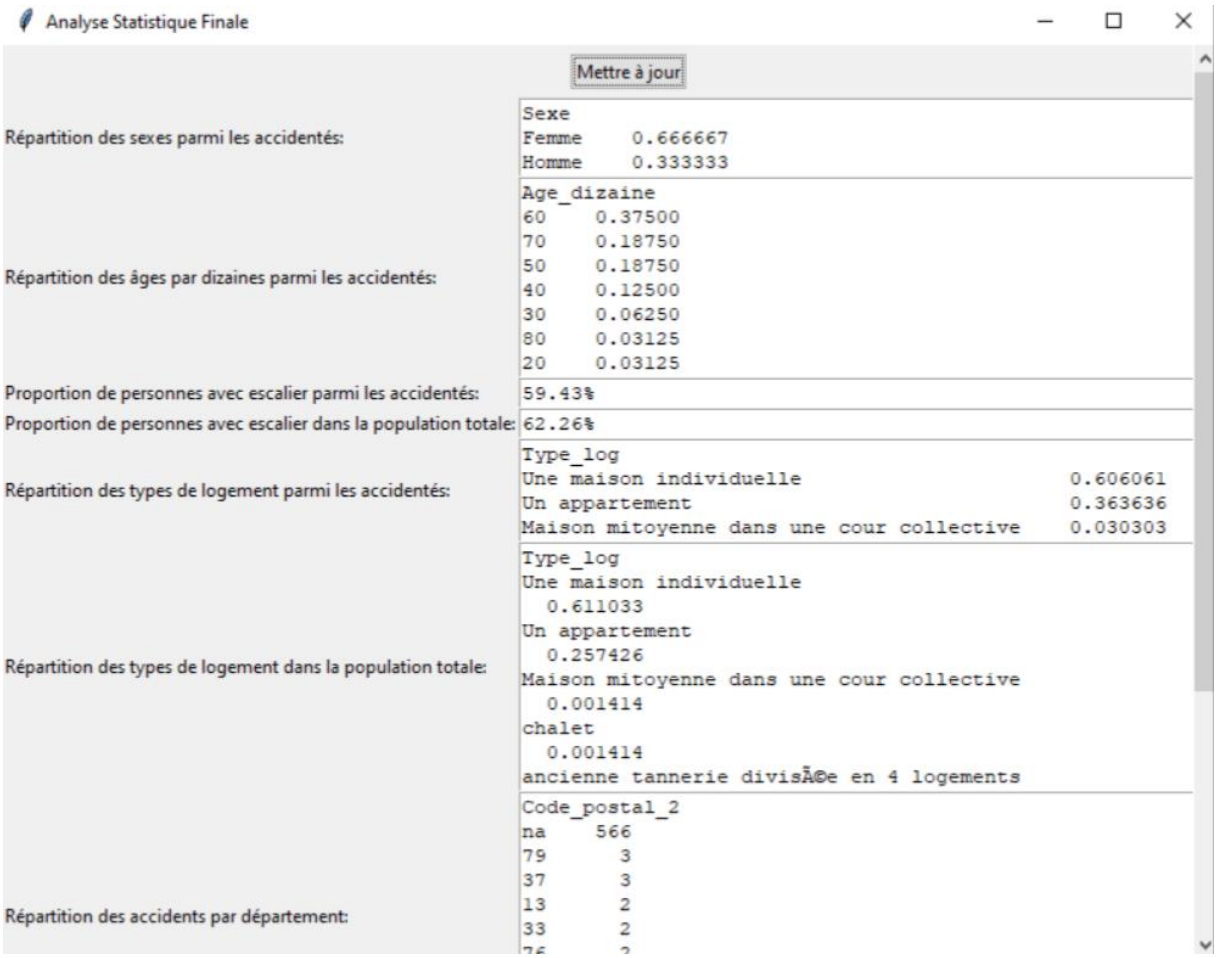
II. Revue de l'interface utilisateur

1. *Visualisation*

Le premier bouton « Visualisation » placé en haut à gauche nous permet de visualiser en détail les résultats des calculs effectués dans le fichier traitement.py dont nous avons parlé précédemment, comme la répartition par sexe des accidentés, la répartition des âges par dizaine des



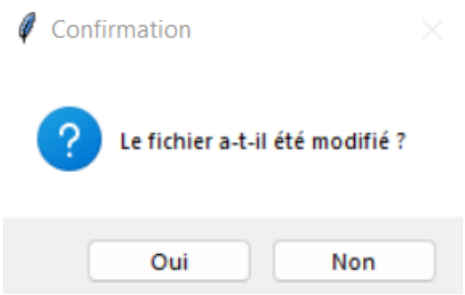
accidentés ou encore la proportion des personnes avec escaliers parmi les accidentés. À l'aide de la barre de défilement placé sur la droite de la fenêtre, nous pouvons également voir les résultats du test du Khi-deux en fonction des accidents et du sexe des accidentés. Voici à quoi ressemble cette fenêtre. Il suffit d'appuyer sur le bouton « Mettre à jour » en haut pour que les résultats s'affichent.



