


climatLyon-1

May 11, 2019



Cahier d'exercices pour l'enseignement et l'apprentissage de programmation issu de la collection "Climat et météo tremplin pour l'enseignement des sciences" (PIA IFÉ ENS de Lyon - Météofrance ENM Toulouse). Le dispositif clef en main repose sur l'utilisation d'une RaspberryPi chargée avec le système d'exploitation Debian enrichi, fourni par le projet. Les sources et les exécutable sont accessibles dans [l'espace collaboratif de la forge github](#); plus d'information sur les [blogs d'accompagnement](#) systèmes d'exploitation sur [la page des OS de Raspberries Pi](#). Toutes les ressources issues du projet sont fournies sous licence [Creative Commons](#) ou sous les licences libres d'origine des outils utilisés. Les ressources du projet peuvent être utilisées dans tout autre environnement compatible. 

Auteur : G. Vidal

1 Une approche du changement climatique à Lyon

Ce cahier d'exercices propose plusieurs voies d'exploration d'un jeu de données issues des simulations climatiques de Météofrance. Le lot utilisé est issu d'une modélisation RCP 2.6. Le travail a été effectué avec la température maximale mais il est directement transposable à toutes les variables du modèle. Cette première partie traite de la méthode pour qu'un enseignant puisse extraire les données dont il a besoin d'un fichier commandé sur le site [DRIAS](#). Ce cahier manipule des données multidimensionnelles et doit être réservé à des étudiants avancés si on souhaite l'utiliser en classe. Le dessin des figures a été désactivé dans le dépôt sur la forge pour ne pas alourdir inutilement le fichier transféré.

1.1 Préparation de l'environnement et ouverture du fichier de données

Importer d'abord le module `netcdf4` et `numpy`, attention les majuscules sont impératives pour le nom `netCDF4`. Ces deux modules permettent de traiter les fichiers multidimensionnels au format `netCDF` utilisés dans le monde de la météorologie et de l'océanographie principalement.

```
In [1]: import netCDF4 as nc
import numpy as np
from datetime import datetime
from array import array
import sys, datetime, os
```

Importation des données de températures maximales depuis le fichier obtenu auprès du site [DRIAS](#) pour la région lyonnaise et intégration dans un fichier pour le traitement, puis affichage de la description du contenu, de la liste des variables. L'exemple choisi ici a été réalisé avec une grille de 10 x 10 noeuds centrés sur la ville de Lyon, pour obtenir un jeu de données se reporter au manuel numérique réalisé par E. Le Jan et CArole Larose dans le cadre du projet "Climat et Météo Tremplin pour l'enseignement des sciences". Les deux affichages proposés permettent de vérifier les propriétés du fichier obtenu ainsi que les variables qui pourront être utilisées. Ces affichages sont facultatifs et peuvent être commentés sans conséquence pour la suite.

```
In [2]: tMaxLyon = nc.Dataset('/home/vidal/TremplinDesSciences/2019/ClimatLyon/tasmax_metro_CNRM_Aladin_rcp2.6_QT_RCP2.6_20060101-2100123
# print('Description des données issues du modèle : \n', tMaxLyon, '\n')
print('Variables disponibles :', tMaxLyon.variables.keys(), '\n') # get all variable names
```

```
Variables disponibles : OrderedDict([('i', 'j', 'lat', 'lon', 'tasmax', 'time', 'x', 'y'])
```

1.2 Liste des dimensions et des variables du système de données

À partir de la liste des variables obtenue ci-dessus on renomme les jeux de données de chacune des variables qui seront exploitées pour effectuer les calculs et contrôle de la taille des échantillons. Les affichages proposés permettent de contrôler que les paramètres présents sont effectivement ceux qui sont attendus.

```
In [3]: #for dim in tMaxLyon.dimensions.items():
#       print(dim[1])
lyon_temp = tMaxLyon.variables['tasmx'] # variable temperature
lyon_date = tMaxLyon.variables['time'] # variable temps
lyon_lat,lyon_lon = tMaxLyon.variables['lat'], tMaxLyon.variables['lon'] # latitude longitude
lyon_x,lyon_y = tMaxLyon.variables['x'], tMaxLyon.variables['y'] # coordonnées métriques
lyon_gridi,lyon_gridj = tMaxLyon.variables['i'], tMaxLyon.variables['j'] # coordonnées grille Aladin
#print ('Variables |t  Forme |t|t Taille |t type :  \n')
#for var in tMaxLyon.variables.keys() :
#    print (var, '|t|t', tMaxLyon.variables[var].dimensions, '|t|t',
#           tMaxLyon.variables[var].shape, '|t', tMaxLyon.variables[var].dtype)
#print (' \n Unités : \n', lyon_temp.units, '|t',
#       lyon_date.units, '|t',
#       lyon_lat.units, '|t',
#       lyon_lon.units, '|t',
#       lyon_x.units, '|t',
#       lyon_y.units, '|n')
#print(lyon_gridi[:])
```

