Fil Rouge Documentation

Version 1.0

PAILLEUX Jean-Didier - ZEMNI Malek - KLOTZ Sonny

mai 28, 2017

Table des matières

CHAPITRE 1

Description

Bienvenue sur la documentation technique du projet Fil Rouge.

Dans ce document se trouvent les documentations des éléments de code source de l'application :

- Une API de chargement des données
- Une API d'analyse descriptive des données
- Une interface web exploitant les API

En plus de ce travail, un script `demoAPI.py` est mis à disposition.

> Le développement de ce script est motivé par la volonté de donner un exemple d'utilisation des API en passant par l'automatisation de tâches qui seraient lourdes via l'interface web seule.
>
> Mais aussi, il permet de fournir en sortie un compte-rendu statistique réutilisable sur le graphe d'où proviennent les données.

Ainsi, cette documentation fait également part d'une page de description de ce script, de son utilisation et de ses résultats.

CHAPITRE 2

Lancer l'application

2.1 Avant de commencer

Cette application fonctionne à l'aide d'outils à installer préalablement :

- Python 3+ : langage de programmation principal.
- Flask 0.12.x : framework d'applciations web en Python.
- Sphinx 1.6.1 : génération automatique de documentation.
- *sphinx_rtd_theme, et guzzle_sphinx_theme* : templates pour sphinx

2.2 Applet : interface web

L'instruction qui permet de lancer l'appli est :

```
make run
```

2.3 Tests unitaires

Exécuter tous les tests :

```
make test
```

Exécuter les tests du module interface_web :

```
make test-interface_web
```

Exécuter les tests du module chargement_des_donnees :

```
make test-chargement_des_donnees
```

Exécuter les tests du module add :

```
make test-add
```

2.4 Génération et lancement de la documentation

Générer et lancer la documentation au format html :

```
make doc-html
```

Générer et lancer la documentation au format pdf :

```
make doc-latexpdf
```

CHAPITRE 3

Documentation

3.1 Interface web

3.1.1 gestionFlux.py

interface_web.gestionFlux.**fenetre_choix_fichier**()
Affiche le template « choix_fichier.html » lorsque la requette HTTP « / » est indiquée.
Vide le dossier "uploads/" avec la fonction `removeFiles`

Retourne le rendu du template `choix_fichier.html`

interface_web.gestionFlux.**fenetre_role_choix_colonne**(*file*)
Affiche le template « role_choix_colonne.html » lorsque la requette HTTP « /fenetre_role_choix_colonne/ » est indiquée.
Le template affiche un message d'erreur si le fichier `file` n'est pas valide, sinon elle affiche son contenu.

Paramètres **file** (*str*) – représente le nom du fichier chargé.

Retourne le rendu du template `role_choix_colonne.html`

interface_web.gestionFlux.**fenetre_resultat_ADD**(*file*)
Affiche le template « resultat_ADD.html » lorsque la requette HTTP « /fenetre_resultat_ADD/ » est indiquée.
Avant d'envoyer la page web, les analyses descriptives sur les colonnes sont effectuées pour renseigner les données `json` nécessaires.

Retourne le rendu du template `resultat_ADD.html`

interface_web.gestionFlux.**remove**(*file*)
Supprime le fichier `file` du dossier `uploads`.

Paramètres **file** (*str*) – nom du fichier `.csv`

Retourne redirige vers la route `index`

interface_web.gestionFlux.**removeFiles**()
Vide le dossier `uploads` contenant les fichiers chargé.

interface_web.gestionFlux.**manuel**()
Affiche le template manuel.html » lorsque la requette HTTP « /manuel/ » est indiquée.

Ce template contiend le pdf du manuel utilisateur.

Retourne le rendu du template `manuel.html`

interface_web.gestionFlux.**sauvegardeResultats**(*file*)
Sauvegarde les résultats de l'analyse descriptives dans un fichier `.csv` et lance son téléchargement.

