

\*

0cm7ex plus 1ex minus .2ex1.5ex plus .2ex \*

0cm4.5ex plus 1ex minus .2ex1.5ex plus .2ex \*

\*

\*

# Spécifications

Sonny Klotz - Jean-Didier Pailleux - Malek Zemni

*Interface de chargement, de contrôle  
et d'analyse statistique des données  
pour la constitution d'un graphe de flux*

17 avril 2017



Module *Projet*

## Table des matières

<b>1</b>	<b>Package Chargement des données</b>	<b>1</b>
1.1	Module Vérification format fichier . . . . .	1
1.2	Module Analyse contenu fichier . . . . .	3
<b>2</b>	<b>Package Analyse descriptive des données</b>	<b>4</b>
2.1	Module Analyse de données qualitatives . . . . .	5
2.2	Module Analyse de données quantitatives discrètes . . . . .	7
2.3	Module Analyse de données quantitatives continues . . . . .	10
<b>3</b>	<b>Package Interface web</b>	<b>12</b>
3.1	Module Gestion des flux . . . . .	12
3.2	Module Fenêtre choix fichier . . . . .	15
3.3	Module Fenêtre rôle et choix colonne . . . . .	17
3.4	Module Fenêtre résultats ADD . . . . .	17
<b>4</b>	<b>Glossaire des types</b>	<b>19</b>

# Introduction

Ce document va décrire l'ensemble des exigences fonctionnelles que doit satisfaire notre produit final, c'est-à-dire les différentes fonctionnalités que notre application va fournir. Cette description va prendre en compte les caractéristiques des outils de développement choisis.

Notre outil, Python, est un langage de programmation hybride. On utilisera d'une part la programmation fonctionnelle pour les calculs, et d'autre part la programmation objet pour le développement des interfaces graphiques. Dans Python, le type de données n'est connu qu'à l'exécution (typage dynamique), par conséquent, ces types ne seront pas indiqués dans les signatures des fonctions et les structures des classes. Ils seront précisés dans des paragraphes explicatifs.

Pour les parties qui s'appuient sur une interaction avec l'utilisateur, notre démarche de description des fonctionnalités va essentiellement prendre en compte l'*expérience utilisateur*<sup>1</sup>. Cette description sera donc axée sur la qualification du résultat et du ressenti de l'utilisateur lors de la manipulation de l'interface fournie (une illustration à l'aide de croquis), plutôt que sur les points techniques de l'application (fonctions et classes).

Les fonctionnalités de notre application seront présentées selon les modules de l'organigramme établi dans le cahier des charges. Ces modules eux-mêmes seront regroupés en packages. Ce document va donc décrire, pour chaque package de l'organigramme, les fonctionnalités de ses modules : d'abord ceux du package de chargement des données, puis ceux du package d'analyse descriptive des données et ensuite ceux du package de l'interface web. Les types Python utilisés seront décrits dans la dernière partie, le glossaire des types.

## 1 Package Chargement des données

Ce package est composé de 2 modules qui ont pour fonction de traiter le fichier de données fourni : une vérification de son format et une analyse de son contenu. On pourra aussi parler de API puisque ce package peut être éventuellement livré en sortie.

### 1.1 Module Vérification format fichier

Ce module va vérifier le format du fichier de données fourni en entrée en 3 points. Il aura donc 3 fonctionnalités de vérification et une fonctionnalité globale d'ouverture du fichier.

1. Fonctionnalité de vérification de l'ouverture du fichier :

---

1. <http://uxdesign.com/ux-defined>

```

1         verifOuverture(fichierCSV)
2

```

Paramètres :

[style=unboxed,leftmargin=0.2cm]fichierCSV : TextIoWrapper - représente le fichier CSV fourni.

Retour : variable de type entier signalant un succès (0) ou une erreur (1).

Description : cette fonction prend en entrée le fichier CSV ouvert. Elle vérifie que le paramètre fichierCSV contient bien des informations représentant un fichier quelconque, et renvoie l'entier 0 si c'est le cas, l'entier 1 sinon.

2. Fonctionnalité de vérification de l'extension du fichier ouvert :

```

1         verifExtension(fichierCSV)
2

```

Paramètres :

[style=unboxed,leftmargin=0.2cm]fichierCSV : TextIoWrapper - représente le fichier CSV fourni.

Retour : variable de type entier signalant un succès (0) ou une erreur (2).

Description : cette fonction prend en entrée le fichier CSV ouvert. Elle vérifie que l'information décrivant l'extension du fichier dans le paramètre fichierCSV correspond bien au format CSV, et renvoie l'entier 0 si c'est le cas, l'entier 2 sinon.

3. Fonctionnalité de vérification de l'accessibilité en lecture du fichier :

```

1         verifLecture(fichierCSV)
2

```

Paramètres :

[style=unboxed,leftmargin=0.2cm]fichierCSV : TextIoWrapper - représente le fichier CSV fourni.

Retour : variable de type entier signalant un succès (0) ou une erreur (3).

Description : cette fonction prend en entrée le fichier CSV ouvert. Elle vérifie que le paramètre fichierCSV représentant le fichier possède bien la propriété d'accès en lecture. Un test de lecture sera aussi effectué sur le fichier. La fonction renvoie l'entier 0 si l'accès en lecture est permis, l'entier 3 sinon.