1.3 Construction du jeu de données de sortie des moyennes en fonction de : année mois latitude longitude

Création du fichier de sortie au format netCDF où seront stockées les valeurs moyennes calculées pour chaque noeud

```
In [4]: try: os.remove('tasmxLyon-1.nc') # par sécurité efface le fichier portant ce nom ! attention aux pertes possibles
except OSError : pass
extractLyonTempYearMonth = nc.Dataset('tasmxLyon-1.nc',mode='w',format='NETCDF4')
```

Définition et affectation des variables où sont copiées les données conservées et où seront stockés les résultats des calculs. (La syntaxe du

fichier netCDF reste à vérifier). Les années seront calculées pendant le calcul principal, les affichages permettent de vérifier la validité des données utilisées.

```
In [5]: data = []
years = []
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',
          'Aug', 'Sep', 'Oct', 'Nov', 'Dec', 'total']
lenMonthA = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
lenMonthB = [31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
#startDate = date[0] + offset# 0 premier janvier; 90 -> 1er avril ; 212 1er aout
offset = 0
firstyear = int(nc.num2date(lyon_date[offset], lyon_date.units).strftime("%Y"))
lastyear = int(nc.num2date(lyon_date[-1], lyon_date.units).strftime("%Y"))
lenmonths = len(months)
lenyears = lastyear - firstyear + 1
lengridi = lyon_gridi.shape[0]
lengridj = lyon_gridj.shape[0]
lentime = len(lenMonthA) * lenyears
print ("DateTime de départ de l'étude : ", firstyear,
      "\nDateTime de fin de l'étude : ", lastyear,
      "\nDurée de l'étude : ", lenyears, 'ans')
extractLyonTempYearMonth.createDimension('i', lengridi)      # latitude axis
extractLyonTempYearMonth.createDimension('j', lengridj)      # longitude axis
extractLyonTempYearMonth.createDimension('month', lenmonths) # month axis
extractLyonTempYearMonth.createDimension('year', lenyears)   # year axis
# Define two variables with the same names as dimensions,
# a conventional way to define "coordinate variables".
i = extractLyonTempYearMonth.createVariable('i', np.int, ('i',))
i.long_name = 'cell index along first dimension'
i[:] = lyon_gridi[:]
j = extractLyonTempYearMonth.createVariable('j', np.int, ('j',))
j.long_name = 'cell index along second dimension'
j[:] = lyon_gridj[:]
lat = extractLyonTempYearMonth.createVariable('lat', np.float32, ('j', 'i'))
```

```

lat.units = 'degrees_north'
lat.long_name = 'latitude'
lat.standard_name = 'latitude'
lat._CoordinateAxisType = 'Lat'
lat[:, :] = lyon_lat[:, :]
lon = extractLyonTempYearMonth.createVariable('lon', np.float32, ('j', 'i',))
lon.units = 'degrees_east'
lon.long_name = 'longitude'
lon.standard_name = 'longitude'
lon._CoordinateAxisType = 'Lon'
lon[:, :] = lyon_lon[:, :]
month = extractLyonTempYearMonth.createVariable('month', np.uint, ('month',))
month.units = 'number'
month.long_name = 'month'
month.standard_name = 'month'
month[:] = np.arange(0,13)
year = extractLyonTempYearMonth.createVariable('year', np.uint, ('year',))
year.units = 'date'
year.long_name = 'year'
year.standard_name = 'year'
# Define a 3D variable to hold the data
#tempMY = extractLyonTempYearMonth.createVariable('tempMY', np.float64, ('year', 'month', 'j', 'i')) # note: unlimited dimension is l
temp = extractLyonTempYearMonth.createVariable('temp', np.float64, ('year', 'month', 'j', 'i'))
temp.units = '°C' # degrees Kelvin
temp.standard_name = 'air_temperature' # this is a CF standard name
extractLyonTempYearMonth.title = 'Extrait TSMAX par moyenne mensuelle'
extractLyonTempYearMonth.institution = 'ENS de Lyon'
extractLyonTempYearMonth.institute_id = 'IFE Institut Francais de l Education'
extractLyonTempYearMonth.project_id = 'Climat et meteo tremplin pour l enseignement des sciences'
extractLyonTempYearMonth.model_id = 'CNRM-ALADIN52'
extractLyonTempYearMonth.product = 'output'
extractLyonTempYearMonth.contact = 'gerard.vidal@ens-lyon.fr'
extractLyonTempYearMonth.creation_date = str(datetime.datetime.now())

```

```

extractLyonTempYearMonth.driving_experiment_name = 'DRIAS2014'
extractLyonTempYearMonth.experiment = 'RCP2.6'
extractLyonTempYearMonth.model = 'ALADIN-Climat'
extractLyonTempYearMonth.comment = "Extraction des moyennes de la région Lyonnaise de 2006 à 2100 \
                                   et quelques exemples d'affichage"

#print(gridi[:,gridj[:]])
#print(gridi.shape,gridj.shape)
#print(lat[2,:])

