Dimensionner la récupération d'eau de pluie pour être autonome

Aurelpere

https://wiki.lowtechlab.org/wiki/Dimensionner_la_r%C3%A9cup%C3%A9ration_d%27eau_de_pluie_pour_%C3%AAtre_autonome

Dernière modification le 11/03/2024

Difficulté **Moyen**

Durée **1 heure(s)**

Coût **0 EUR (€)**

Description

Dimensionner la récupération d'eau de pluie hors réseau

Sommaire

Sommaire

Introduction

Dans des cas où on souhaite être 100% autonome hors réseau, la question de l'eau est essentielle.
C'est d'ailleurs le premier élément à considérer par exemple dans les démarches de permaculture (phase d'observation).
J'ai initialement fait les calculs ci dessous pour "autonomiser" un mobile home avec l'idée d'utiliser des modules photovoltaïques dans un assemblage récupérateur d'eau, comme le proposait le concept ulta chaata (https://www.facebook.com/weultachaata/?locale=fr_FR et https://fr.futuroprossimo.it/2023/03/ulta-chaata-ombrello-magico-che-puo-dare-acqua-e-luce-allindia/) avant d'être récupérée à Los Angeles.

On peut s'interroger sur la façon adéquate de dimensionner des installations pour récupérer de l'eau de pluie.

Pour cela, on peut utiliser les données de Météo France afin d'avoir un regard rétrospectif sur les précipitations saisonnières et ajuster ainsi les dimensions des récupérateurs.

Démo web interactive ici :
https://vpn.matangi.dev/water

Étape 1 - Prérequis logiciel

Dans ce tuto, on utilise des données météos synop disponibles ici :
http://data.cquest.org/meteo-france/synop/ dont vous trouverez la description ici:
http://data.cquest.org/meteo-france/synop/doc_parametres_synop_168.pdf

Vous pouvez également les télécharger sur le site de Meteo France:
https://donneespubliques.meteofrance.fr/?fond=produit&id_produit=90&id_rubrique=32

Télécharger tous les fichiers csv des années et mois à partir desquels vous souhaitez que les calculs soient effectués, placer les dans un repertoire de votre choix et dezipper les (format archive gz). Placer également dans ce répertoire le fichier processing.py contenant le code partagé dans ce tutoriel.

Exemple en ligne de commande linux debian pour télécharger et dézipper tous les csv de l'année 2020 dans un repertoire ~/synop:
(Dans la suite du tutoriel on utilise toutes les données des années 2010 à 2020)

```
cd ~

mkdir -p synop && cd synop


wget http://data.cquest.org/meteo-france/synop/synop.202001.csv.gz && gzip -d synop.202001.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202002.csv.gz && gzip -d synop.202002.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202003.csv.gz && gzip -d synop.202003.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202004.csv.gz && gzip -d synop.202004.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202005.csv.gz && gzip -d synop.202005.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202006.csv.gz && gzip -d synop.202006.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202007.csv.gz && gzip -d synop.202007.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202008.csv.gz && gzip -d synop.202008.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202009.csv.gz && gzip -d synop.202009.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202010.csv.gz && gzip -d synop.202010.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202011.csv.gz && gzip -d synop.202011.csv.gz

wget http://data.cquest.org/meteo-france/synop/synop.202012.csv.gz && gzip -d synop.202012.csv.gz
```

Pour utiliser python sous un autre os que linux, débrouillez vous avec vos daubes propriétaires intrusives.

Sous linux, python est généralement installé et pour utiliser le code partagé dans ce tuto, il suffira de copier coller le code dans un fichier processing.py puis entrer

```
python processing.py
```

Cependant, il faut installer la librairie pandas qui est très largement utilisé dans le monde de la finance et dans le monde scientifique, notamment pour sa gestion efficace des séries temporelles et sa capacité de vectorisation des données.

Pour cela voici les commandes à entrer dans linux debian avant de lancer le script processing.py pour être tranquille:

```
sudo apt install python3 python3-venv python3-pip python-is-python3

cd ~ && python -m venv venv

source venv/bin/activate

pip install pandas
```

Penser ensuite à activer l'environnement virtuel dans lequel est installé pandas chaque fois que vous utilisez le script (apres un reboot ou si vous fermez et reouvrez votre terminal) en lançant la commande:

```
cd ~ && source venv/bin/activate
```

Nous sommes en 2024 et si vous êtes cibles d'entraves anti éco-terroristes de psychopathes comme moi, et en bon scientifique qui se respecte vous inspectez vos instruments de mesures avant de les utiliser, vous pouvez inspectez le code source de pandas qui est évidemment logiciel libre ici : https://github.com/pandas-dev/pandas , ou vous pouvez faire l'hypothèse discutable qu'on peut avoir confiance dans un logiciel aussi massivement utilisé dans le monde de la finance et de la science.
Python reposant sur des librairies C pour un certains nombres d'opérations de base, le hack, y compris le hack scientifique, n'est jamais impossible mais on laissera de coté ces considérations pro-lowtech qui n'entrent pas dans le champ de réflexion de ce tutoriel.

Étape 2 - Evaluer les besoins

Pour évaluer les besoins pour des installations domestiques, rien de plus efficace que le compteur d'eau. Une première approche peut être de faire une règle de trois à partir de votre consommation hebdomadaire. Vous pouvez aussi mesurer individuellement chaque poste de consommation (douche, machine à laver, cuisine, jardin, wc, etc) afin de faire des projections saisonnières plus justes.
Pour un mobile home avec toilettes sèches on a :

```
conso solo - 2 douches semaine (L)

douche (L)   50

boisson    4

vaisselle   10

cuisine    4

machine a laver   50

semaine (2douches, 1 machine) 276

journalier   39

trimestre (13 semaines)  3588


conso à deux - 4 douches semaine (L)

douche    50

boisson    4

vaisselle   10

cuisine    4

machine a laver   50

semaine (4douches, 1 machine) 752

journalier   107

trimestre (13 semaines)  9776
```

Étape 3 - Calculer rétrospectivement les précipitations journalieres et saisonnieres

Pour dimensionner correctement le stockage, il convient d'abord d'avoir des informations sur les précipitations "moyennes" des années précédentes.
Pour cela, on fournit le bout de code python suivant (valable avec les données météo france mais adaptable à d'autres données météos ailleurs, le synop étant un encodage de données utilisé par l'OMM):

```
import math
import os
import pandas as pd

# Attention si vous utilisez ce bout de code dans d'autres pays que la france, il faut ajouter
# les stations météos adhoc

# Processing des data
print("\nprocessing des data\n")
files=os.listdir('.')
csv=[a for a in files if a[-3:]=='csv']
combined_df = pd.concat((pd.read_csv(f,sep=';') for f in csv), ignore_index=True)
#07510 bordeaux
#07535 gourdon

#stations météos "hard coded"
stations=[{'ID': '07005', 'Nom': 'ABBEVILLE', 'Latitude': '50.136000', 'Longitude': '1.834000', 'Altitude': '69'}, {'ID': '07015', 'Nom': 'LILLE-LESQUIN', 'Latitude'
```