Retourne téléchargement du fichier `Resultats.csv`.

3.1.2 choixFichier.py

interface_web.choixFichier.**FileWithSGF**()
Upload d'un fichier lors du parcours dans le système de gestion des fichiers.

Retourne redirection de la page web vers `fenetre_rôle_choix_colonne` avec comme parmètre le chemin du fichier upload.

interface_web.choixFichier.**FileWithDragDrop**()
Upload d'un fichier après avoir déposé ce fichier dans la zone de Drag&Drop.

Retourne redirection de la page web vers `fenetre_rôle_choix_colonne` avec comme paramètre le chemin du fichier upload.

3.1.3 addRoutes.py

Description

Les fonctions de ce fichiers servent à répondre aux requêtes client pour récupérer des fichiers sur le serveur selon le modèle `ajax`.

En effet, après la requête, le serveur n'envoie pas toute une page web, mais seulement les données qui intéressent le navigateur.

Cette technique permet de communiquer plus efficacement les données entre le client et le serveur.

De plus, le format de fichier `json` est un format léger, et est donc très adapté pour ce cas de figure.

Les fonctions

interface_web.addRoutes.**iStats**()
Echange ajax entre le serveur et la page web pour envoyer le compte-rendu statistique.

Retourne dictionnaire convertit en objet javascript contenantt les informations de "stats.js"

interface_web.addRoutes.**timeSeries**()
Echange ajax entre le serveur et la page web pour envoyer les données des séries temporelles.

Retourne dictionnaire convertit en objet javascript contenantt les informations de "timeSeries.js"

interface_web.addRoutes.**distribution**()
Echange ajax entre le serveur et la page web pour envoyer les données des ditributions.

Retourne dictionnaire convertit en objet javascript contenantt les informations de "distribution.js"

interface_web.addRoutes.**distributionCumulative**()
Echange ajax entre le serveur et la page web pour envoyer les données des ditributions cumulatives.

Retourne dictionnaire convertit en objet javascript contenantt les informations de "distributionCumulative.js"

interface_web.addRoutes.**boxplot**()
Echange ajax entre le serveur et la page web pour envoyer les données des boîtes de Tukey.

Retourne dictionnaire convertit en objet javascript contenantt les informations de "boxplot.js"

3.2 Analyse descriptive de données

3.2.1 addQualitatives.py

add.addQualitatives.**nbElemListeCouple**(*listeEffectifs*)
Calcul l'effectif total des données de `listeEffectifs`.

La fonction se charge simplement de calculer l'effectif total des données contenu dans `listeEffectifs` en sommant l'effectif de chaque tuple : `couple[1]`.

Paramètres **listeEffectifs** – liste de tuples (donnée, effectif)

Retourne l'effectif total des données de listeEffectifs

Type retourné int

add.addQualitatives.**calculEffectifs**(*listeDonnees*)
Calcule l'effectifs pour chaque données contenu dans `listeDonnees`.

La fonction prend en entrée une liste contenant les données à analyser. Elle calculera les effectifs pour chaque valeur à l'aide d'un dictionnaire.

Ce dictionnaire sera converti en liste de tuples et un tri sera effectué pour ordonner les tuples.

Paramètres **listeDonnees** – liste contenant les données à analyser

Retourne liste de tuples (donnée, effectif)

Type retourné list

add.addQualitatives.**calculEffectifsCumules**(*listeEffectifs*)
Calcule les effectifs cumulés avec l'aide de `listeEffectifs`.

La fonction prend en entrée la liste des effectifs. Elle calculera dans une nouvelle liste les effectifs cumulés à partir de `listeEffectifs` en remplaçant l'effectif par l'effectif cumulé correspondant.

Paramètres **listeEffectifs** – liste de tuples (donnée, effectif)

Retourne liste de tuples (donnée, effectif cumulé)

Type retourné list

add.addQualitatives.**calculFrequences**(*listeEffectifs*)
Calcule les fréquences d'apparition des valeurs.