2. *Ajouter/Modifier*

Si nous revenons sur la page d'accueil et cliquons maintenant sur le bouton « Ajouter/Modifier des données », nous allons atterrir sur cette page :

Il suffit tout d'abord de cliquer sur le petit dossier jaune afin de sélectionner le fichier à ajouter ou modifier. Si vous souhaitez l'ajouter, il vous suffit de cliquer, ensuite, sur le bouton « Ajouter ». Une fenêtre vous demandant si le fichier a été modifié va apparaître. En fonction de la réponse, notre programme va faire en sorte de modifier le fichier pour qu'il corresponde à nos attentes.



Tout d'abord, les chemins des fichiers sont stockés dans deux variables différentes fichier1 et fichier2. La variable « modif » correspond à si oui ou non le fichier a été modifié lorsque l'on souhaite l'ajouter. Et les variables modif1 et modif2 correspondent aux noms des fichiers quand on souhaite modifier. Ces variables sont des booléens.

```
# Variable :
fichier1 = ""  # Chemin du premier fichier
fichier2 = ""  # Chemin du deuxième fichier
modif = False  # Stocker si le fichier "ajouter" a été modifié
modif1 = False  # Stocker si le premier fichier du "modifier" a été modifié
modif2 = False  # Sstocker si le deuxième fichier du "modifier" a été modifié
fichier_csv = 'accident.csv' # Nom du notre fichier csv
feuille='Accident' # Nom de la feuille excel (à récupérer sur le fichier excel)
```

Voici la fonction qui va nous permettre d'ajouter un fichier :

```
# Fenetre pour le bouton ajout
def fenetre_ajout():
    def parcourir_fichier_ajout():
        global fichier1  # Utilisation de la variable globale fichier1
        chemin_fichier = filedialog.askopenfilename()
        if chemin_fichier:
            fichier1 = chemin_fichier  # Stockage du chemin du fichier sélectionné dans la variable fichier1
            label_chemin_ajout.config(text=fichier1)
            nom_fichier1 = obtenir_nom_fichier(fichier1)
        if modif == FALSE:
            traitement.convertisseur(nom_fichier1, fichier_csv, feuille)
            accListe, nomCol = traitement.ouverture(fichier_csv)
            traitement.modification(accListe, nomCol)
        else:
            traitement.renommer_en_accident_traite(fichier1)
```

```
def ouvrir_popup_ajouter():
    global modif
    if not fichier1:
        messagebox.showwarning("Attention", "Veuillez d'abord sélectionner un fichier.")
        return

    reponse = messagebox.askyesno("Confirmation", "Le fichier a-t-il été modifié ?")
    modif = reponse  # Stocker la réponse dans la variable modif
    if reponse:
        messagebox.showinfo("Info", "Vous avez indiqué que le fichier a été modifié.")
    else:
        messagebox.showinfo("Info", "Vous avez indiqué que le fichier n'a pas été modifié.")
```

Tout d'abord, on vérifie que l'utilisateur a choisi un fichier. Le même processus sera utilisé lorsque l'on souhaite modifier un fichier.
La partie de code en rouge est ce qui va faire en sorte que le fichier soit ajoutée en fonction de s'il a été modifié ou non. S'il a été modifié, le fichier va être converti et ouvert puis modifié à l'aide des fonctions 'convertisseur', 'ouverture' et 'modification' qui ont été expliquées dans la partie « Collecter et traiter » de ce rapport. Si le fichier n'a pas été modifié, il va simplement être renommé à l'aide de la fonction « renommer_en_accident_traite ».

Maintenant, si l'on souhaite modifier le fichier, un deuxième fichier sera sollicité. Ces deux fichiers passeront les mêmes vérifications pour savoir s'ils ont été modifiés ou non.

```
def parcourir_fichier_modifier():
    global fichier2, modif2
    chemin_fichier = filedialog.askopenfilename()
    if chemin_fichier:
        fichier2 = chemin_fichier  # Stockage du chemin du fichier sélectionné dans la variable fichier2
        label_chemin_modif.config(text=fichier2)
        reponse = messagebox.askyesno("Confirmation", "Le fichier modifié a-t-il été modifié ?")
        modif2 = reponse  # Stocker la réponse pour le deuxième fichier dans la variable modif2
        nom_fichier2 = obtenir_nom_fichier(fichier1)
        nom_fichier2 = obtenir_nom_fichier(fichier2)
        if reponse:
            messagebox.showinfo("Info", "Vous avez indiqué que le fichier a été modifié.")
        else:
            messagebox.showinfo("Info", "Vous avez indiqué que le fichier n'a pas été modifié.")
        traitement.concatener(fichier1, modif1, fichier2, modif2)
```

Dans la fonction ci-dessus, si le fichier n'a pas été modifié on fait appel à la fonction 'concatener' qui a été expliquée auparavant.