4. Fonctionnalité globale d'ouverture du fichier CSV :

```

1         ouvrir(chemin)
2

```

Paramètres :

[style=unboxed,leftmargin=0.2cm]chemin : **str** - chaîne de caractères représentant le chemin relatif ou absolu du fichier CSV fourni.

Retour : variable de type entier signalant un succès (0) ou une erreur (1, 2, ou 3).

Description : cette fonction prend en entrée le chemin du fichier CSV fourni. Elle ouvre ce fichier en lecture à l'aide de la fonction **open**(chemin, "r") et le stocke dans une variable de type `TextIoWrapper`. Elle applique ensuite dans l'ordre les trois fonctions de vérifications du module et renvoie le code de la fonction qui fait échouer l'ouverture (l'entier 1, 2 ou 3), ou bien un code signalant le succès de l'ouverture (l'entier 0).

## 1.2 Module Analyse contenu fichier

Ce module va analyser le contenu du fichier fourni en lisant une à une les données de ce fichier et en repérant les données erronées. Il aura donc 2 fonctionnalités :

### 1. Fonctionnalité de lecture du contenu du fichier CSV :

```
1 lecture(fichierCSV)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]fichierCSV : `TextIoWrapper` - représente le fichier CSV fourni.

Retour : une variable de type **list** : une liste dont chaque élément est un dictionnaire contenant les données d'une ligne du fichier CSV.

Description : cette fonction prend en entrée le fichier CSV ouvert. Elle crée une liste pour y sauvegarder le contenu de ce fichier, ligne par ligne. Elle lit les caractères du fichier en entier à l'aide de la fonction `read()`. Ensuite, pour chaque ligne du fichier, elle applique la fonction `split(',')` pour séparer les caractères de cette ligne selon la virgule et les stocker dans un dictionnaire qui sera finalement stocké dans la liste englobante. L'utilisation d'un dictionnaire permet d'indexer les données de la ligne par le nom de la colonne :

```
1 [
2     {"nomColonne1" : donnée1Ligne1, "nomColonne2"
3     donnée2Ligne1},
4     {"nomColonne1" : donnée1Ligne2, "nomColonne2"
5     donnée2Ligne2}
6 ]
```

Ces noms de colonnes proviennent du fichier CSV.

## 2. Fonctionnalité de description du type et des erreurs des colonnes du fichier CSV :

```
1 descriptionColonnes(lignesCSV)
2
```

### Paramètres :

[style=unboxed,leftmargin=0.2cm]lignesCSV : **list** - liste représentant les données du fichier CSV ligne par ligne.

Retour : une variable de type **dict** : un dictionnaire de 2 dictionnaires indexés par "**types**" et "**erreurs**".

Description : cette fonction prend en entrée le fichier CSV préalablement lu et stocké dans la liste lignesCSV. Elle crée un dictionnaire de 2 éléments de type dictionnaire également, pour y renseigner le type et les erreurs de chaque colonne de ce fichier.

Elle remplit d'abord le champ types du dictionnaire par les types attendus. Ensuite, elle lit la liste lignesCSV colonne par colonne. Pour chaque donnée d'une colonne, elle compare son type au type attendu et, en cas d'erreur, renseigne le numéro de cette ligne dans une liste dans le champ erreurs du dictionnaire.

```
1 {
2     "types" : {"nomColonne1" : typeColonne1,
3     "nomColonne2" typeColonne2},
4     "erreurs" :
5     {
6         "nomColonne1" : [liste des numéros de
7         lignes des données erronées de la colonne 1],
8         "nomColonne2" : [liste des numéros de
9         lignes des données erronées de la colonne 2]
10    }
11 }
```

## 2 Package Analyse descriptive des données

Ce package est composé de trois modules d'ADD<sup>2</sup>. Ces informations pourront être utilisées pour le filtrage des valeurs dans l'échantillon de la fenêtre rôle et choix colonne, mais aussi servir pour les compte-rendu statistiques des colonnes. Ce package va être livré au client pour une intégration externe. On pourra donc parler de API.

---

2. ADD : analyse descriptive de données

## 2.1 Module Analyse de données qualitatives

Ce module va effectuer les calculs d'effectifs, d'effectifs cumulés, de fréquences et de fréquences cumulés. Il s'occupera également de fournir les informations nécessaires pour la construction d'un diagramme de secteur et d'un histogramme. Il aura donc 6 fonctionnalités :

### 1. Fonctionnalité de calcul de l'effectif :

```
1 calculEffectif(donneeQualitative)
2
```

#### Paramètres :

[style=unboxed,leftmargin=0.2cm]donneeQualitative : **list** - structure contenant les données de type qualitatif de la colonne.

Retour : variable de type **list** : représente la sortie du calcul des effectifs de chaque valeur de la colonne.

Description : cette fonction prendra en paramètre une liste qui contiendra les données de type qualitatif pour le calcul. Elle calculera les effectifs pour chaque valeur et effectuera un tris pour ordonner ces valeurs.

### 2. Fonctionnalité de calcul de l'effectif cumulé :