: '50.570000', 'Longitude': '3.097500', 'Altitude': '47'}, {'ID': '07020', 'Nom': 'PTE DE LA HAGUE', 'Latitude': '49.725167', 'Longitude': '-1.939833', 'Altitude': '6'
}, {'ID': '07027', 'Nom': 'CAEN-CARPIQUET', 'Latitude': '49.180000', 'Longitude': '-0.456167', 'Altitude': '67'}, {'ID': '07037', 'Nom': 'ROUEN-BOOS', 'Latitude':
'49.383000', 'Longitude': '1.181667', 'Altitude': '151'}, {'ID': '07072', 'Nom': 'REIMS-PRUNAY', 'Latitude': '49.209667', 'Longitude': '4.155333', 'Altitude': '95'}, {
'ID': '07110', 'Nom': 'BREST-GUIPAVAS', 'Latitude': '48.444167', 'Longitude': '-4.412000', 'Altitude': '94'}, {'ID': '07117', 'Nom': "PLOUMANAC'H", 'Latitude': '
48.825833', 'Longitude': '-3.473167', 'Altitude': '55'}, {'ID': '07130', 'Nom': 'RENNES-ST JACQUES', 'Latitude': '48.068833', 'Longitude': '-1.734000', 'Altitude'
: '36'}, {'ID': '07139', 'Nom': 'ALENCON', 'Latitude': '48.445500', 'Longitude': '0.110167', 'Altitude': '143'}, {'ID': '07149', 'Nom': 'ORLY', 'Latitude': '48.716833',
'Longitude': '2.384333', 'Altitude': '89'}, {'ID': '07168', 'Nom': 'TROYES-BARBEREY', 'Latitude': '48.324667', 'Longitude': '4.020000', 'Altitude': '112'}, {'ID': '0
7181', 'Nom': 'NANCY-OCHEY', 'Latitude': '48.581000', 'Longitude': '5.959833', 'Altitude': '336'}, {'ID': '07190', 'Nom': 'STRASBOURG-ENTZHEIM', 'Latitude'
: '48.549500', 'Longitude': '7.640333', 'Altitude': '150'}, {'ID': '07207', 'Nom': 'BELLE ILE-LE TALUT', 'Latitude': '47.294333', 'Longitude': '-3.218333', 'Altitude'
: '34'}, {'ID': '07222', 'Nom': 'NANTES-BOUGUENAIS', 'Latitude': '47.150000', 'Longitude': '-1.608833', 'Altitude': '26'}, {'ID': '07240', 'Nom': 'TOURS', 'Latitud
e': '47.444500', 'Longitude': '0.727333', 'Altitude': '108'}, {'ID': '07255', 'Nom': 'BOURGES', 'Latitude': '47.059167', 'Longitude': '2.359833', 'Altitude': '161'}, {'I
D': '07280', 'Nom': 'DIJON-LONGVIC', 'Latitude': '47.267833', 'Longitude': '5.088333', 'Altitude': '219'}, {'ID': '07299', 'Nom': 'BALE-MULHOUSE', 'Latitude': '
47.614333', 'Longitude': '7.510000', 'Altitude': '263'}, {'ID': '07314', 'Nom': 'PTE DE CHASSIRON', 'Latitude': '46.046833', 'Longitude': '-1.411500', 'Altitude':
'11'}, {'ID': '07335', 'Nom': 'POITIERS-BIARD', 'Latitude': '46.593833', 'Longitude': '0.314333', 'Altitude': '123'}, {'ID': '07434', 'Nom': 'LIMOGES-BELLEGAR
DE', 'Latitude': '45.861167', 'Longitude': '1.175000', 'Altitude': '402'}, {'ID': '07460', 'Nom': 'CLERMONT-FD', 'Latitude': '45.786833', 'Longitude': '3.149333', 'A
ltitude': '331'}, {'ID': '07471', 'Nom': 'LE PUY-LOUDES', 'Latitude': '45.074500', 'Longitude': '3.764000', 'Altitude': '833'}, {'ID': '07481', 'Nom': 'LYON-ST EXU
PERY', 'Latitude': '45.726500', 'Longitude': '5.077833', 'Altitude': '235'}, {'ID': '07510', 'Nom': 'BORDEAUX-MERIGNAC', 'Latitude': '44.830667', 'Longitude': '
-0.691333', 'Altitude': '47'}, {'ID': '07535', 'Nom': 'GOURDON', 'Latitude': '44.745000', 'Longitude': '1.396667', 'Altitude': '260'}, {'ID': '07558', 'Nom': 'MILLAU',
'Latitude': '44.118500', 'Longitude': '3.019500', 'Altitude': '712'}, {'ID': '07577', 'Nom': 'MONTELIMAR', 'Latitude': '44.581167', 'Longitude': '4.733000', 'Altitude
': '73'}, {'ID': '07591', 'Nom': 'EMBRUN', 'Latitude': '44.565667', 'Longitude': '6.502333', 'Altitude': '871'}, {'ID': '07607', 'Nom': 'MONT-DE-MARSAN', 'Latitude'
: '43.909833', 'Longitude': '-0.500167', 'Altitude': '59'}, {'ID': '07621', 'Nom': 'TARBES-OSSUN', 'Latitude': '43.188000', 'Longitude': '0.000000', 'Altitude': '360'
}, {'ID': '07627', 'Nom': 'ST GIRONS', 'Latitude': '43.005333', 'Longitude': '1.106833', 'Altitude': '414'}, {'ID': '07630', 'Nom': 'TOULOUSE-BLAGNAC', 'Latitude
': '43.621000', 'Longitude': '1.378833', 'Altitude': '151'}, {'ID': '07643', 'Nom': 'MONTPELLIER', 'Latitude': '43.577000', 'Longitude': '3.963167', 'Altitude': '2'}, {'
ID': '07650', 'Nom': 'MARIGNANE', 'Latitude': '43.437667', 'Longitude': '5.216000', 'Altitude': '9'}, {'ID': '07661', 'Nom': 'CAP CEPET', 'Latitude': '43.079333', '
Longitude': '5.940833', 'Altitude': '115'}, {'ID': '07690', 'Nom': 'NICE', 'Latitude': '43.648833', 'Longitude': '7.209000', 'Altitude': '2'}, {'ID': '07747', 'Nom': 'PERP
IGNAN', 'Latitude': '42.737167', 'Longitude': '2.872833', 'Altitude': '42'}, {'ID': '07761', 'Nom': 'AJACCIO', 'Latitude': '41.918000', 'Longitude': '8.792667', 'Altit
ude': '5'}, {'ID': '07790', 'Nom': 'BASTIA', 'Latitude': '42.540667', 'Longitude': '9.485167', 'Altitude': '10'}, {'ID': '61968', 'Nom': 'GLORIEUSES', 'Latitude': '-11.
582667', 'Longitude': '47.289667', 'Altitude': '3'}, {'ID': '61970', 'Nom': 'JUAN DE NOVA', 'Latitude': '-17.054667', 'Longitude': '42.712000', 'Altitude': '9'}, {'ID':
'61972', 'Nom': 'EUROPA', 'Latitude': '-22.344167', 'Longitude': '40.340667', 'Altitude': '6'}, {'ID': '61976', 'Nom': 'TROMELIN', 'Latitude': '-15.887667', 'Longit
ude': '54.520667', 'Altitude': '7'}, {'ID': '61980', 'Nom': 'GILLOT-AEROPORT', 'Latitude': '-20.892500', 'Longitude': '55.528667', 'Altitude': '8'}, {'ID': '61996', 'N
om': 'NOUVELLE AMSTERDAM', 'Latitude': '-37.795167', 'Longitude': '77.569167', 'Altitude': '27'}, {'ID': '61997', 'Nom': 'CROZET', 'Latitude': '-46.432500', '
Longitude': '51.856667', 'Altitude': '146'}, {'ID': '61998', 'Nom': 'KERGUELEN', 'Latitude': '-49.352333', 'Longitude': '70.243333', 'Altitude': '29'}, {'ID': '67005',
'Nom': 'PAMANDZI', 'Latitude': '-12.805500', 'Longitude': '45.282833', 'Altitude': '7'}, {'ID': '71805', 'Nom': 'ST-PIERRE', 'Latitude': '46.766333', 'Longitude': '-
56.179167', 'Altitude': '21'}, {'ID': '78890', 'Nom': 'LA DESIRADE METEO', 'Latitude': '16.335000', 'Longitude': '-61.004000', 'Altitude': '27'}, {'ID': '78894', 'N
om': 'ST-BARTHELEMY METEO', 'Latitude': '17.901500', 'Longitude': '-62.852167', 'Altitude': '44'}, {'ID': '78897', 'Nom': 'LE RAIZET AERO', 'Latitude': '16.2
64000', 'Longitude': '-61.516333', 'Altitude': '11'}, {'ID': '78922', 'Nom': 'TRINITE-CARAVEL', 'Latitude': '14.774500', 'Longitude': '-60.875333', 'Altitude': '26'},
{'ID': '78925', 'Nom': 'LAMENTIN-AERO', 'Latitude': '14.595333', 'Longitude': '-60.995667', 'Altitude': '3'}, {'ID': '81401', 'Nom': 'SAINT LAURENT', 'Latitude':
'5.485500', 'Longitude': '-54.031667', 'Altitude': '5'}, {'ID': '81405', 'Nom': 'CAYENNE-MATOURY', 'Latitude': '4.822333', 'Longitude': '-52.365333', 'Altitude': '4
'}, {'ID': '81408', 'Nom': 'SAINT GEORGES', 'Latitude': '3.890667', 'Longitude': '-51.804667', 'Altitude': '6'}, {'ID': '81415', 'Nom': 'MARIPASOULA', 'Latitude':
'3.640167', 'Longitude': '-54.028333', 'Altitude': '106'}, {'ID': '89642', 'Nom': "DUMONT D'URVILLE", 'Latitude': '-66.663167', 'Longitude': '140.001000', 'Altitu
de': '43'}]

```
def distance(lat1, lon1, lat2, lon2):
    """
    Calcule la distance entre deux points géographiques en utilisant la formule de la distance euclidienne.
    """
    return math.sqrt((lat2 - lat1)**2 + (lon2 - lon1)**2)

def station_la_plus_proche(x, y, stations):
    """
    Trouve la station météo la plus proche en utilisant les coordonnées x et y (latitude et longitude).
    """
    distance_min = float('inf')
    station_proche = None

    for station in stations:
        lat_station = float(station['Latitude'])
        lon_station = float(station['Longitude'])
        d = distance(x, y, lat_station, lon_station)
        if d < distance_min:
            distance_min = d
            station_proche = station

    return station_proche

# Demander à l'utilisateur d'entrer la latitude et la longitude
x_input = input("Entrez la latitude de votre lieux: ")
y_input = input("Entrez la longitude de votre lieux: ")

# Remplacer les virgules par des points
x_input = float(x_input.replace(',', '.'))
y_input = float(y_input.replace(',', '.'))
```

```
# Utilisez les valeurs entrées par l'utilisateur comme variables x et y pour trouver la station météo la plus proche
station_proche = station_la_plus_proche(x_input, y_input, stations)
print("La station météo la plus proche est:", station_proche['Nom'])

result=combined_df[combined_df['numer_sta']==int(station_proche['ID'])]

# Convertir la colonne 'date_column' dans un format datetime et la mettre en index trié
result['datetime'] = pd.to_datetime(result['date'], format='%Y%m%d%H%M%S')
result.set_index('datetime', inplace=True)
result = result.sort_index()

# remplacer les données manquantes par 0
result['rr3']=result['rr3'].replace('mq','0')
result['rr3']=result['rr3'].astype('float')

# Ne garder que la colonne des précipitations des 3 dernieres heures
result=result['rr3']

# Calculer les sommes de précipitations par jour
resultday=result.resample('D').sum()
print("\nMoyenne par jour (mm):\n", resultday.mean())
print("Minimum par jour (mm):\n", resultday.min())
print("Maximum par jour  (mm):\n", resultday.max())


# Calculer les sommes de précipitations par semaine
resultweek=result.resample('W').sum()

# Calculer les sommes de précipitations par mois
resultmonth=result.resample('ME').sum()

# Calculer les sommes de précipitations par trimestre
resulttrim=result.resample('QE').sum()
resulttrim=resulttrim.rename_axis('trimestre')
print(resulttrim)

# Calculer les sommes de précipitations par an
resultyear=result.resample('YE').sum()
print("\nPrécipitations annuelles moyennes (mm):\n",resultyear.mean())

# Calculer le nombre de jours consécutifs maximum sans pluie
max_streak = 0
current_streak = 0
for value in resultday:
    if value == 0:
        current_streak += 1
        max_streak = max(max_streak, current_streak)
    else:
        current_streak = 0  # Reset the streak if the value is not zero
print(f"\nNombre de jours consecutifs maximum sans pluie: {max_streak}")

# Moyenne par trimestre pour chaque trimestre
moyenne_trimestrielle_par_trimestre = resulttrim.groupby(resulttrim.index.quarter).mean()

# Minimum par trimestre pour chaque trimestre
min_trimestrielle_par_trimestre = resulttrim.groupby(resulttrim.index.quarter).min()

# Maximum par trimestre pour chaque trimestre
max_trimestrielle_par_trimestre = resulttrim.groupby(resulttrim.index.quarter).max()

# Imprimer les résultats
print("\nMoyenne par trimestre pour chaque trimestre (mm):\n", moyenne_trimestrielle_par_trimestre)
print("\nMinimum par trimestre pour chaque trimestre (mm):\n", min_trimestrielle_par_trimestre)
print("\nMaximum par trimestre pour chaque trimestre (mm):\n", max_trimestrielle_par_trimestre)

# Minimum par jour pour chaque trimestre
min_par_jour_par_trimestre = resultday.groupby(resultday.index.quarter).min()
min_par_jour_par_trimestre=min_par_jour_par_trimestre.rename_axis('trimestre')

# Maximum par jour pour chaque trimestre
max_par_jour_par_trimestre = resultday.groupby(resultday.index.quarter).max()
max_par_jour_par_trimestre=max_par_jour_par_trimestre.rename_axis('trimestre')

# Moyenne par jour pour chaque trimestre
```

```
# Moyenne par jour pour chaque trimestre
moyenne_par_jour_par_trimestre = resultday.groupby(resultday.index.quarter).mean()
moyenne_par_jour_par_trimestre=moyenne_par_jour_par_trimestre.rename_axis('trimestre')

# Imprimer les résultats
print("\nMinimum par jour pour chaque trimestre (mm):\n", min_par_jour_par_trimestre)
print("\nMaximum par jour pour chaque trimestre (mm):\n", max_par_jour_par_trimestre)
print("\nMoyenne par jour pour chaque trimestre (mm):\n", moyenne_par_jour_par_trimestre)
```