3. Datavisualisation et tutoriel

Les deux boutons restant renvoient vers notre Power BI ainsi que notre tutoriel d'utilisation. Nous vous invitons à les regarder autant de fois que nécessaire pour visualiser les résultats ou tout simplement le tutoriel d'utilisation du TKinter.

```
# Commande pour ouvrir le fichier Power BI (Dans le même répèrtoire)
def powerbi():
    os.startfile("acciStat.pbix")

# Commande pour ouvrir une page web (notre tuto youtube)
def tuto():
    webbrowser.open("https://www.youtube.com")
```

III. Fonctionnalité

1. Collecte et traitement des données

Pour pouvoir analyser les données, nous les avons tout d'abord collectés et traités. Pour la collecte de données, nous avons tout d'abord récupéré le nom du fichier à l'aide de la fonction 'obtenir_nom_fichier'. Puis, nous avons créé une procédure convertisseur qui permet de convertir un fichier xlsx en fichier csv, car le traitement d'un fichier csv est plus simple.

```
def convertisseur(fichier_xlsx, fichier_csv, feuille):
    # Ouvrir le fichier Excel
    wb = openpyxl.load_workbook(fichier_xlsx)
    ws = wb[feuille]

    # Ouvrir le fichier CSV en écriture
    with open(fichier_csv, mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file, delimiter=';')

        # Itérer sur les lignes de la feuille Excel
        for row in ws.iter_rows(values_only=True):
            # Convertir les valeurs en int si elles sont des float sans partie décimale
            new_row = [int(cell) if isinstance(cell, float) and cell.is_integer() else cell for cell in row]
            # Écrire la ligne dans le fichier CSV
            writer.writerow(new_row)
```

Pour continuer, nous avons créé une fonction ouverture qui permet d'ouvrir le fichier et de le transformer en liste de liste, il permet également de séparer dans une autre liste le nom des colonnes, toujours dans l'optique de facilité l'analyse.

```python
# Fonction pour ouvrir le fichier et le transformer en liste de listes
def ouverture(fichier):
    accListe = []
    nomCol = []
    try:
        with open(fichier, newline='', encoding='utf-8') as csvfile:
            liste = csv.reader(csvfile, delimiter=';')
            for i, ligne in enumerate(liste):
                if i == 0:
                    nomCol = list(ligne)  # Garde dans une liste l'en-tête avec le nom des colonnes
                else:
                    accListe.append(list(ligne))
    except FileNotFoundError:
        print("Fichier introuvable !")
    return accListe, nomCol
```

En ce qui concerne le traitement des données, nous avons créé une procédure 'modification' qui permet de fusionner les différents lieux d'accident qui sont repartis sur plusieurs colonnes dans notre jeu de données et de supprimer les colonnes qui sont vides.

```python
# Fonction de modification des colonnes telles que : "précisez le lieux ..."
def modification(accListe, nomCol):
    nouvelle_accListe = []
    for ligne in accListe:
        if ligne[2] == 'Oui':
            lieuPrecis = ''.join(ligne[10:20])
            activitePratique = ''.join(ligne[21:27])
            sportPratique = ''.join(ligne[29:45])
            nv=ligne[3]
            ligne[3]=nv[:10]
            ligne[10] = lieuPrecis
            ligne[21] = activitePratique
            ligne[29] = sportPratique

        nouvelle_accListe.append(ligne)

    # Suppression des colonnes inutiles et mise à jour des noms de colonnes
    for ligne in nouvelle_accListe:
        del ligne[11:20]
        del ligne[13:18]
        del ligne[15:30]

    nomCol[10] = "LieuPrécis"
    nomCol[21] = "ActivitéPratiquée"
    nomCol[29] = "SportPratiqué"

    del nomCol[11:20]
    del nomCol[13:18]
    del nomCol[15:30]
```

```
fichier_sortie = "accident_traite.csv"

# Ajouter les noms de colonnes à la première ligne de la nouvelle liste
nouvelle_accListe.insert(0, nomCol)

# Réécriture dans un fichier csv pour export
with open(fichier_sortie, 'w', newline='', encoding='UTF-8') as csvfile:
    writer = csv.writer(csvfile, delimiter=';')
    for ligne in nouvelle_accListe:
        writer.writerow(ligne)

print("Le fichier CSV a été créé avec succès !")
```

Ensuite, nous avons écrit la procédure 'concaténer' qui est utilisée lorsque l'on souhaite ajouter un nouveau fichier à la base. En effet, elle permet d'ajouter à la suite de notre fichier existant de nouvelle ligne extraite d'un fichier CSV.