La fonction prend en entrée la liste des effectifs. Elle calculera dans une nouvelle liste la fréquence à partir de `listeEffectifs` en divisant l'effectif par la taille du jeu de données.

Paramètres **listeEffectifs** – liste de tuples (donnée, effectif)

Retourne liste de tuples (donnée, frequence)

Type retourné list

add.addQualitatives.**calculFrequencesCumulees**(*listeEffectifsCumules*)
Calcule les fréquences cumulées.

La fonction prend en entrée la liste des effectifs cumulés. Elle calculera dans une nouvelle liste la fréquence à partir de `listeEffectifsCumules` en divisant l'effectif cumulé par l'effectif total.

Paramètres **listeFrequences** – liste de tuples (donnée, frequence)

Retourne liste de tuples (donnée, frequence cumulée)

Type retourné list

add.addQualitatives.**infoSecteurs**(*listeFrequences*)
Création d'un dictionnaire pour les informations d'un diagramme de secteurs.

Couples (clé, valeur) :
— donnée1 : fréquence d'apparition de la donnée 1
— donnée2 : fréquence d'apparition de la donnée 2

Il y a autant d'éléments que de données distinctes.

Paramètres **listeFrequences** – liste de tuples (donnée, fréquence)

Type retourné dict

add.addQualitatives.**infoHistogramme**(*listeEffectifs*)
Création d'un dictionnaire pour les informations d'un histogramme (diagramme en batons).

Couples (clé, valeur) :
— « x » : liste des abscisses, les données
— « value » : liste des ordonnées, les effectifs respectifs

Paramètres **listeEffectifs** – liste de tuples (donnée, effectif)

Type retourné dict

3.2.2 addQuantitativesContinues.py

add.addQuantitativesContinues.**discretisation**(*nombreClasses*, *donneesContinues*)
Discrétise des données continues du paramètre `donneesContinues`.
La fonction se charge de décomposer l'étendue `[min ; max]` de l'ensemble de données en `nombreClasses` intervalles de même étendue.
Ensuite de remplacer les occurrences des données par l'intervalle auquel la donnée appartient.

Paramètres **donneesContinues** – liste de nombres flottants

Retourne liste d'intervalles, et étendue discrétisée

add.addQuantitativesContinues.**calculNombreClasses**(*donneesContinues*)
Calcule le nombre de classes nécessaire à une discrétisation selon la règle de Sturges.

Retourne nombre de classes utilisé pour la discrétisation des valeurs

Type retourné int

add.addQuantitativesContinues.**preparationIntervallesAnalyse**(*listeIntervalles*)
Prépare les données pour l'utilisation des éléments de calcul du module `addQuantitativesDiscretes.py`.
Pour effectuer les analyses descriptives dans le cas continu, la démarche est la même (sauf quantiles) que pour le cas discret.
On utilisera cependant comme données les centres des intervalles.

Paramètres **listeIntervalles** – liste issue de la discrétisation des valeurs.

Retourne liste de flottants.

add.addQuantitativesContinues.**interpolationLineaire**(*p1*, *p2*, *y*)
Calcule l'abscisse par interpolation linéaire
Les points `p1`, `p2` nous permettent de définir une fonction linéaire f(x) = pente * x + ordonnée à l'origine (oo).
On retrouve ensuite l'abscisse du point d'ordonnée `y` se trouvant sur la courbe de la fonction, x = (`y` - oo) / pente.

Retourne abscisse de l'ordonnée `y` par rapport à la droite (`p1`, `p2`)

Type retourné float

add.addQuantitativesContinues.**quantileContinu**(*ordre*, *listeFrequencesCumulees*, *intervalles*)

Calcule les quantiles d'ordre `ordre` pour une analyse de données conitnues.