Pour la latitude 44.2 et longitude 0.6 on obtient:

```
processing des data

Entrez la latitude de votre lieux: 44.2
Entrez la longitude de votre lieux: 0.6
La station météo la plus proche est: GOURDON

Moyenne par jour (mm):
 2.022896963663514
Minimum par jour (mm):
 -0.6000000000000001
Maximum par jour  (mm):
 55.0
trimestre
2010-03-31    202.0
2010-06-30    245.4
2010-09-30    132.2
2010-12-31    201.2
2011-03-31    126.7
2011-06-30    102.2
2011-09-30    164.6
2011-12-31    207.0
2012-03-31     99.8
2012-06-30    341.0
2012-09-30    100.0
2012-12-31    188.8
2013-03-31    248.4
2013-06-30    307.5
2013-09-30    136.6
2013-12-31    247.8
2014-03-31    253.8
2014-06-30    201.8
2014-09-30    192.8
2014-12-31    139.0
2015-03-31    176.4
2015-06-30    155.7
2015-09-30    184.6
2015-12-31     82.6
2016-03-31    322.1
2016-06-30    300.4
2016-09-30     29.0
2016-12-31    115.1
2017-03-31    213.0
2017-06-30    216.3
2017-09-30    133.2
2017-12-31    155.3
2018-03-31    252.8
2018-06-30    251.9
2018-09-30    103.7
2018-12-31    199.3
2019-03-31    100.1
2019-06-30    203.5
2019-09-30    138.8
2019-12-31    350.9
2020-03-31    149.5
2020-06-30    150.9
2020-09-30     66.9
2020-12-31    237.4
Freq: QE-DEC, Name: rr3, dtype: float64

Précipitations annuelles moyennes (mm):
 738.9090909090909
```