Pour finir le traitement de nos données, nous avons créé la fonction 'renommer_en_accident_traite' qui permet de renommer le fichier qui contient les informations sur les accidents en 'traiter en accidents_traite.csv'. Elle prend en entrée un chemin du fichier.

```
def renommer_en_accident_traite(chemin_fichier):
    """
    Cette fonction prend un chemin de fichier en entrée et renomme ce fichier en 'accident_traite.csv'.
    """
    try:
        dossier = os.path.dirname(chemin_fichier)
        nouveau_nom = os.path.join(dossier, 'accident_traite.csv')
        os.rename(chemin_fichier, nouveau_nom)
        print("Le fichier a été renommé avec succès.")
        return nouveau_nom
    except Exception as e:
        print(f"Erreur lors du renommage du fichier : {e}")
        return None
```

2. Analyse Statistique

Nous avons décidé de faire une jointure entre nos deux tables, « indiv.csv » et « accident_traite.csv » pour pouvoir mettre en relation les informations présentent dans les deux. Pour ce faire, nous avons écrit une procédure : « jointure ».

```python
def jointure():

    convertisseur("MAVIE_BD_STID_mai_2024.xlsx", "indiv.csv", "BD_3quest")
    convertisseur("MAVIE_BD_STID_mai_2024.xlsx", "accident.csv", feuille)
    accListe, nomCol=ouverture("accident.csv")
    modification(accListe, nomCol)

    # Charger les fichiers CSV dans des DataFrames en utilisant l'encodage iso-8859-1
    df_accidents = pd.read_csv("accident_traite.csv", delimiter=";", encoding="iso-8859-1")
    df_indiv = pd.read_csv("indiv.csv", delimiter=";", encoding="iso-8859-1")

    # Effectuer la jointure entre les deux DataFrames en utilisant la colonne Id_volontaire
    df_joint = pd.merge(df_accidents, df_indiv, on="Id_volontaire")

    # Afficher les premières lignes du DataFrame résultant
    print(df_joint.head())

    # Enregistrer le DataFrame résultant dans un nouveau fichier CSV
    df_joint.to_csv("data.csv", index=False, sep=';')
```

Ensuite, pour chaque calcul statistique nous avons écrit une fonction qui renvoie le résultat de chaque calcul, comme ci-dessous :

```python
# calcul la part des accidentés par tranche d'âge de 10 ans
def calculate_age_parts(df):
    df_acc = df[df['Acc'] == 'Oui']
    df_acc.loc[:, 'Age_dizaine'] = (df_acc.loc[:, 'Age_actuel'] // 10) * 10
    df_acc = df_acc[df_acc['Age_dizaine'] != 0]
    age_counts = df_acc['Age_dizaine'].value_counts()
    age_parts = age_counts / df_acc.shape[0]
    age_parts_sorted = age_parts.sort_values(ascending=False)
    return age_parts_sorted
```

Nous en avons aussi fait une pour faire un test du khi-deux d'indépendance :

```python
# test du khi-deux pour voir si il y a indépendance ou non entre avoir un accident et le genre
def calculer_khi_deux(df, colonne1, colonne2):
    if colonne1 not in df.columns or colonne2 not in df.columns:
        raise ValueError("Colonne non trouvée dans le DataFrame.")
    if not isinstance(df, pd.DataFrame):
        raise TypeError("df doit être un DataFrame pandas.")
    table_cont = pd.crosstab(df[colonne1], df[colonne2])
    khi2_cal, p_val, ddl, effectif_theorique = chi2_contingency(table_cont)
    resultats = {
        'statistique_khi_deux': khi2_cal,
        'p_valeur': p_val,
        'degres_de_liberte': ddl,
        'effectifs_theoriques': effectif_theorique,
        'table_contingence': table_cont
    }
    return resultats
```

Nos fonctions renvoient toutes le résultat des calculs pour pouvoir les insérer dans l'interface Tkinter.

Tutoriel d'utilisation

Vous pouvez retrouver une tutoriel vidéo en suivant ce lien :
https://youtu.be/XOiF5RKzbGM?si=OiNEz2w0UuXKBLWM