Le quantile discret nous permet de retrouver le centre de l'intervalle qui contient le vrai quantile. Ensuite, à partir de l'intervalle et de l'ordre, on en déduit une valeur plus précise par interpolation linéaire.

La fonction linéaire est définie à l'aide des bornes de l'intervalle, on a besoin de deux points :

L'ordonnée de la borne supérieure est la fréquence cumulée du centre de l'intervalle (1 si borne = max)

L'ordonnée de la borne inférieur est la fréquence cumulée du centre de l'intervalle précédent (0 si borne = min)

Retourne le quantile d'ordre `ordre`

add.addQuantitativesContinues.**anomaliesTukeyContinu**(*listeEffectifs*, *etendueIntervalles*)

Liste les valeurs aberrantes de la list, cas continu.

Une valeur est dite aberrante selon la règle de Tukey si elle n'appartient pas à un intervalle I définit tel que : I = [Q1 - k * IQ ; Q3 + k * IQ] , k constante réelle Q1 et Q3 les quartiles, IQ l'écart inter-quartiles.

La constante k est choisie arbitrairement égale à 1,5. La valeur 1.5 est selon Tukey une valeur pragmatique, qui a une raison probabiliste. Si une variable suit une distribution normale, alors la zone délimitée par la boîte et les moustaches devrait contenir 99,3 % des observations.

Type retourné list

Retourne Collection contenant les données anormales pour la distribution des valeurs.

add.addQuantitativesContinues.**infoDistributionCumulativeContinue**(*listeEffectifsCumules*, *intervalles*)

Création d'un dictionnaire pour les informations sur la distribution cumulative des données.

Couples (clé, valeur) :

— « x » : liste des abscisses, les données
— « value » : liste des ordonnées, les effectifs cumulés respectifs

Paramètres **listeEffectifCumules** – liste de couples (centre de l'intervalle, effectif cumulé).

Type retourné dict

add.addQuantitativesContinues.**infoBoiteTukeyContinu**(*listeEffectifs*, *etendueIntervalles*)

Création d'un dictionnaire pour la boîte à moustaches de Tukey, cas continu.

Cette fonction est compatible pour les données discrètes et continues.

Couples (clé, valeur) :

— « q1 » : premier quartile
— « q3 » : troisième quartile
— « median » : la médiane de la série de données
— « left » : extrémité de la moustache gauche, Q1 - k * (Q3 - Q1)
— « droite » : extrémité de la moustache droite, Q3 + k * (Q3 - Q1)
— « outliers » : une liste des anomalies statistiques

Paramètres

— **listeEffectifs** – liste de couples (valeur, effectif).
— **etendueIntervalles** – partition de l'étendue de la série des données, issue de la discrétisation.

Type retourné dict

add.addQuantitativesContinues.**infoStats**(*listeEffectifs*, *etendueIntervalles*)

Création d'un dictionnaire pour le résumé des informations statistiques.

Couples (clé, valeur) :

— « Min » : la valeur minimale de la série
— « Max » : la valeur maximale de la série
— « Range » : la différence entre le `Max` et le `Min`
— « IQR » : l'écart inter-quartiles, valeur de l'étendue regroupant 75% des données
— « Mean » : la moyenne arithmétiqued
— « Median » : la médiane de la série de données
— « StdDev » : l'écart-type de la série de données
— « Outliers » : une liste des anomalies statistiques

Paramètres
— **`listeEffectifCumules`** – liste de couples (centre de l'intervalle, effectif cumulé).
— **`etendueIntervalles`** – partition de l'étendue de la série des données, issue de la discrétisation.

Type retourné dict

3.2.3 addQuantitativesDiscretes.py

`add.addQuantitativesDiscretes.`**`moyenne`**(*listeEffectifs*)
Calcule la moyenne arithmétique.