Nombre de jours consecutifs maximum sans pluie: 44

Moyenne par trimestre pour chaque trimestre (mm):
 trimestre
1   194.963636
2   225.145455
3   125.672727
4   193.127273
Name: rr3, dtype: float64

Minimum par trimestre pour chaque trimestre (mm):
 trimestre
1   99.8
2   102.2
3   29.0
4   82.6
Name: rr3, dtype: float64

Maximum par trimestre pour chaque trimestre (mm):
 trimestre
1   322.1
2   341.0
3   192.8
4   350.9
Name: rr3, dtype: float64

Minimum par jour pour chaque trimestre (mm):
 trimestre
1   -0.6
2   -0.4
3   -0.3
4   -0.5
Name: rr3, dtype: float64

Maximum par jour pour chaque trimestre (mm):
 trimestre
1   42.8
2   55.0
3   50.2
4   28.0
Name: rr3, dtype: float64

Moyenne par jour pour chaque trimestre (mm):
 trimestre
1   2.159718
2   2.474126
3   1.366008
4   2.099209
Name: rr3, dtype: float64

Étape 4 - Dimensionner la surfce de récupération et le stockage

Pour dimensionner, le stockage, on se rappellera utilement que 1m2 de surface donne un équivalent d'1L pour 1mm de précipitations.

On peut alors faire des calculs avec les précipitations moyennes précédemment estimées et les consommations moyennes précédemment mesurées.

Exemple pour 1m2:

annuel (L) 739

max jour (L) 55

min trimestre (L) 29

max trimestre (L) 350

moyenne min trimestre(L) 125

Besoins:

Max 44 j sans pluie solo (L) 1735

Max 44 j sans pluie duo (L) 4727

consommation solo trimestre 3549

consommation duo trimestre 9669

Estimation "grosse louche":
conso trimestre/precipitations moyenne min trimestre =
solo : 3588/125=29
duo: 9776/125=78

=> Il faut 29m2 pour satisfaire les besoins solo avec les hypothèses etape 1
=> Il faut 78m2 pour satisfaire les besoins duo avec les hypothèses etape 1

Les fortes précipitations sont généralement regroupées (grand écartype a la moyenne),
et on prendra par conséquent un réservoir minimum dimensionné à deux fois et demi
ce qui est nécessaire pour la précipitation journalière maximum.

Contraintes précipitations importantes:
Il faut a minima un reservoir de 3987L en solo (2.5*précipitations journaliere max*29)
et 10725L en duo (2.5*précipitations max *78)

Mais il faut également un minimum pour les périodes de sécheresses:
1735L en solo (44j consécutifs max sans pluie) de réserve avec les hypothèses etape 1
4727L en duo (44j consécutifs max sans pluie) de réserve avec les hypothèses etape 1
ce qui est satisfait avec la contrainte précédente.

On va maintenant utiliser ces résultats de surface minimum et de stockage minimum
comme hypothèse de base et ajouter un "data-test" avec des itérations (50% de la surface min
et 100% du volume min comme points de depart de l'iteration) sur la surface
de récupération et le volume de stockage pour vérifier qu'on n'a pas de tarissement
de notre stockage (on fait l'hypothèse qu'il y a une gestion du trop plein et qu'on n'a donc
pas de problèmes de réservoir qui déborde) et qu'on satisfait les besoins de consommation.

Étape 5 - Optimiser le stockage et prendre en compte une utilisation supplémentaire d'eau en periode estivale

