# Inputs : paramètres XML

L’ensemble des paramètres répertoriés dans ce fichier XML correspond à des noms de variables de l’application développée sur python. Sur python, une fonction permet de charger le fichier de paramètres XML et de récupérer la valeur des variables. La signification de chacun des paramètres est détaillée ci-dessous.

**Groupe « Files » : Utilisation par un développeur**

Ce groupe recense les noms des fichiers utilisés par l’application python.

* **datasets**: Il s’agit du nom du fichier contenant l’ensemble des données nécessaires au fonctionnement de l’application python (principalement des séries temporelles). Dans le « faux » backend il n’y a pas de données. Ce paramètre n’est donc pas à manipuler.

**Groupe « ForecastInputs » : Utilisation par un acteur de l’eau**

Ce groupe contient l’ensemble des paramètres nécessaires pour lancer une prévision de la ressource en eau. **Ce sont ces paramètres qui seront modifiés par un acteur de l’eau potable en Bretagne**.

* **watershed\_target\_id** : Il s’agit de l’identifiant national d’une station hydrologique. Une station hydrologique mesure les débits de rivières au pas de temps journalier. Pour cette application, cela nous permet d’identifier chaque cours d’eau par un identifiant. Comme le fichier associé au paramètre datasets n’est pas disponible, un fichier au format csv recensant les identifiants des cours d’eau a été ajouté au dossier backend (map.csv).
* **forecast\_horizon** : Il s’agit de l’échéance de la prévision en nombre de jours. Certains acteurs de l’eau souhaiteront anticiper le niveau de leur ressource 1 semaine à l’avance, d’autres 3 mois à l’avance.

**Groupe « Similarity » : Utilisation par un développeur**

Ce groupe contient des paramètres influençant le calcul de la prévision saisonnière. C’est **un groupe utilisé uniquement par le développeur scientifique pour y effectuer des tests**. Le nom de ce groupe provient de la méthode de calcul utilisée pour réaliser les prévisions. Elles sont basées sur le principe de similarité des séries temporelles. Concrètement, cela signifie que l’on va chercher dans le passé les événements climatiques et hydrologiques qui ressemblent à celui en cours. Ce groupe contient alors des sous-groupes de paramètres correspondant aux différentes données temporelles utilisées dans l’application (**specific\_discharge** pour le débit spécifique des cours d’eau; **recharge** pour la quantité d’eau qui s’infiltre dans les sols rechargeant les nappes; **runoff** pour la quantité d’eau qui ruisselle à la surface et rejoint rapidement les cours d’eau).

Pour chaque catégorie de données (specific\_discharge, recharge, runoff), il y a un groupe de paramètres pour le calcul des similarités/corrélations entre deux séries temporelles et un groupe pour la sélection des scénarios, c’est-à-dire la sélection des simulations correspondant aux coefficients de corrélations supérieurs à un seuil défini. Les paramètres sont donc les suivants :

* **correlation** : Il s’agit du nom de la méthode pour calculer la corrélation entre deux séries temporelles.
* **scale** : Pour certaines données comme les débits de rivières, il peut être nécessaire de convertir les données en logarithmes. Ce paramètre permet de définir l’échelle des données (normale ou logarithme).
* **threshold** : Il s’agit du seuil minimal de sélection des scénarios. Seuls les coefficients de corrélation supérieurs à ce seuil sont conservés pour le calcul des prévisions saisonnières.

**Groupe « TimeManagement » : Utilisation par un développeur**

Ce groupe définit la façon de gérer la dimension temporelle de l’application, soit le pas de temps des séries temporelles et la période de calcul des similitudes entre deux séries temporelles. Ce groupe est uniquement utilisé par le développeur scientifique pour calibrer et tester son application. Il contient les paramètres suivants :

* **similarity\_period** : La période de calcul de la similarité peut-être soit une période bien définie (par exemple entre mars et mai), soit un nombre de jours avant le jour de la prévision (90 jours avant la date de prévision).
* **ndays\_before\_forecast**: Si le paramètre précédant définit la période de similarité comme un nombre de jour, ce nombre sera alors définit dans ce paramètre.
* **time\_step** : Il s’agit du pas de temps des variables. Elles peuvent être journalières (D), hebdomadaires (W) ou mensuelles (M).