Paramètres **`listeEffectifs`** – liste de couples (valeur, occurences)

Retourne moyenne arithmétique des valeurs de la liste

`add.addQuantitativesDiscretes.`**`quantileDiscret`**(*ordre*, *listeFrequencesCumulees*)
Calcule le quantile d'ordre `ordre`.
Quantile non défini si l'ordre n'est pas compris entre 0 exclus et 1 exclus

Paramètres
— **`ordre`** – Nombre flottant compris entre 0 et 1.
— **`listeFrequencesCumulees`** – liste de couples (valeur, frequence cumulée) triée selon les valeurs

Retourne La première valeur telle que la fréquence cumulée correspondante soit supérieure ou égale à l'ordre.

Type retourné float

Note : La médiane est le quantile d'ordre 0.5.
Les quartiles sont les quantiles d'ordre 0.25 et 0.75.

`add.addQuantitativesDiscretes.`**`variance`**(*listeEffectifs*)
Calcule la variance.
La variance représente statistiquement l'écart quadratique à la moyenne.
Plus les valeurs du eu de données sont proches de la moyenn, plus la variance est faible.

Paramètres **`listeEffectifs`** – liste de couples (valeur, occurences)

Type retourné float

`add.addQuantitativesDiscretes.`**`ecartType`**(*variance*)
Calcule l'écart-type.
L'écart-type est la racine carrée de la variance.
Cette statistique permet de justifier la pertinence ou non de la valeur moyenne.

Type retourné float

add.addQuantitativesDiscretes.**anomaliesTukeyDiscret**(*listeEffectifs*)

Liste les valeurs aberrantes de la list, cas discret.

Une valeur est dite aberrante selon la règle de Tukey si elle n'appartient pas à un intervalle I définit tel que : I = [Q1 - k * IQ ; Q3 + k * IQ] , k constante réelle Q1 et Q3 les quartiles, IQ l'écart inter-quartiles.

La constante k est choisie arbitrairement égale à 1,5. La valeur 1.5 est selon Tukey une valeur pragmatique, qui a une raison probabiliste. Si une variable suit une distribution normale, alors la zone délimitée par la boîte et les moustaches devrait contenir 99,3 % des observations.

Type retourné list

Retourne Collection contenant les données anormales pour la distribution des valeurs.

add.addQuantitativesDiscretes.**symetrie**(*listeEffectifs*)

Calcule le coefficient de symétrie de Fisher.

Si le coefficient est proche 0, la distribution est approximativement symétrique.

Si le coefficient est positif, la distribution est étalée sur la droite.

Si le coefficient est négatif, la distribution est étalée sur la gauche.

En théorie si l'écart-type est égal à 0, la symétrie n'est pas définie.

Cependant un écart-type égal à 0 s'interprète :

Si toutes les valeurs de la distribution sont égales à la moyenne, notre écart-type va être nul.

On peut alors considérer la distribution parfaitement symétrique, toutes les données sont regroupées en un point, la moyenne.

Type retourné float

add.addQuantitativesDiscretes.**aplatissement**(*listeEffectifs*)

Calcule le coefficient d'aplatissement de Fisher.

Si le coefficient est égal à 3, la distribution suit une loi normale centrée réduite.

Si le coefficient est inférieur à 3, la distribution est aplatie.

Si le coefficient est supérieur à 3, les valeurs de la distribution est concentrée autour de la moyenne.

Non défini si l'écart-type est nul

Type retourné float

add.addQuantitativesDiscretes.**infoDistribution**(*listeEffectifs*)

Création d'un dictionnaire pour la distribution des données.

Cette fonction est compatible pour les données discrètes et continues.

Couples (clé, valeur) :

— « x » : liste des abscisses, les données
— « value » : liste des ordonnées, les effectifs cumulés respectifs

Paramètres **listeEffectifs** – liste de couples (valeur, effectif).

Type retourné dict

add.addQuantitativesDiscretes.**infoDistributionCumulativeDiscrete**(*listeEffectifsCumules*)

Création d'un dictionnaire pour la distribution cumulative des données discrètes.