Le bout de code python amélioré est le suivant (les commentaires expliquent chaque etape).
Pour expliquer un peu l'étape de l'itération : On démarre aux volume0 de capacité et surface0 de surface précalculées dans l'étape précédente. On fait des boucles d'itérations sur les données de précipitations et chaque jour on fait la soustraction eau recupérée-consommation journalière+consommation estivale si en periode estivale En cas de tarissement: on a une première boucle d'iteration 6 fois de +33 % de la surface, et pour chaque itération de surface, on a une deuxième boucle d'iteration 40 fois de +50 % du volume. On s'arrete à chaque boucle d'iteration dès que le dimensionnement convient et on enregistre le resultat. On affiche les résultats à la fin des itérations.

```
import math
import os
```

```
import pandas as pd
import time
# Attention si vous utilisez ce bout de code dans d'autres pays que la france, il faut ajouter
# les stations météos adhoc

# Processing des data
print("\nprocessing des data\n")
files=os.listdir('.')
csv=[a for a in files if a[-3:]=='csv']
combined_df = pd.concat((pd.read_csv(f,sep=';') for f in csv), ignore_index=True)
#07510 bordeaux
#07535 gourdon

#stations météos "hard coded"
stations=[{'ID': '07005', 'Nom': 'ABBEVILLE', 'Latitude': '50.136000', 'Longitude': '1.834000', 'Altitude': '69'}, {'ID': '07015', 'Nom': 'LILLE-LESQUIN', 'Latitude': '50.570000', 'Longitude': '3.097500', 'Altitude': '47'}, {'ID': '07020', 'Nom': 'PTE DE LA HAGUE', 'Latitude': '49.725167', 'Longitude': '-1.939833', 'Altitude': '6'}, {'ID': '07027', 'Nom': 'CAEN-CARPIQUET', 'Latitude': '49.180000', 'Longitude': '-0.456167', 'Altitude': '67'}, {'ID': '07037', 'Nom': 'ROUEN-BOOS', 'Latitude': '49.383000', 'Longitude': '1.181667', 'Altitude': '151'}, {'ID': '07072', 'Nom': 'REIMS-PRUNAY', 'Latitude': '49.209667', 'Longitude': '4.155333', 'Altitude': '95'}, {'ID': '07110', 'Nom': 'BREST-GUIPAVAS', 'Latitude': '48.444167', 'Longitude': '-4.412000', 'Altitude': '94'}, {'ID': '07117', 'Nom': "PLOUMANAC'H", 'Latitude': '48.825833', 'Longitude': '-3.473167', 'Altitude': '55'}, {'ID': '07130', 'Nom': 'RENNES-ST JACQUES', 'Latitude': '48.068833', 'Longitude': '-1.734000', 'Altitude': '36'}, {'ID': '07139', 'Nom': 'ALENCON', 'Latitude': '48.445500', 'Longitude': '0.110167', 'Altitude': '143'}, {'ID': '07149', 'Nom': 'ORLY', 'Latitude': '48.716833', 'Longitude': '2.384333', 'Altitude': '89'}, {'ID': '07168', 'Nom': 'TROYES-BARBEREY', 'Latitude': '48.324667', 'Longitude': '4.020000', 'Altitude': '112'}, {'ID': '07181', 'Nom': 'NANCY-OCHEY', 'Latitude': '48.581000', 'Longitude': '5.959833', 'Altitude': '336'}, {'ID': '07190', 'Nom': 'STRASBOURG-ENTZHEIM', 'Latitude': '48.549500', 'Longitude': '7.640333', 'Altitude': '150'}, {'ID': '07207', 'Nom': 'BELLE ILE-LE TALUT', 'Latitude': '47.294333', 'Longitude': '-3.218333', 'Altitude': '34'}, {'ID': '07222', 'Nom': 'NANTES-BOUGUENAIS', 'Latitude': '47.150000', 'Longitude': '-1.608833', 'Altitude': '26'}, {'ID': '07240', 'Nom': 'TOURS', 'Latitude': '47.444500', 'Longitude': '0.727333', 'Altitude': '108'}, {'ID': '07255', 'Nom': 'BOURGES', 'Latitude': '47.059167', 'Longitude': '2.359833', 'Altitude': '161'}, {'ID': '07280', 'Nom': 'DIJON-LONGVIC', 'Latitude': '47.267833', 'Longitude': '5.088333', 'Altitude': '219'}, {'ID': '07299', 'Nom': 'BALE-MULHOUSE', 'Latitude': '47.614333', 'Longitude': '7.510000', 'Altitude': '263'}, {'ID': '07314', 'Nom': 'PTE DE CHASSIRON', 'Latitude': '46.046833', 'Longitude': '-1.411500', 'Altitude': '11'}, {'ID': '07335', 'Nom': 'POITIERS-BIARD', 'Latitude': '46.593833', 'Longitude': '0.314333', 'Altitude': '123'}, {'ID': '07434', 'Nom': 'LIMOGES-BELLEGARDE', 'Latitude': '45.861167', 'Longitude': '1.175000', 'Altitude': '402'}, {'ID': '07460', 'Nom': 'CLERMONT-FD', 'Latitude': '45.786833', 'Longitude': '3.149333', 'Altitude': '331'}, {'ID': '07471', 'Nom': 'LE PUY-LOUDES', 'Latitude': '45.074500', 'Longitude': '3.764000', 'Altitude': '833'}, {'ID': '07481', 'Nom': 'LYON-ST EXUPERY', 'Latitude': '45.726500', 'Longitude': '5.077833', 'Altitude': '235'}, {'ID': '07510', 'Nom': 'BORDEAUX-MERIGNAC', 'Latitude': '44.830667', 'Longitude': '-0.691333', 'Altitude': '47'}, {'ID': '07535', 'Nom': 'GOURDON', 'Latitude': '44.745000', 'Longitude': '1.396667', 'Altitude': '260'}, {'ID': '07558', 'Nom': 'MILLAU', 'Latitude': '44.118500', 'Longitude': '3.019500', 'Altitude': '712'}, {'ID': '07577', 'Nom': 'MONTELIMAR', 'Latitude': '44.581167', 'Longitude': '4.733000', 'Altitude': '73'}, {'ID': '07591', 'Nom': 'EMBRUN', 'Latitude': '44.565667', 'Longitude': '6.502333', 'Altitude': '871'}, {'ID': '07607', 'Nom': 'MONT-DE-MARSAN', 'Latitude': '43.909833', 'Longitude': '-0.500167', 'Altitude': '59'}, {'ID': '07621', 'Nom': 'TARBES-OSSUN', 'Latitude': '43.188000', 'Longitude': '0.000000', 'Altitude': '360'}, {'ID': '07627', 'Nom': 'ST GIRONS', 'Latitude': '43.005333', 'Longitude': '1.106833', 'Altitude': '414'}, {'ID': '07630', 'Nom': 'TOULOUSE-BLAGNAC', 'Latitude': '43.621000', 'Longitude': '1.378833', 'Altitude': '151'}, {'ID': '07643', 'Nom': 'MONTPELLIER', 'Latitude': '43.577000', 'Longitude': '3.963167', 'Altitude': '2'}, {'ID': '07650', 'Nom': 'MARIGNANE', 'Latitude': '43.437667', 'Longitude': '5.216000', 'Altitude': '9'}, {'ID': '07661', 'Nom': 'CAP CEPET', 'Latitude': '43.079333', 'Longitude': '5.940833', 'Altitude': '115'}, {'ID': '07690', 'Nom': 'NICE', 'Latitude': '43.648833', 'Longitude': '7.209000', 'Altitude': '2'}, {'ID': '07747', 'Nom': 'PERPIGNAN', 'Latitude': '42.737167', 'Longitude': '2.872833', 'Altitude': '42'}, {'ID': '07761', 'Nom': 'AJACCIO', 'Latitude': '41.918000', 'Longitude': '8.792667', 'Altitude': '5'}, {'ID': '07790', 'Nom': 'BASTIA', 'Latitude': '42.540667', 'Longitude': '9.485167', 'Altitude': '10'}, {'ID': '61968', 'Nom': 'GLORIEUSES', 'Latitude': '-11.582667', 'Longitude': '47.289667', 'Altitude': '3'}, {'ID': '61970', 'Nom': 'JUAN DE NOVA', 'Latitude': '-17.054667', 'Longitude': '42.712000', 'Altitude': '9'}, {'ID': '61972', 'Nom': 'EUROPA', 'Latitude': '-22.344167', 'Longitude': '40.340667', 'Altitude': '6'}, {'ID': '61976', 'Nom': 'TROMELIN', 'Latitude': '-15.887667', 'Longitude': '54.520667', 'Altitude': '7'}, {'ID': '61980', 'Nom': 'GILLOT-AEROPORT', 'Latitude': '-20.892500', 'Longitude': '55.528667', 'Altitude': '8'}, {'ID': '61996', 'Nom': 'NOUVELLE AMSTERDAM', 'Latitude': '-37.795167', 'Longitude': '77.569167', 'Altitude': '27'}, {'ID': '61997', 'Nom': 'CROZET', 