**Conseils pour les sujets dédiés à l’interface des paramètres (sujets 1,2,3) :**

Note : Ces pistes sont indicatives, nous laissons une liberté de développement et la possibilité d’utiliser une méthode non évoquée.

**Sujet 1 :** **Affichage différencié du fichier de paramètres xml en fonction du rôle de l’utilisateur**

Les deux objectifs de ce sujet sont :

* La définition des rôles d’utilisateur (par exemple : développeur/acteur de l’eau; débutant/avancé).
* L’affichage différencié des groupes en fonction du rôle utilisateur.

Pour une version simplifié (pour l’acteur de l’eau), nous souhaitons visualiser uniquement le groupe « ForecastInputs ».

**Sujet 2 :**  **Interaction des paramètres en fonction de la valeur de saisie**

L’objectif principal de ce sujet est :

* L’implémentation d’une méthode pour interagir sur la valeur et/ou sur l’affichage d’autres paramètres.

Pour le moment il n’y aucune interaction entre les différents paramètres disponibles. Aucun paramètre n’a d’influence sur un autre. Voici quelques pistes de réflexion :

Pour modifier l’affichage en fonction de la valeur de saisie :

* Si la valeur de similarity\_period est équivalent à « period », désactiver l’affichage du paramètre ndays\_before\_forecast. En effet ce dernier paramètre n’a d’importance que lorsque que similarity\_period est équivalent à « ndays ».
* Créer un paramètre « User » et désactiver l’affichage des groupes de paramètres Files, Similarity et TimeManagement si User correspond à un utilisateur débutant et donc au mode simplifié de l’interface.

Pour modifier la valeur des paramètres en fonction de la valeur de saisie :

* Pour le moment laisser cette partie de côté et se concentrer sur l’affichage en fonction de la valeur de saisie.

**Sujet 3 : Affichage simplifié et ergonomique du fichier de paramètres xml**

L’objectif principal de ce sujet est :

* Le remplacement de l’interface complexe des paramètres XML (qui existerait toujours dans un mode avancé mais qui ne serait pas visible d’un utilisateur débutant) par la mise en place de widgets. Ces widgets permettront de modifier la valeur de certains paramètres XML.

Les paramètres à modifier sont ceux du groupe « ForecastInputs », soit le choix du cours d’eau (à partir de son identifiant, par exemple une liste déroulante) et l’échéance de la prévision (par exemple un « Slider ».

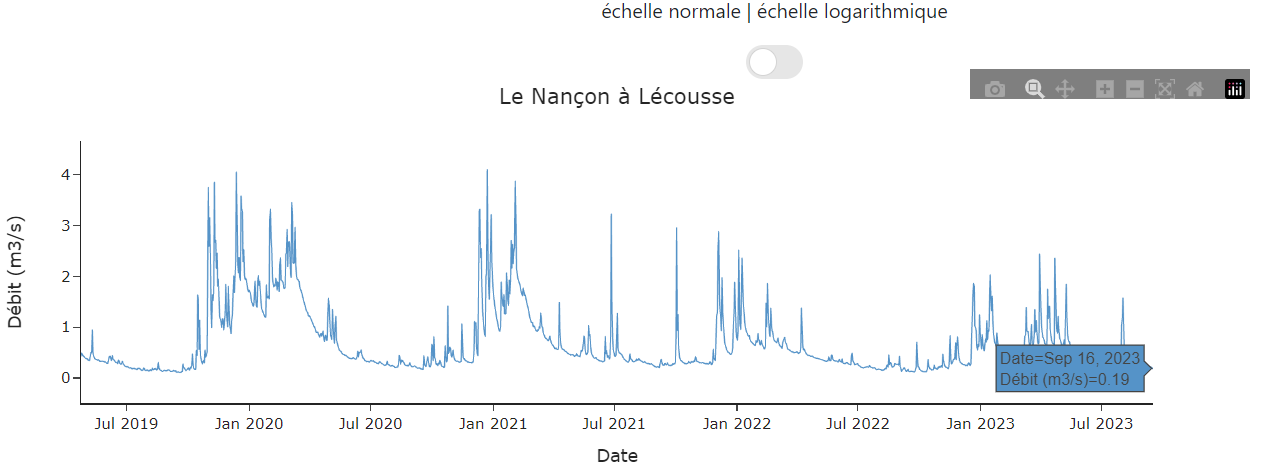
**Les sujets 4, 5, 6 et 7 n’interviennent pas sur la manipulation et l’affichage des entrées de l’application.**