Couples (clé, valeur) :

— « x » : liste des abscisses, les données
— « value » : liste des ordonnées, les effectifs cumulés respectifs

Paramètres **listeEffectifCumules** – liste de couples (valeur, effectif cumulé).

Type retourné dict

add.addQuantitativesDiscretes.**infoBoiteTukeyDiscret**(*listeEffectifs*)

Création d'un dictionnaire pour la boîte à moustaches de Tukey, cas discret.

Cette fonction est compatible pour les données discrètes et continues.

Couples (clé, valeur) :

— « q1 » : premier quartile
— « q3 » : troisième quartile
— « median » : la médiane de la série de données
— « left » : extrémité de la moustache gauche, Q1 - k * (Q3 - Q1)
— « droite » : extrémité de la moustache droite, Q3 + k * (Q3 - Q1)
— « outliers » : une liste des anomalies statistiques

Paramètres **listeEffectifs** – liste de couples (valeur, effectif).

Type retourné dict

3.2.4 intervalle.py

Description

Ce module est utilisé pour une manipulation réduite des intervalles dans **R**.

```
>>> from add import intervalle
>>> i = Intervalle(0, 1, true ,false)
>>> i.contient(0.8)
True
```

L'intervalle i de l'exemple sert à représenter `[0, 1[` en notation mathématiques.

Son utilisation dans l'application permet d'avoir des objets simples à manipuler lors de la discrétisation de données continues.

La classe intervalle.py

class `add.intervalle.`**Intervalle**(*borneInf*, *borneSup*, *infInclus*, *supInclus*)

Ensemble d'objets pour représenter les intervalles de **R**.

Attributs :

— borneInf `float` : borne inférieure de l'intervalle
— borneSup `float` : borne supérieure de l'intervalle
— infInclus `boolean` : indique si la borne inférieure est comprise ou non dans l'intervalle
— supInclus `boolean` : indique si la borne supérieure est comprise ou non dans l'intervalle
— centre `float` : centre de l'intervalle

contient(*nombre*)

Retourne vrai si l'argument appartient à l'intervalle, faux sinon

Les fonctions

`add.intervalle.`**rechercheIntervalle**(*intervalles*, *nombre*)

Retrouve l'intervalle contenant un nombre en parmaètre

Les intervalles de la liste sont distincts. Les intervalles de la liste forment une partition de l'étendue qu'ils représentent et sont triés.

Ainsi, le nombre (on suppose qu'il appartient à l'étendue) appartient à un unique intervalle de la liste.

Retourne l'intervalle auquel `nombre` appartient, `False` sinon

3.3 Chargement des données

3.3.1 verificationFormatFichier.py

chargement_des_donnees.verificationFormatFichier.**verifExistence**(*chemin*)
Vérifie l'existance du fichier pour l'ouverture.

Paramètres **chemin** (*str*) – chemin du fichier

Retourne entier 0 ou une erreur `not an existing file`

chargement_des_donnees.verificationFormatFichier.**verifExtension**(*chemin*)
Vérification l'extension du fichier.

Paramètres **chemin** (*str*) – chemin du fichier

Retourne entier 0 ou une erreur `file extension not .csv`

chargement_des_donnees.verificationFormatFichier.**verifLecture**(*fichierCSV*)
Vérifie l'accès au contenu du fichier ainsi que sa nature.

Paramètres **fichierCSV** (*TextIoWrapper*) – le fichier CSV ouvert

Retourne entier 0, une erreur `not a raw text file` ou `file content not readable`

chargement_des_donnees.verificationFormatFichier.**verifCSV**(*fichierCSV*)
Vérifie si le fichier CSV présumé est strucutré.
Examine si le fichier contient bien un caractère délimitant les valeurs entre elles.