'Latitude': '-46.432500', 'Longitude': '51.856667', 'Altitude': '146'}, {'ID': '61998', 'Nom': 'KERGUELEN', 'Latitude': '-49.352333', 'Longitude': '70.243333', 'Altitude': '29'}, {'ID': '67005', 'Nom': 'PAMANDZI', 'Latitude': '-12.805500', 'Longitude': '45.282833', 'Altitude': '7'}, {'ID': '71805', 'Nom': 'ST-PIERRE', 'Latitude': '46.766333', 'Longitude': '-56.179167', 'Altitude': '21'}, {'ID': '78890', 'Nom': 'LA DESIRADE METEO', 'Latitude': '16.335000', 'Longitude': '-61.004000', 'Altitude': '27'}, {'ID': '78894', 'Nom': 'ST-BARTHELEMY METEO', 'Latitude': '17.901500', 'Longitude': '-62.852167', 'Altitude': '44'}, {'ID': '78897', 'Nom': 'LE RAIZET AERO', 'Latitude': '16.264000', 'Longitude': '-61.516333', 'Altitude': '11'}, {'ID': '78922', 'Nom': 'TRINITE-CARAVEL', 'Latitude': '14.774500', 'Longitude': '-60.875333', 'Altitude': '26'}, {'ID': '78925', 'Nom': 'LAMENTIN-AERO', 'Latitude': '14.595333', 'Longitude': '-60.995667', 'Altitude': '3'}, {'ID': '81401', 'Nom': 'SAINT LAURENT', 'Latitude': '5.485500', 'Longitude': '-54.031667', 'Altitude': '5'}, {'ID': '81405', 'Nom': 'CAYENNE-MATOURY', 'Latitude': '4.822333', 'Longitude': '-52.365333', 'Altitude': '4'}, {'ID': '81408', 'Nom': 'SAINT GEORGES', 'Latitude': '3.890667', 'Longitude': '-51.804667', 'Altitude': '6'}, {'ID': '81415', 'Nom': 'MARIPASOULA', 'Latitude': '3.640167', 'Longitude': '-54.028333', 'Altitude': '106'}, {'ID': '89642', 'Nom': "DUMONT D'URVILLE", 'Latitude': '-66.663167', 'Longitude': '140.001000', 'Altitude': '43'}]


def distance(lat1, lon1, lat2, lon2):
    """
    Calcule la distance entre deux points géographiques en utilisant la formule de la distance euclidienne.
    """
    return math.sqrt((lat2 - lat1)**2 + (lon2 - lon1)**2)

def station_la_plus_proche(x, y, stations):
    """
    Trouve la station météo la plus proche en utilisant les coordonnées x et y (latitude et longitude).
    """
    distance_min = float('inf')
    station_proche = None

    for station in stations:
        lat_station = float(station['Latitude'])
        lon_station = float(station['Longitude'])
```

```
        d = distance(x, y, lat_station, lon_station)
        if d < distance_min:
            distance_min = d
            station_proche = station

    return station_proche

# Demander à l'utilisateur d'entrer la latitude et la longitude
x_input = input("Entrez la latitude de votre lieux: ")
y_input = input("Entrez la longitude de votre lieux: ")

# Remplacer les virgules par des points
x_input = float(x_input.replace(',', '.'))
y_input = float(y_input.replace(',', '.'))

# Utilisez les valeurs entrées par l'utilisateur comme variables x et y pour trouver la station météo la plus proche
station_proche = station_la_plus_proche(x_input, y_input, stations)
print("\nLa station météo la plus proche est:", station_proche['Nom'])

# Demander à l'utilisateur d'entrer sa consommation d'eau hebdomadaire
waterconsohebdo = input("Entrez la consommation d'eau hebdomadaire constante(L): ")

# Remplacer les virgules par des points
waterconsohebdo = float(waterconsohebdo.replace(',', '.'))

# Calcul consommation journaliere moyenne
waterconsojour = waterconsohebdo/7

# Demander à l'utilisateur d'entrer le mois de debut de la periode estivale
moisdebutete = input("Entrez le mois de début pour la consommation d'eau hebdomadaire supplémentaire en periode estivale (1,2,3,4,5,6,7,8,9,10,11,12
): ")
try:
    _=int(moisdebutete)
    moisdebutete=_
except Exception as err:
    moisdebutete=5
    print(f"\nerreur de type ou valeur utilisateur vide, poursuite avec utilisation de moisdebutete={moisdebutete}")

# Demander à l'utilisateur d'entrer le mois de fin de la periode estivale
moisfinete = input("Entrez le mois de fin pour la consommation d'eau hebdomadaire supplémentaire en periode estivale (1,2,3,4,5,6,7,8,9,10,11,12): ")
try:
    _=int(moisfinete)
    moisfinete=_
except Exception as err:
    moisfinete=9
    print(f"\nerreur de type ou valeur utilisateur vide, poursuite avec utilisation de moisfinete={moisfinete}")

# Demander à l'utilisateur d'entrer la consommation d'eau supplémentaire en periode estivale
waterconsohebdoete = input("Entrez la consommation d'eau hebdomadaire supplémentaire en periode estivale (L) - 0 L par defaut: ")
try:
    _=float(waterconsohebdoete.replace(',', '.')) # Remplacer les virgules par des points

    waterconsohebdoete=_
except Exception as err:
    waterconsohebdoete=0
    print(f"\nerreur de type ou valeur utilisateur vide, poursuite avec utilisation de waterconsohebdoete={waterconsohebdoete}L")

# Calcul consommation journaliere moyenne
waterconsojourete = waterconsohebdoete/7

result=combined_df[combined_df['numer_sta']==int(station_proche['ID'])]

# Convertir la colonne 'date_column' dans un format datetime et la mettre en index trié
result['datetime'] = pd.to_datetime(result['date'], format='%Y%m%d%H%M%S')
result.set_index('datetime', inplace=True)
result = result.sort_index()

# remplacer les données manquantes par 0
result['rr3']=result['rr3'].replace('mq','0')
result['rr3']=result['rr3'].astype('float')

# Ne garder que la colonne des précipitations des 3 dernieres heures
result=result['rr3']
```

```
# Calculer les sommes de précipitations par jour
resultday=result.resample('D').sum()
resultdaymonthindex=resultday.copy()
resultdaymonthindex.index=resultdaymonthindex.index.month
print("\nMoyenne par jour (mm):\n", resultday.mean())
print("Minimum par jour (mm):\n", resultday.min())
print("Maximum par jour  (mm):\n", resultday.max())


# Calculer les sommes de précipitations par semaine
resultweek=result.resample('W').sum()

# Calculer les sommes de précipitations par mois
resultmonth=result.resample('ME').sum()

# Calculer les sommes de précipitations par trimestre
resulttrim=result.resample('QE').sum()
resulttrim=resulttrim.rename_axis('trimestre')
print(resulttrim)

# Calculer les sommes de précipitations par an
resultyear=result.resample('YE').sum()
print("\nPrécipitations annuelles moyennes (mm):\n",resultyear.mean())

# Calculer le nombre de jours consécutifs maximum sans pluie
max_streak = 0
current_streak = 0
for value in resultday:
    if value == 0:
        current_streak += 1
        max_streak = max(max_streak, current_streak)
    else:
        current_streak = 0  # Reset the streak if the value is not zero
print(f"\nNombre de jours consecutifs maximum sans pluie: {max_streak}")

# Moyenne par trimestre pour chaque trimestre
moyenne_trimestrielle_par_trimestre = resulttrim.groupby(resulttrim.index.quarter).mean()