# Outputs : Carte, graphique, tableau

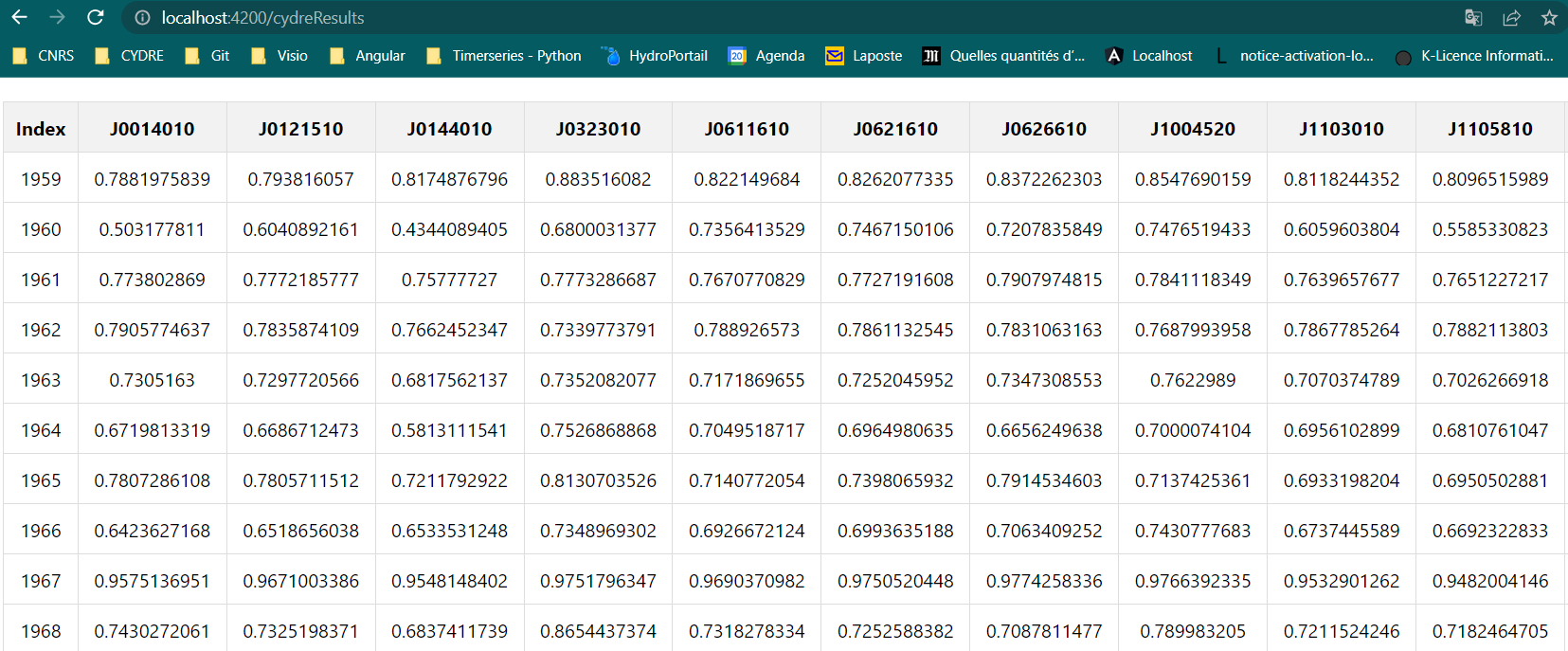
Cette partie traite de l’affichage et de la manipulation des sorties de l’application python. L’application python n’étant pas encore disponible, un jeu de données et des exemples de sorties ont été fournis dans le dossier « data » du « faux » backend.