```
1 calculEffectifCumule(listeEffectif)
2
```

#### Paramètres :

[style=unboxed,leftmargin=0.2cm]listeEffectif : **list** - correspond au résultat de la fonction calculEffectif.

Retour : listeEffectif : **list** : copie de listeEffectif mais dont les effectifs seront remplacés par les effectifs cumulés.

Description : cette fonction prend en entrée les résultats de la fonction calculEffectif préalablement stockés dans une liste. Elle va copier dans une nouvelle liste celle rentrée en paramètre, et remplacera les effectifs par les effectifs cumulés. La fonction retournera cette nouvelle liste.

### 3. Fonctionnalité pour calculer les fréquences d'apparition :

```
1 calculFrequence(donneeQualitative)
2
```

#### Paramètres :

[style=unboxed,leftmargin=0.2cm]donneeQualitative : **list** - structure contenant les données de type qualitatif de la colonne.



Retour : variable de type **list** - correspondant à la sortie du calcul des fréquences de chaque valeur de la colonne.

Description : cette fonction prendra en paramètre une liste qui contiendra les données de type qualitatif pour le calcul. Elle calculera les fréquences pour chaque valeur et effectuera un tris pour ordonner ces valeurs.

#### 4. Fonctionnalité pour calculer les fréquences cumulées :

```
1 calculFrequenceCumule(listeFrequence)
2
```

##### Paramètres :

[style=unboxed,leftmargin=0.2cm]listeFrequence : **list** - correspond au résultat de la fonction calculFrequence.

Retour : variable de type **list** - copie de listeFréquence mais dont les fréquences seront remplacé par les fréquences cumulés.

Description : cette fonction prend en entrée les résultats de la fonction calculFrequence préalablement stocké dans une liste. Elle va copier dans une nouvelle liste celle rentrée en paramètre, et remplacera les fréquences par les fréquences cumulés. La fonction retournera cette nouvelle liste.

#### 5. Fonctionnalité pour les données du diagramme en secteur :

```
1 infoSecteur(frequence)
2
```

##### Paramètres :

[style=unboxed,leftmargin=0.2cm]frequence : **list** - correspondant à la sortie du calcul des fréquences.

Retour : fichier au format `.json` - contenant les informations nécessaire pour la construction d'un diagramme de secteur.

Description : cette fonction prend en entrée le résultat du calcul des fréquences préalablement stocké dans une liste `frequence`. Elle va créer un fichier `.json` pour y stocker les données pour construire le diagramme de secteur. Pour chaque couple (fréquence-valeur) elle va associer un angle compris entre  $0^\circ$  et  $360^\circ$  tel que la somme de tout les angles soit égale à  $360^\circ$ .

#### 6. Fonctionnalité pour les données de l'histogramme :

```
1 infoHistogramme(effectif)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]effectif : **list** - correspondant à la sortie du calcul des effectifs.

Retour : fichier au format `.json` - contenant les informations nécessaire pour la construction d'un histogramme.

Description : cette fonction prend en entrée le résultat du calcul des effectifs préalablement stocké dans une liste `effectif`. Elle va créer un fichier `.json` pour y stocker les données pour construire l'histogramme.

## 2.2 Module Analyse de données quantitatives discrètes

### 1. Calcul de la moyenne

```
1 moyenne(listeEffectifs)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]listeEffectifs : **list** - structure contenant les différents effectifs des données de l'ensemble analysé.

Type retour : **float** : moyenne arithmétique de l'ensemble de données

Description : Calcul de la moyenne arithmétique en fonction des données et de leur occurrences dans la liste.

### 2. Calcul des quantiles (cas discret)

```
1 quantileDiscret(ordre, listeFrequencesCumulees)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]ordre : **float** - Nombre flottant compris entre 0 et 1 listeFrequencesCumulees : **list** - structure contenant les différentes fréquences cumulées des données de la colonne analysées

Type retour : **float**

Description : Valeur de la liste séparant les données en deux ensembles, le premier contenant la fraction `ordre` des données, le deuxième contenant le reste. La médiane est le quantile d'ordre  $1/2$ .

### 3. Calcul de la variance

```
1 variance(listeEffectifs)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]listeEffectif : **list** - structure contenant les différents effectifs des données de l'ensemble analysé.

Type retour : **float**

Description : Variance de l'ensemble de données analysé

#### 4. Calcul de l'écart-type

```
1 ecartType(variance)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]variance : **float**

Type retour : **float**

Description : Racine carrée de la variance.

#### 5. Détection des anomalies

```
1 anomaliesTukey(listeEffectifs)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]listeEffectifs : **list** - structure contenant les différents effectifs des données de l'ensemble analysé.

Type retour : **list** : collection contenant les données considérées anormales pour la distribution des valeurs

Description : Selon la méthode de Tukey, toute donnée n'étant pas contenue dans l'intervalle  $[Q1 - 1.5 * (Q3 - Q1) ; Q3 + 1.5 * (Q3 - Q1)]$ , avec  $Q1$ ,  $Q3$  les quartiles, est considérée comme une anomalie statistique.<sup>3</sup>

#### 6. Calcul du coefficient de symétrie

```
1 symetrie(listeEffectifs)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]listeEffectifs : **list** - structure contenant les différents effectifs des données de l'ensemble analysé.

---

3. <https://hal.archives-ouvertes.fr/halshs-00287751/document>

Type retour : **float** - Nombre flottant compris entre -1 et 1.

Description : Coefficient d'asymétrie de Pearson calculé avec la moyenne, la médiane et l'écart-type : s'il est nul la distribution est symétrique, s'il est positif, la série est étalée à droite, et s'il est négatif, la série est étalée à gauche.