# Minimum par trimestre pour chaque trimestre
min_trimestrielle_par_trimestre = resulttrim.groupby(resulttrim.index.quarter).min()

# Maximum par trimestre pour chaque trimestre
max_trimestrielle_par_trimestre = resulttrim.groupby(resulttrim.index.quarter).max()

# Imprimer les résultats
print("\nMoyenne par trimestre pour chaque trimestre (mm):\n", moyenne_trimestrielle_par_trimestre)
print("\nMinimum par trimestre pour chaque trimestre (mm):\n", min_trimestrielle_par_trimestre)
print("\nMaximum par trimestre pour chaque trimestre (mm):\n", max_trimestrielle_par_trimestre)


#Prise en comptes changement climatiques (hypothèses conservatrices multimodeles drias precipitations):
# Definir les impacts sur les volumes de précipitations par saison
adjustments = {0: -15, 1: -10, 2: -50, 3: -15}  # Adjust line 1 by -15%, line 2 by -10%, line 3 by -50%, line 4 by -15%

# Appliquer les impacts sur les précipitations moyennes par trimestres:
cc_moyenne_trimestrielle_par_trimestre=moyenne_trimestrielle_par_trimestre.copy()
for line, adjustment in adjustments.items():
    cc_moyenne_trimestrielle_par_trimestre.iloc[line] = cc_moyenne_trimestrielle_par_trimestre.iloc[line]+cc_moyenne_trimestrielle_par_trimestre.iloc[line]
*adjustment/100

print("\nMoyenne par trimestre pour chaque trimestre avec prise en compte du changement climatique (mm):\n", cc_moyenne_trimestrielle_par_trimestre)


# Minimum par jour pour chaque trimestre
min_par_jour_par_trimestre = resultday.groupby(resultday.index.quarter).min()
min_par_jour_par_trimestre=min_par_jour_par_trimestre.rename_axis('trimestre')

# Maximum par jour pour chaque trimestre
max_par_jour_par_trimestre = resultday.groupby(resultday.index.quarter).max()
max_par_jour_par_trimestre=max_par_jour_par_trimestre.rename_axis('trimestre')

# Moyenne par jour pour chaque trimestre
moyenne_par_jour_par_trimestre = resultday.groupby(resultday.index.quarter).mean()
moyenne_par_jour_par_trimestre=moyenne_par_jour_par_trimestre.rename_axis('trimestre')
```

```
moyenne_par_jour_par_trimestre = moyenne_par_jour_par_trimestre.rename_axis('trimestre')

# Imprimer les résultats
print("\nMinimum par jour pour chaque trimestre (mm):\n", min_par_jour_par_trimestre)
print("\nMaximum par jour pour chaque trimestre (mm):\n", max_par_jour_par_trimestre)
print("\nMoyenne par jour pour chaque trimestre (mm):\n", moyenne_par_jour_par_trimestre)


#Calcul seuil mini surface de recuperation:
surf0=(1/2)*math.ceil((13*waterconsohebdo)/min(moyenne_trimestrielle_par_trimestre))
print(f"""Seuil surface recuperation avec hypothèse entrée et données fournies par l'utilisateur (m2)
hypothese:(conso trimestre / precipitations moyenne min trimestre)
{int(math.ceil(surf0))} m2""")

#Calcul seuil mini reservoir:
contraintejourmax=(resultday.max())*surf0
contraintejourszero=max_streak*waterconsojour
volume0=math.ceil(max(2.5*contraintejourmax,contraintejourszero))
print(f"""\nSeuil volume avec hypothèse entrée et données fournies par l'utilisateur (L)
hypothèse: max((2.5*précipitations journaliere maxi*Seuil surface recuperation),(44j consécutifs max sans pluie*conso journaliere))
{int(math.ceil(volume0))} L""")

surf0_input = input("\n\nSi vous souhaitez corriger la valeur initiale de surface (m2) pour les itérations, entrer votre valeur, sinon appuyer sur entree")
try:
    _=float(surf0_input)
    surf0=_
except Exception as err:
    print(f"\nerreur de type ou valeur utilisateur vide, poursuite avec utilisation de surf0={surf0}m2")


volume0_input = input("\n\nSi vous souhaitez corriger la valeur initiale de volume (L) pour les itérations, entrer votre valeur, sinon appuyer sur entree")
try:
    _=float(volume0_input)
    volume0=_
except Exception as err:
    print(f"\nerreur de type ou valeur utilisateur vide, poursuite avec utilisation de volume0={volume0}L")

# Itérations algorithmiques stockage&consommation

#hypothèse récupérateur 2/3 plein à t0
water=(2/3)*volume0
resultsurfvolume=(volume0,surf0)
#boucle iteration
listsurf0=[surf0*(1+i*0.33) for i in range(0,999)]
listvolume0=[volume0*(1+i*0.5) for i in range(0,999)]
listeday=list(resultday)
listemonth=list(resultdaymonthindex.index)
listresult=[]
#fonction check surface volume
def iterv(data, v0,s0):
    "fonction check surface volume"
    water=(2/3)*v0
    for k in range(0,len(data)):
        recupday=data[k]*s0
        #print(f'recupday:{recupday}')
        if listemonth[k] in range(moisdebutete, moisfinete + 1):
            consoday = waterconsojour + waterconsojourete
        else:
            consoday = waterconsojour
        water=water+recupday-consoday
        #print(f'water:{water}')
        if water>v0:
            print("récupérateur plein")
            water=v0 #hypothese gestion du trop plein ok
            continue
        if water<0:
            print("récupérateur vide")
            #time.sleep(1)
            return (0,0)
    print("les surfaces et volumes permettent de subvenir à la consommation d'eau sur le dataset")
    return (v0,s0)

for i in range(0,6): #boucle iteration surface
    for k in range(0,len(listeday)):
        recupday=listeday[k]*listsurf0[i]
```

```
        recupday=listeday[k]*listsurf0[i]
        print(f'recupday:{recupday}')
        if listemonth[k] in range(moisdebutete,moisfinete+1):
            consoday=waterconsojour+waterconsojourete
        else:
            consoday = waterconsojour
        water=water+recupday-consoday
        print(f'water:{water}')
        if water>volume0:
            print("récupérateur plein")
            water=volume0 #hypothese gestion du trop plein ok
            continue
        if water<0:
            print("récupérateur vide, iteration avec hypothèse volume de récupération plus grand")
            #time.sleep(1)
            for j in range (1,i+40):
                resultsurfvolume=iterv(listeday,listvolume0[j],listsurf0[i])
                if resultsurfvolume!=(0,0):
                    listresult.append(resultsurfvolume)
                    break
                else:
                    continue
            break

for k in listresult:
    print(f"""avec les données fournies par l'utilisateur, et
un volume de {int(k[0])}L et
une surface de {int(k[1])}m2,
on satisfait aux besoins utilisateurs ({waterconsohebdo}L/semaine constant)
et {waterconsohebdoete}L/semaine en periode estivale (du mois {moisdebutete} au mois {moisfinete})
entrées en hypothèse\n""")
```