Exemples de sorties de l’application python :

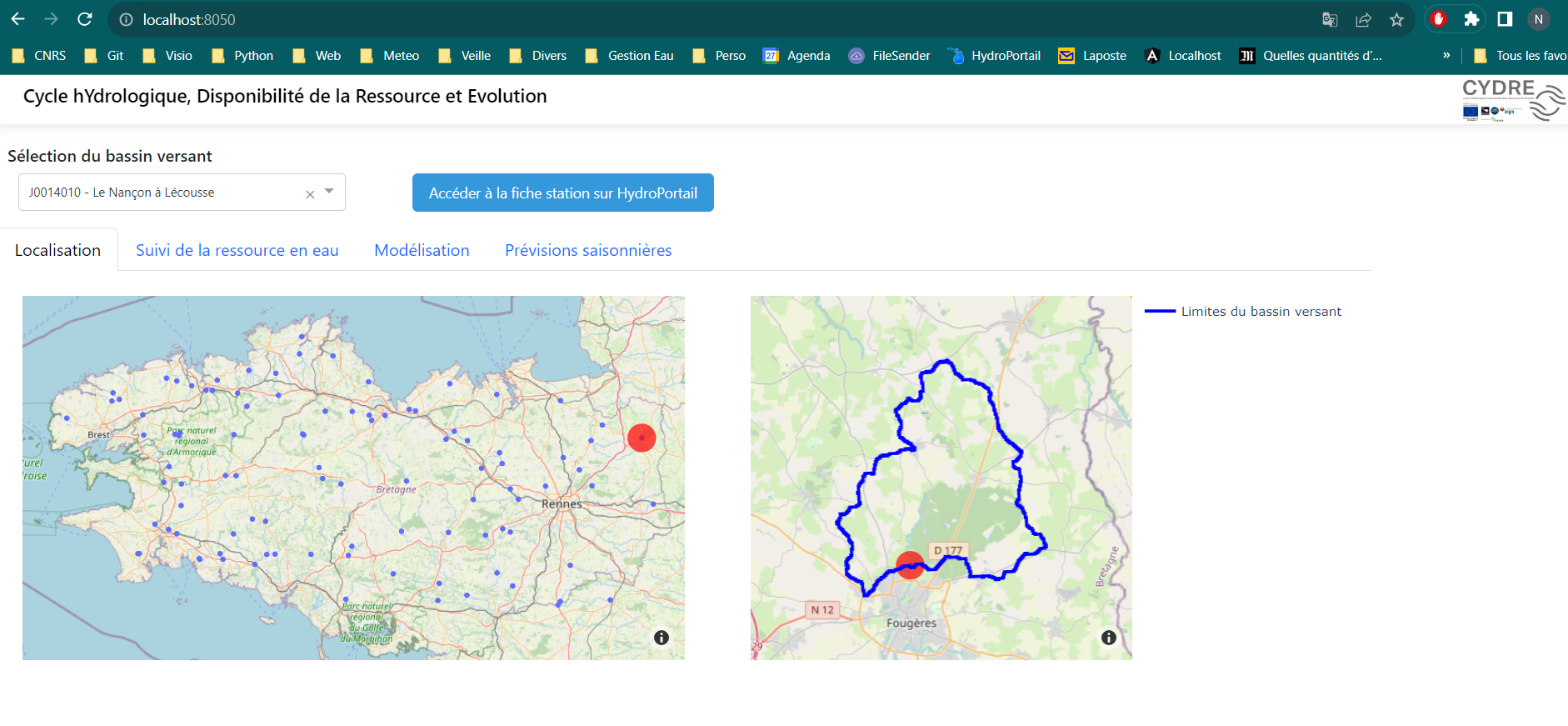
* Courbe (scatter plot) représentant l’évolution d’une donnée en fonction du temps (exemple : l’évolution des débits sur le cours d’eau du Nançon mesurée dans la commune de Lécousse). Le jeu de données d’exemple associé est « scatter\_plot.csv ».