## 7. Calcul du coefficient d'aplatissement

```
1      aplatissement(listeEffectifs)
2
```

`listeEffectifs` : **list** - structure contenant les différents effectifs des données de l'ensemble analysé. Paramètres :

[style=unboxed,leftmargin=0.2cm]

Type retour : **float**

Description : Coefficient d'aplatissement de Fisher, calculé à partir de la moyenne et de l'écart-type. Si le coefficient est égal à 3, la série suit une loi normale centrée réduite, s'il est inférieur la distribution des valeurs est aplatie, et s'il est supérieur, les valeurs sont concentrées autour de la moyenne.

## 8. Fonctionnalité pour les données de la distribution des valeurs

```
1      infoDistribution(listeEffectifs)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]`listeEffectifs` : **list** - structure contenant les différents effectifs des données de l'ensemble analysé.

Retour : fichier au format `.json` - contenant les couples (valeur, effectif)

Description : Préparation de l'objet contenant les informations nécessaires à l'affichage de la distribution des valeurs.

## 9. Fonctionnalité pour les données de la distribution cumulative des valeurs (cas discret)

```
1      infoDistributionCumulativeDiscrete(listeEffectifsCumules)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]`listeEffectifsCumules` : **list** - structure contenant les différents effectifs cumulés des données de l'ensemble analysé.

Retour : fichier au format `.json` - contenant les couples (valeur, effectif cumulé)

Description : Préparation de l'objet contenant les informations nécessaires à l'affichage de la distribution cumulative des valeurs (diagramme en escaliers)

#### 10. Fonctionnalité pour les données de la boîte à moustaches de Tukey

```
1 infoBoiteTukey(listeEffectifs)
2
```

Paramètres :

`[style=unboxed,leftmargin=0.2cm]listeEffectifs` : **list** - structure contenant les différents effectifs des données de l'ensemble analysé.

Retour : fichier au format `.csv` - contenant les données du paramètre.

Description : La représentation graphique met en valeur les quartiles et la médiane, ainsi que les anomalies de Tukey.

#### 11. Fonctionnalité pour les données de la série temporelle

```
1 infoSerieTemporelle(listeSerieTemporelle)
2
```

Paramètres :

`[style=unboxed,leftmargin=0.2cm]listeSerieTemporelle` : **list** - liste des couples (Timestamp, valeur) des données analysées.

Retour : fichier au format `.json` - contenant les couples (Timestamp, valeur)

Description : La représentation graphique associée consiste à afficher l'évolution des valeurs dans le temps.

## 2.3 Module Analyse de données quantitatives continues

#### 1. Fonctionnalité pour la discrétisation des valeurs

```
1 discretisation(nombreClasses, donneesContinues)
2
```

Paramètres :

`[style=unboxed,leftmargin=0.2cm] !nombreClasses` : `int !` - indique en combien de classes d'intervalle l'étendue des données va être divisée. `donneesContinues` : **list** - liste de flottants représentant les mesures analysées.

Type retour : **list** - collection d'Intervalle (voir glossaire des types)

Description : La fonction se charge de décomposer l'étendue [min ; max] de l'ensemble de données en `nombreClasses` intervalles de même étendue, puis de remplacer les occurrences des données par l'intervalle auquel la donnée appartient.

## 2. Calcul du nombre de classes optimal pour la discrétisation des valeurs

```
1 calculNombreClasses(donneesContinues)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]`donneesContinues` : **list** - liste de flottants représentant les mesures analysées.

Type retour : **int**

Description : Calcul le nombre de classes selon la règle de Sturges.

## 3. Préparation des données pour l'utilisation des éléments de calcul du module ADD quantitatives discrètes

```
1 preparationIntervallesAnalyse(listeIntervalles)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]`listeIntervalles` : **list** - issue de la discrétisation des valeurs.

Type retour : **list** - liste de flottants

Description : Pour effectuer les analyses descriptives dans le cas continu, la démarche est la même (sauf quantiles) que pour le cas discret en utilisant comme données les centres des intervalles.

## 4. Calcul des quantiles (cas continu)

```
1 quanileContinu(ordre, listeFrequencesCumulees)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]`ordre` : **float** - Nombre flottant compris entre 0 et 1 `listeFrequencesCumulees` : **list** - structure contenant les différentes fréquences cumulées des données de la colonne analysées

Type retour : **float**

Description : Les quantiles dans le cas continu sont calculés tout d'abord en identifiant la

classe d'intervalle dans laquelle le quantile se trouve à l'aide des fréquences cumulées, puis par interpolation linéaire.

#### 5. Calcul d'abscisse par interpolation linéaire

```
1 interpolationLineaire(p1, p2, y)
2
```

##### Paramètres :

[style=unboxed,leftmargin=0.2cm]p1 : **float** Premier point de la droite p2 : **float**  
Second point de la droite y : **float** ordonnée du point dont on veut retrouver l'abscisse

Type retour : **float** - abscisse

Description : Les points p1, p2 nous permettent d'identifier une fonction linéaire. On retrouve ensuite l'abscisse du point d'ordonnée y se trouvant sur la courbe de la fonction.

#### 6. Fonctionnalité pour les données de la distribution cumulative des valeurs (cas continu)