Paramètres **fichierCSV** (*TextIoWrapper*) – le fichier CSV ouvert

Retourne entier 0 ou une erreur `not a structured CSV file`

chargement_des_donnees.verificationFormatFichier.**ouvrir**(*chemin*)
Fonctionnalité principale d'ouverture du fichier CSV et de vérification

Paramètres **chemin** (*str*) – chemin du fichier

Retourne le fichier CSV ouvert ou la description de l'erreur rencontrée lors de l'ouverture

3.3.2 analyseContenuFichier.py

chargement_des_donnees.analyseContenuFichier.**lecture**(*fichierCSV*, *toClose*)
Lit le contenu du fichier CSV ligne par ligne.
Lors du parcours du fichier `fichierCSV`, la fonction se charge de remplir une structure contenant les données du fichier.

Paramètres **fichierCSV** (*TextIoWrapper*) – fichier CSV ouvert et vérifié

Retourne liste dont chaque élément est une sous-liste contenant les données d'une ligne du fichier.

chargement_des_donnees.analyseContenuFichier.**typeDeDonnee**(*chaine*)
Détecte le type des données depuis une chaine de caractères.

Paramètres **fichierCSV** (*TextIoWrapper*) – fichier CSV

Retourne liste dont chaque élément est une sous-liste contenant les données d'une ligne du fichier

chargement_des_donnees.analyseContenuFichier.**removeDateSuffix**(*chaineDate*)
Supprime les suffixes des jours du mois dans une chaine de caractères représentant une date

Paramètres **chaineDate** – chaine de caractères représentant une date

Retourne chaine de caractères de la date sans suffixes

chargement_des_donnees.analyseContenuFichier.**descriptionColonnes**(*lignesCSV*)
Renseignement des descriptions du nom, du type et des erreurs des colonnes du fichier CSV.
On va enregistrer dans un dictionnaire `descCSV` des informations concernant :

— Le nom des colonnes.
— Le type attendu pour chacune des colonnes.
— Un mention d'erreur ou `correct` pour chaque donnée du fichier par rapport au type attendu.

Paramètres **`lignesCSV`** (*`list`*) – lignes du fichier CSV

Retourne dictionnaire de 3 sous-listes ayant pour clés : « nom », « type » et « erreurs »

`chargement_des_donnees.analyseContenuFichier.`**`analyseFichier`**(*fichierCSV*)

Fonctionnalité principale d'analyse du contenu du fichier CSV ouvert.

Cette fonctionnalité réutilise deux fonctions, `lecture` et `descriptionColonnes`.

Objectifs :

— Lire les données présentes dans le fichier `.csv`.
— Fournir un description de ce fichier : nom des colonnes, type des données, erreurs relevées.

Paramètres **`fichierCSV`** (*`TextIoWrapper`*) – le fichier CSV ouvert et vérifié

Retourne un couple (données du fichier, description de ces données)

3.4 Exemples

3.4.1 demoAPI.py

Description

Principe : Pour chaque arc (i.e. chaque couple (enfant, parent) distinct), et pour chaque colonne contenue dans le fichier, nous écrivons en sortie, un compte-rendu statistique des mesures relevées.

Appel en ligne de commande :

```
python demoAPI.py monFichier.csv maSortie.csv
```

Le fichier `monFichier.csv` est au même format que ceux qui peuvent être analyser par l'interface web.

Quant au fichier `maSortie.csv` son format est similaire, voici ses colonnes :

— stats : information statistique calculée pour la ligne
— enfant : numéro du noeud
— parent : numéro du noeud
— mesures : information de la colonne *stats* calculée pour toutes les données la colonne mesure de `monFichier.csv` à l'arc correspondant.

Les colonnes mesures correspondent aux données mesurées sur le réseau. Un exemple sera plus parlant.