* Tableau synthétisant les coefficient de corrélation de la recharge de nappe d’un cours d’eau avec l’ensemble des autres cours d’eau de la région. Dans cet exemple chaque cellule représente la corrélation entre 1) la recharge de la Loisance en 2023 et 2) la recharge de l’ensemble des cours d’eau bretons sélectionnés et leurs années de mesure. Pour simplifier l’ergonomie et l’accessibilité de ce tableau, les cellules pourraient être symbolisées par une couleur à la place des valeurs (par exemple : dégradé d’une couleur; rouge pour les coefficients de corrélation élevés et bleu pour les moins élevés, etc…). Le jeu de données d’exemple associé est « table.csv ».



* Carte représentant la localisation des stations de mesure des débits des cours d’eau. Le jeu de données d’exemple associé est « map.csv ».



Note : Voici quelques pistes de réflexions pour les différents sujets associés à l’affichage et à la manipulation des sorties du code python. Ces pistes sont indicatives, nous laissons une liberté de développement et la possibilité d’utiliser une méthode non évoquée ou d’utiliser une version simplifiée des données.

**Conseils pour les sujets dédiés à l’interface des sorties (sujets 4,5,6) :**

**Sujet 4 : Ergonomie des sorties de l’application python**

Ce sujet a pour objectif principal d’afficher une sortie python sur l’interface web et de la rendre interactive (étiquette affichant la valeur d’une courbe, capacité de zoomer un graphique, etc..)

Il s’agit de :

1. Créer un bouton sur l’interface web pour déclencher une fonction sur le « faux » backend.
2. Récupérer l’information de la fonction python : un graphique (développé avec la librairie plotly) ou un tableau de valeur (au format json).
3. Afficher le graphique ou le tableau sur l’interface.

Plusieurs possibilités :

* Récupérer le jeu de données brut et réaliser le graphique ou le tableau dans Angular.
* Récupérer le graphique plotly, réalisé sous python et converti en json, et l’afficher dans le code html à partir de la balise <plotly-plot [src]="'assets/mon\_graphe\_plotly.json'" [style]="{ width: 'X%', height: 'Xpx' }"></plotly-plot> (à vérifier, d’autres options sont possibles).

Un jeu de données et des scripts d’exemples ont été ajoutés pour tester différent types de sorties.

**Sujet 5 : Historique des prévisions saisonnières**

Ce sujet a pour objectif de stocker les résultats d’une prévision. Il nécessite la démarche globale détaillée dans le sujet précédent (bouton pour exécuter une fonction python et affichage d’une sortie). Pour simplifier, le groupe traitant de ce sujet peut utiliser une seule sortie simple de l’application (par exemple le tableau).

L’application n’utilisant pas de base de données distante, la sauvegarde pourrait se faire par exemple sur l’ordinateur du client avec une possibilité de charger la sauvegarde pour l’afficher sur le serveur plusieurs jours après la prévision. Une autre possibilité est de stocker les sorties côté serveur dans un dossier spécifique à chaque client. Nous sommes ouvert à toutes propositions sur ce sujet.

**Sujet 6 : Amélioration de l’expérience utilisateur pendant le chargement de l’application**

Ce sujet a pour objectif d’apporter des informations au client pendant le chargement d’une application. Cela peut passer par une indication du temps de chargement mais aussi par la visualisation de sorties python intermédiaires même si le code n’a pas fini d’être exécuté (par exemple en fragmentant le code python en plusieurs fonctions).

Un script exemple a été fourni avec différentes fonctions générant des sorties différentes. Il est possible de développer un script générant les 3 sorties en simulant des temps de pauses. Toutes les propositions sont bonnes à prendre sur ce sujet.

**Les sujets 1, 2, 3 et 7 n’interviennent pas sur les entrées de l’application.**