```
1 infoDistributionCumulativeContinue(listeFrequencesCumulees)
2
```

##### Paramètres :

[style=unboxed,leftmargin=0.2cm]listeFrequencesCumulees : **list** - structure contenant les différentes fréquences cumulées des données de l'ensemble analysé.

Retour : fichier au format **.json** - contenant les couples (valeur, fréquence cumulée)

Description : Préparation de l'objet contenant les informations nécessaires à l'affichage de la distribution cumulative des valeurs. L'objet de retour est une collection de points que l'on va relier avec des droites.

## 3 Package Interface web

Ce package est composé de 3 modules de fenêtre graphique web qui assurent l'interaction avec l'utilisateur, et d'un module de gestion des flux qui s'occupe du lancement de l'application et de la transition des informations entre ses différents packages.

### 3.1 Module Gestion des flux

Ce module va lancer l'application et sera au centre du programme. Il s'occupera de gérer les branchements pour l'exécution, de communiquer les informations entre les différents modules et

arrêtera le programme en cas d'erreur. Il aura ?? fonctionnalités ainsi que le `main` de l'application :

#### 1. La fonction `main`.

```
1      main()  
2
```

##### Paramètres :

[style=unboxed,leftmargin=0.2cm]Ne prend aucun paramètre.

Retour : Ne retourne rien.

Description : la fonction `main()` ne prend aucun paramètre en entrée et ne retourne rien en sortie. Elle s'occupe de lancer l'application en faisant appel à la première fenêtre choix fichier, et de faire tourner l'application. Elle s'occupe également de gérer les erreurs et donc de l'arrêt du programme.

#### 2. Fonctionnalité pour générer la fenêtre choix fichier.

```
1      choix_fichier()  
2
```

##### Paramètres :

[style=unboxed,leftmargin=0.2cm]Ne prend aucun paramètre.

Retour : `render_template()` : fonction qui renvoie le contenu de la page web mis en paramètre qui va être envoyée au client.

Description : cette fonction correspond à la requête HTTP (c'est à dire le chemin de la page demandée) envoyée par le client et reçue par le serveur qui sera soit un GET soit un POST. Pour cela il faudra indiquer avant de déclarer la fonction la ligne suivante : `@app.route('/choix_fichier/', methods=['GET', 'POST'])`. Elle retournera le contenu de la page web mis en paramètre de la fonction `render_template()` associé au code HTML de la fenêtre choix fichier. Des variables en tant qu'arguments optionnels peuvent-être utilisés dans le cas où l'on souhaiterait afficher des valeurs.

#### 3. Fonctionnalité pour générer la fenêtre rôle et choix colonne.

```
1      role_choix_colonne()  
2
```

##### Paramètres :

[style=unboxed,leftmargin=0.2cm]Ne prend aucun paramètre.



Retour : `render_template()` : fonction qui renvoie le contenu de la page web mis en paramètre qui va être envoyée au client.

Description : cette fonction correspond à la requête HTTP (c'est à dire le chemin de la page demandée) envoyée par le client et reçue par le serveur qui sera soit un GET soit un POST. Pour cela il faudra indiquer avant de déclarer la fonction la ligne suivante : `@app.route('/role_choix_colonne/', methods=['GET', 'POST'])`. Elle retournera le contenu de la page web mis en paramètre de la fonction `render_template()` associé au code HTML de la fenêtre rôle et choix colonne. Des variables en tant qu'arguments optionnels peuvent-être utilisé dans le cas où l'on souhaiterait afficher des valeurs.

#### 4. Fonctionnalité pour générer la fenêtre résultats ADD.

```
1 resultat_ADD()
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]Ne prend aucun paramètre.

Retour : `render_template()` : fonction qui renvoie le contenu de la page web mis en paramètre qui va être envoyée au client.

Description : cette fonction correspond à la requête HTTP (c'est à dire le chemin de la page demandée) envoyée par le client et reçue par le serveur qui sera soit un GET soit un POST. Pour cela il faudra indiquer avant de déclarer la fonction la ligne suivante : `@app.route('/resultat_ADD/', methods=['GET', 'POST'])`. Elle retournera le contenu de la page web mis en paramètre de la fonction `render_template()` associé au code HTML de la fenêtre résultat ADD. Des variables en tant qu'arguments optionnels peuvent-être utilisé dans le cas où l'on souhaiterait afficher des valeurs.

#### 5. Fonctionnalité pour la validation du choix colonne. En attente de Malek

```
1 validChoixColonne(signal)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]signal :

Retour :

Description :

#### 6. Fonctionnalité pour la demande d'exportation. En attente de Sonny

```
1 demandeExport(signal)
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]signal :

Retour :

Description :

### 3.2 Module Fenêtre choix fichier

Ce module est une fenêtre graphique web qui va permettre à l'utilisateur de charger un fichier CSV. Il aura 3 fonctionnalités :

1. Fonctionnalité pour ouvrir le fichier avec le Système de Gestion de fichier :

```
1 FileWithSGF()  
2
```

Paramètres :

[style=unboxed,leftmargin=0.2cm]Ne demande aucun paramètre en entrée.

Retour : variable de type **str** - chaîne de caractères représentant le chemin relatif ou absolu du fichier CSV fourni.

Description : cette fonction ne prend pas de paramètre en entrée. Elle permet de récupérer l'emplacement du fichier CSV avec la technique du Drag&Drop. Elle renvoie donc une chaîne de caractères correspondant à ce chemin.