L'image ci-dessous sert d'example pour `monFichier.csv`, on reconnaît les timestamps, et pour l'arc (275, 22), les mesures des quatre colonnes, sont faites à desinstants différents.

timestam	enfant	parent	Q_CPT_NI	PUI_CPT_	TP_ENT_C	TP_SORT_CPT_NRJ (°)
January 1	275	22	26644.23	176.253	59.136	53.363
January 1	275	22	26622	178.489	59.401	53.534
January 1	275	22	26750	221.367	64.36	57.126
January 2	275	22	27650	307.794	85.471	75.558
January 2	275	22	27886	335.894	89.025	78.405
January 2	275	22	27562	325.494	89.381	78.961
January 2	275	22	27356	263.461	85.466	77.468
January 2	275	22	26992	344.05	90.449	79.495
January 2	275	22	26810	303.883	87.839	77.924
January 2	275	22	27012	318.756	87.353	76.951
January 2	275	22	26588	326.572	90.962	80.133
January 3	275	22	27916	242.544	85.121	77.481
January 3	275	22	27560	300.556	89.958	79.962
January 3	275	22	26902	287.544	88.652	79.178
January 3	275	22	27078	313.622	90.249	79.834
January 3	275	22	27030	320.794	90.148	79.9

Enfin, l'image ci-dessous sert d'example pour `maSortie.csv`. On peut toujours voir l'arc (275, 22). Cependant pour les quatre colonnes, il ne s'agit de mesures, mais de calculs dont les descriptions figurent dans la colonne *stats*.

stats	enfant	parent	Q_CPT_NI	PUI_CPT_	TP_ENT_C	TP_SORT_CPT_NRJ (°)
min	274	75	59.403	164.8	8292.0	42.188
max	274	75	90.471	301.844	9288.0	61.484
range	274	75	31.0680(	137.043(	996.0	19.296
mean	274	75	84.44015	273.629(	8625.952	57.28428235294118
median	274	75	86.88623	285.6478	8320.457	59.20356363636364
IQR	274	75	3.584769	16.1961(	28.45714	2.280436363636362
stdDev	274	75	8.007832	31.34635	257.7882	4.963206136008468
min	275	75	79.401	158.083	22122.0	72.072
max	275	75	93.16	359.078	27300.0	82.672
range	275	75	13.759	200.994(	5178.0	10.599999999999994
mean	275	75	89.64025	285.2907	25940.9(	79.41452491694353
median	275	75	83.33214	212.6387	25338.14	78.34831578947367
IQR	275	75	8.026083	47.85595	385.9378	0.37857142857143344
stdDev	275	75	2.220278	46.91441	892.5925	1.996778345843842
min	275	22	58.639	6.844	11658.0	50.919
max	275	22	92.559	381.967	28374.0	90.45
range	275	22	33.9199(	375.123	16716.0	39.531000000000006
mean	275	22	87.86174	282.8229	26550.8(	78.22574877450981
median	275	22	87.4790(	223.3672	26903.61	55.860375
IQR	275	22	4.085294	36.96557	1657.62(	15.647687500000004
stdDev	275	22	4.730794	62.08783	1894.419	4.691125232295683

Fonctions réutilisées

Les résultats satisfaisants de ce script nous ont inscités à réutiliser ses fonctionnalités au sein même de notre application :

- `safe_open_w` : garantit la création des dossiers après eventuels clean up.
- `ecrireResultats` : pour le téléchargement des résultats d'analyse descriptive.

`demoAPI.`**`safe_open_w`**(*path*)

Ouverture du fichier en écriture et création des dossiers du chemin s'ils n'existent pas.

demoAPI.**ecrireResultats**(*entree*, *sortie*)

Analyses descriptives sur le fichier au chemin `entree` écriture des résultats dans `sortie`

Cette fonction reprend les fonctionnalités de l'API chargement des données pour ouvrir un fichier `.csv` et obtenir son contenu ainsi qu'une description des données.

La description des données est utilisée pour réaliser les analyses descriptives uniquement sur les données valides.

Index des modules Python