On fait le test avec les hypothèses ci dessus (latitude 44.2, longitude 0.6, solo 276L semaine, duo 752L semaine)
On obtient les résultats suivants:
solo:

avec les données fournies par l'utilisateur, et
un volume de 7976L et
une surface de 19m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 5982L et
une surface de 24m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 4985L et
une surface de 28m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 3988L et
une surface de 33m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 3988L et
une surface de 38m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

solo avec 600L/semaine en periode estivale:

avec les données fournies par l'utilisateur, et
un volume de 15642L et
une surface de 45m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 15642L et
une surface de 60m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 12514L et
une surface de 75m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 12514L et
une surface de 90m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 600.0L/semaine en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 12514L et
une surface de 105m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 12514L et
une surface de 120m2,
on satisfait aux besoins utilisateurs (276.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

duo:

avec les données fournies par l'utilisateur, et
un volume de 24133L et
une surface de 51m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 16089L et
une surface de 64m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 13407L et
une surface de 77m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 10726L et
une surface de 90m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 10726L et
une surface de 103m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

duo avec 600L/semaine en periode estivale: :

avec les données fournies par l'utilisateur, et
un volume de 33687L et
une surface de 70m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 24062L et
une surface de 93m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 19250L et
une surface de 116m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 19250L et
une surface de 139m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

avec les données fournies par l'utilisateur, et
un volume de 19250L et
une surface de 162m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

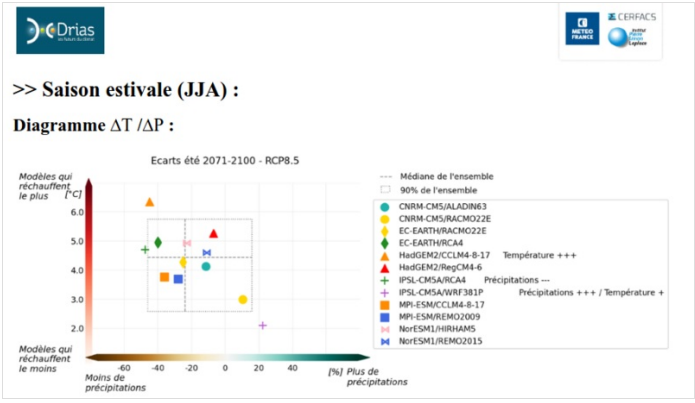
avec les données fournies par l'utilisateur, et
un volume de 19250L et
une surface de 185m2,
on satisfait aux besoins utilisateurs (752.0L/semaine constant)
et 600.0L/semaine supplémentaire en periode estivale (du mois 5 au mois 9)
entrées en hypothèse

Étape 6 - Utiliser des data du changement climatique

Comme très bien expliqué dans cet autre tuto : Estimer la quantité d'eau de pluie récupérable grâce à une toiture, dimensionner son stockage en prenant en compte les changements climatiques, le changement climatique va perturber les précipitations en termes de quantités mais surtout de fréquences.

Pour de la récupération/stockage, c'est particulièrement important notamment pour les périodes de sécheresses qui risques de s'accentuer.

Techniquement, on peut "data-tester" au jour le jour avec les données de prévisions du portail drias (https://drias-climat.fr/). Cependant, il faudrait d'une part data tester avec plusieurs set de données car les modèles sont tres différents les uns des autres, et d'autres part il s'agit de modèles climatiques et pas météorologiques, ce qui limite la pertinence d'utiliser des résultats comme input meteo.



En attendant que les scientifiques affinent leurs modèles pour la prospective à échelle plus fine (spatiale et temporelle), on peut reprendre les estimations d'impact sur les volumes de précipitations par saisons à partir des modèles drias 2070-2100 rcp 8.5 (+10% à -10% selon modèles au printemps, -50% à +20% en été,-15% à +5% en automne, -15% à +30% en hiver)

En étant prudent pour chaque saison, c'est à dire en prenant l'hypothèse la plus conservatrice pour chaque saison, on obtient des volumes réduits de :
-10% au printemps
-50% en été
-15% en automne
-15% en hiver
On peut alors mettre à jour l'algorithme :

Vous noterez qu'on a déjà ajouté les lignes suivantes dans le code de l'etape 5 juste avant le calcul de # Moyenne par trimestre pour chaque trimestre:

```
#Prise en comptes changement climatiques (hypothèses conservatrices
 multimodeles drias precipitations):
# Definir les impacts sur les volumes de précipitations par saison
adjustments = {0: -15, 1: -10, 2: -50, 3: -15}  # Adjust line 1 by -15%, lin
e 2 by -10%, line 3 by -50%, line 4 by -15%

# Appliquer les impacts sur les précipitations moyennes par trimestres:
cc_moyenne_trimestrielle_par_trimestre=moyenne_trimestrielle_par_tri
mestre.copy()
for line, adjustment in adjustments.items():
    cc_moyenne_trimestrielle_par_trimestre.iloc[line] = cc_moyenne_tri
mestrielle_par_trimestre.iloc[line]+cc_moyenne_trimestrielle_par_trime
stre.iloc[line]*adjustment/100

print("\nMoyenne par trimestre pour chaque trimestre avec prise en co
mpte du changement climatique (mm):\n", cc_moyenne_trimestrielle_p
ar_trimestre)
```

Ce qui affiche:

```
Moyenne par trimestre pour chaque trimestre avec prise en compte du
changement climatique (mm):
 trimestre
1    165.719091
2    202.630909
3     62.836364
4    164.158182
Name: rr3, dtype: float64
```

On n'effectue pas le data-test au jour le jour compte tenu des biais trop importants induits par le choix d'un seul modèle de prévision. En première approximation l'impact sur les résultats est de doubler la surface de récupération nécessaire à partir de la méthode décrite dans les étapes précédentes.