2. Fonctionnalité pour ouvrir le fichier avec la technique du Drag&Drop.

```
1 FileWithDragDrop()  
2
```

Paramètres :

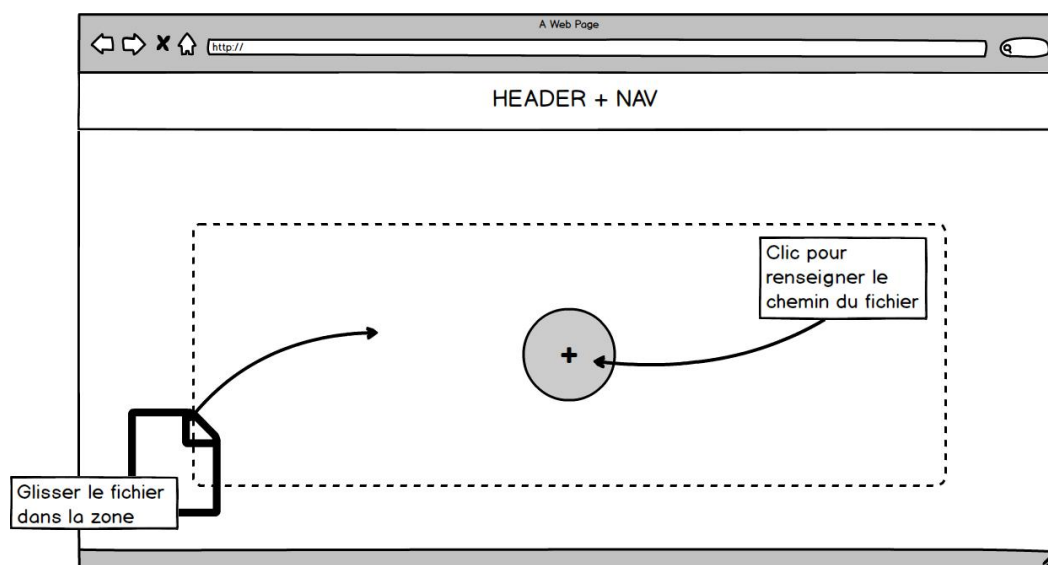
[style=unboxed,leftmargin=0.2cm]Ne demande aucun paramètre en entrée.

Retour : variable de type **str** - chaîne de caractères représentant le chemin relatif ou absolu du fichier CSV fourni.

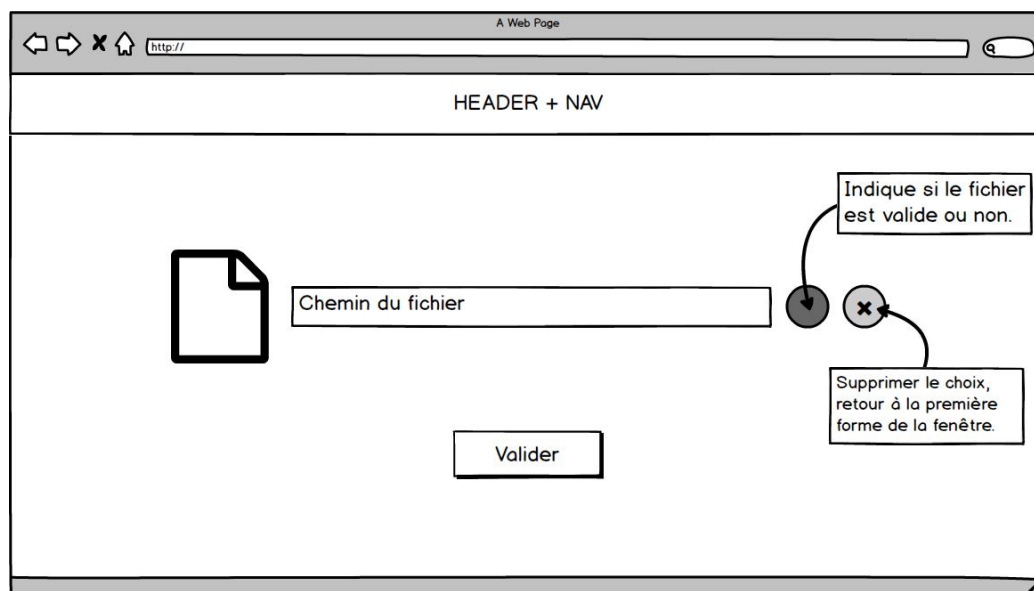
Description : cette fonction ne prend pas de paramètre en entrée. Elle permet de récupérer l'emplacement du fichier CSV en parcourant le système de gestion de fichiers. Elle renvoie donc une chaîne de caractères correspondant à ce chemin.

3. Aspect visuel.

Le croquis ci-dessous représente la fenêtre avant d'avoir choisi un fichier. Le chemin relatif ou absolu sera être renseigné soit en déplaçant dans la zone en pointillé ce dernier, soit en cliquant sur le symbole "+" pour parcourir le système de gestion de fichiers :



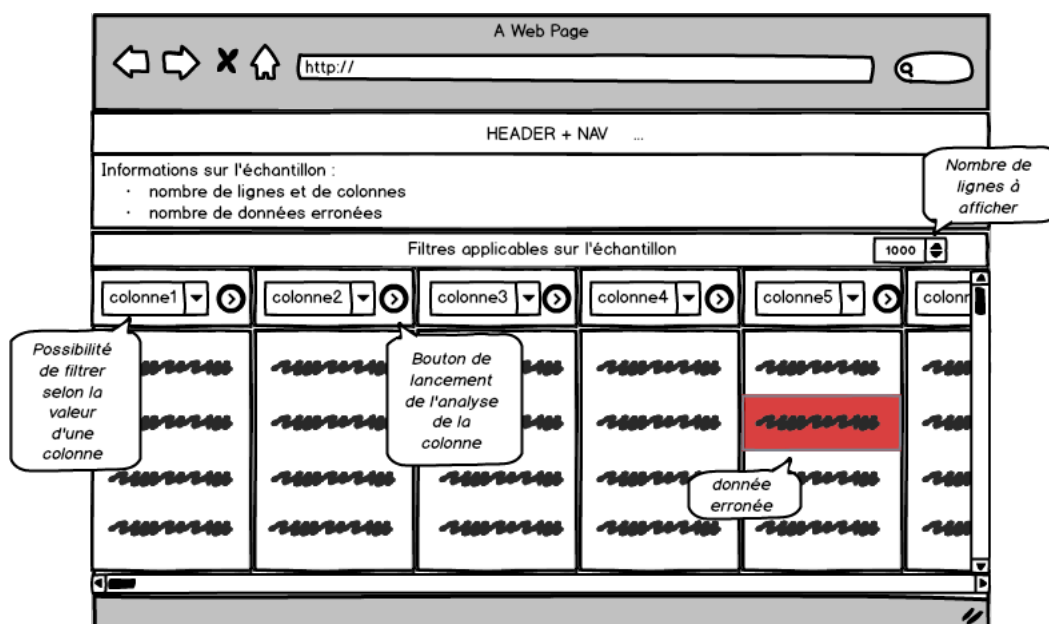
La fenêtre prendra ensuite l'apparence du croquis suivant lorsque l'utilisateur aura choisi un fichier. Son chemin sera affiché à l'écran. A côté de celui-ci sera affiché un indicateur pour la validité du fichier, et un bouton pour supprimé le choix. Dans le dernier cas, la fenêtre reprendra sa première apparence. Puis un bouton sera afficher pour valider le choix :



Les deux fenêtres abordent une apparence simple et épuré. L'effet cherché est que lorsque l'utilisateur regardera la fenêtre, il devra tout de suite comprendre comment elle fonctionnera.

### 3.3 Module Fenêtre rôle et choix colonne

Ce module est une fenêtre graphique web qui permet d'une part d'afficher un échantillon filtré des données du fichier et ses informations relatives, et d'autre part, de choisir une colonne de cet échantillon pour l'analyser. Ce module aura 3 fonctionnalités qui peuvent être illustrées par ce croquis :

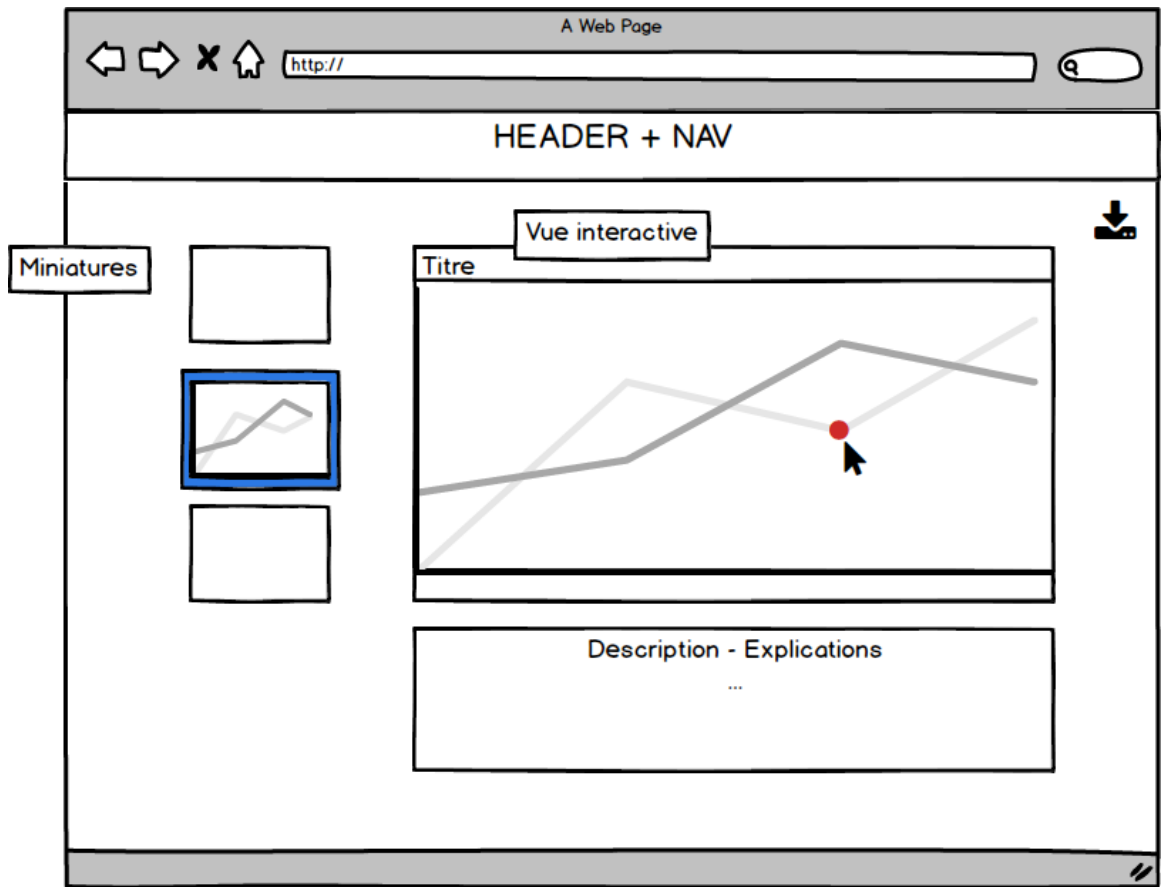


1. Fonctionnalité d'affichage de l'échantillon :
2. Fonctionnalité de filtrage de l'échantillon :
3. Fonctionnalité de choix de colonne à analyser :

### 3.4 Module Fenêtre résultats ADD

Ce module décrit la fenêtre dans laquelle l'utilisateur de l'applet pourra consulter les résultats d'analyse des colonnes du fichier importé.

1. Aspect visuel :



Les résultats doivent se présenter de manière simple et sans surcharge. Les utilisateurs de ce produit doivent pouvoir prendre en main et comprendre les données sans pour autant être formés.

## 2. Interactions avec l'utilisateur :

- Miniatures : Un clic sur une miniature permet de la faire basculer en vue précise.
- Vue précise : Un affichage interactif des valeurs rend la consultation plus intuitive et la charge de la vue plus légère. (une valeur non-demandée n'est pas affichée)
- Sauvegarde des résultats : Une simple icône de téléchargement permettra de lancer la sauvegarde des résultats en local.

## 3. Vues précises :

Ci-dessous, une liste des représentations graphiques disponibles, ainsi que le type d'analyse qui permet de les afficher :

- Histogramme : ADD qualitatives
- Diagramme en secteur : ADD qualitatives
- Série temporelle : ADD quantitatives
- Distribution des valeurs : ADD quantitatives
- Distribution cumulative des valeurs : ADD quantitatives
- Diagramme à moustaches : ADD quantitatives

## 4 Glossaire des types

Dans ce glossaire, on va préciser la définition des types des données qu'on a utilisé dans la description des fonctionnalités.

- `TextIoWrapper` : classe représentant les flux de texte bufferisés (entrées/sorties de texte avec une sauvegarde dans une mémoire tampon). Elle permet donc de représenter les fichiers de texte brut en particulier. Elle définit des méthodes de manipulation de ces flux.
- `str` : classe représentant des séquences de caractères (donc des chaînes de caractères) en Python. Elle définit des méthodes de manipulation de ces chaînes.
- `float` : classe représentant les nombres flottants à double précision en Python.
- `list` : classe représentant des séquences de variables de type quelconque indexés par des entier. Elle définit des méthodes de manipulation de ces séquences.  
L'utilisation de ce type de données pour la manipulation des colonnes du fichier `.csv` est préconisée car, tout d'abord, le type et les fonctions de manipulations sont disponibles directement avec Python, ensuite, ces listes sont implémentées sous forme de tableau ce qui garantie l'accès rapide à chacun de ses éléments, et enfin, leur taille maximale est définie par l'entier `sys.maxsize`<sup>4</sup> qui dépend des machines mais dépasse largement le million sur des PC bureautiques standards. Le million correspond à une borne maximale du nombre de lignes du fichier `.csv` importé, la manipulation des listes peut donc se faire sans risque de crash mémoire.
- `dict` : classe représentant des séquences de couple clés, variables (indexés par cette clé). Elle définit des méthodes de manipulation de ces séquences.
- `tuple` : classe représentant des listes immuables (non modifiables).
- `.json` : (JavaScript Object Notation) format de données textuelles basé sur la notation des objets JavaScript. C'est un moyen simple d'accéder à des données car un flux JSON est une sérialisation (une traduction) d'un objet JavaScript.

---

4. <https://docs.python.org/2/library/sys.htmlsys.maxsize>

— Intervalle : classe à implémenter

```
1 class Intervalle:  
2
```

#### Attributs :

- borneInf : **float** - limite inférieure de l'intervalle
- borneSup : **float** limite supérieure de l'intervalle
- infInclus : boolean - vrai si l'intervalle est fermé à gauche
- supInclus : boolean - vrai si l'intervalle est fermé à droite
- centre : **float** - centre de l'intervalle calculé dans le constructeur

#### Méthodes :

```
1 def __init__(self, borneInf, borneSup,  
2     infInclus, supInclus)
```

#### Description : Constructeur

```
1 def contient(nombre)  
2
```

#### Paramètres :

[style=unboxed,leftmargin=0.2cm]nombre : **float**

Retour : boolean

Description : Renvoie vrai si nombre est contenu dans l'intervalle, faux sinon.

## Conclusion

Bilan : rappel ce qu'on a fait et ce que ça a apporté

Ouvertures :

- Difficultés
- Limites des spécifications dans notre cas : nécessité de tester une 1ere version de l'appli pour bien définir l'expérience utilisateur

Dernière phrase positive :

- Découverte nouveaux outils, nouvelle démarche pour le dev : devops, ux design, balsamiq, les app web (javascript c'est assez bien en vrai)