```

DateTime de départ de l'étude : 2006
 DateTime de fin de l'étude : 2100
 Durée de l'étude : 95 ans

1.4 Calcul principal des moyennes par mois pour chaque noeud et toutes les années

```

In [6]: iteri = offset
        iterj = 0
        iteriFirst = iteri
        while iteri < lyon_temp.shape[0] :
            for iterj in range(lenyears) :
                year[iterj] = (nc.num2date(lyon_date[iteri],lyon_date.units).strftime("%Y"))
            # print(nc.num2date(my_date[iteri],my_date.units), iteri, iterj) # vérification du point de départ
            if (iterj % 4 == 2) :
                for p in range(len(lenMonthB)) :
                    iteriLast = iteri+lenMonthB[p]
                    # moyenne du mois
                    temp[iterj,p,:,:] = np.mean(lyon_temp[iteri:iteriLast,:,:] - 273,axis=0)
                    iteri = iteriLast
            else :
                for p in range(len(lenMonthA)) :
                    iteriLast = iteri+lenMonthA[p]
                    # moyenne du mois
                    temp[iterj,p,:,:] = np.mean(lyon_temp[iteri:iteriLast,:,:] - 273,axis=0)

```

```

        iteri = iteriLast
        # moyenne de l'année
        temp[iterj, lenMonthA, :, :] = np.mean(lyon_temp[iteriFirst:iteriLast, :, :] - 273, axis=0)
        iteriFirst = iteriLast+1
    #print(iteri)
    #print('fin du traitement : ', nc.num2date(lyon_date[iteri-2], lyon_date.units))
    # vérification de la fin d'année
    #years = year[:].tolist()
    #print ('years : \n', years)
    #extractLyonTempYearMonth.close() # Ne pas oublier de fermer le fichier netCDF

```

Les affichages suivants permettent de vérifier que les données obtenues correspondent au format attendu

```

In [7]: #print('température moyenne année 2015 noeud 5 5 : \n', temp[9, :, 5, 5])
        #print("température moyenne mois d'avril sur les 95 années noeud 8 4 : \n", temp[:, 3, 8, 4])
        #print("température moyenne année 2050 mois d'aout tous les noeuds : \n", temp[45, 7, :, :])

```

1.4.1 Calculs pour une seule période de yearInterval années

Calcul de la moyenne de températures du mois de calcMonth sur yearInterval années à partir de l'année yearBegin sur les noeuds allant de (startj, starti) de taille (intervalj, intervali)

```

In [8]: # Années
        yearBegin = 2010
        yearInterval = 30
        calcMonth = 4
        # Grille i j
        starti = 5
        intervali = 4
        startj = 2
        intervalj = 4
        # Variables de calcul
        startYear = yearBegin - 2006
        endYear = startYear + yearInterval

```

```

endi = starti + intervali
endj = startj + intervalj
loc_i = 2
loc_j = 3
locLon = lon[loc_j,loc_i]
locLat = lat[loc_j,loc_i]
locMonth = months[calcMonth]

if not (startYear >= 0) and (endYear <= lenyears) :
    print('starting year or finishing year out of bounds')
    sys.exit('giving up on year bounds')
#print(gridj[loc_j])
#print(gridi[loc_i])
#print(locLon)
#print(locLat)
#print(locMonth)
#print(startYear, ': ', endYear, ', ', calcMonth, ', ', startj, ': ', endj, ', ', starti, ': ', endi)
#print(temp.shape)
#print(np.mean(temp[startYear:endYear, calcMonth, startj:endj, starti:endi]))

```

Une valeur pour un mois donné Calcul de la moyenne de températures du mois de calcMonth sur yearInterval années à partir de l'année yearBegin sur les noeuds allant de (startj, starti) de taille (intervalj, intervali)

```

In [9]: moyAreaInterval = np.mean(temp[startYear:endYear, calcMonth, startj:endj, starti:endi])
        print(moyAreaInterval)

```

19.8438470184803

Enregistrement du fichier 13 valeurs (une par mois et année) Calcul de la moyenne de températures annuelle et de tous les mois sur yearInterval années à partir de l'année yearBegin sur les noeuds allant de (startj, starti) de taille (intervalj, intervali). Les douze moyennes mensuelles sont suivies de la moyenne annuelle.


```
In [10]: slice = lenmonths -1 # slice = lenmonths pour traiter en plus la moyenne annuelle
moyAreaInterval = [None] * slice
moyAreaInterval[:] = np.mean(temp[startYear:endYear,:,startj:endj,starti:endi],axis=(0,2,3))
#print(moyAreaInterval[0:slice])
with open('moyMonth.txt', 'w') as file :
    for p in range(slice) :
        file.write(months[p])
        file.write(';')
        file.write(str(moyAreaInterval[p]))
        file.write('\n')
```

préparation de la création de figures On importe les bibliothèques plotly

```
In [11]: import plotly.offline as py
import plotly.graph_objs as go
from plotly import tools

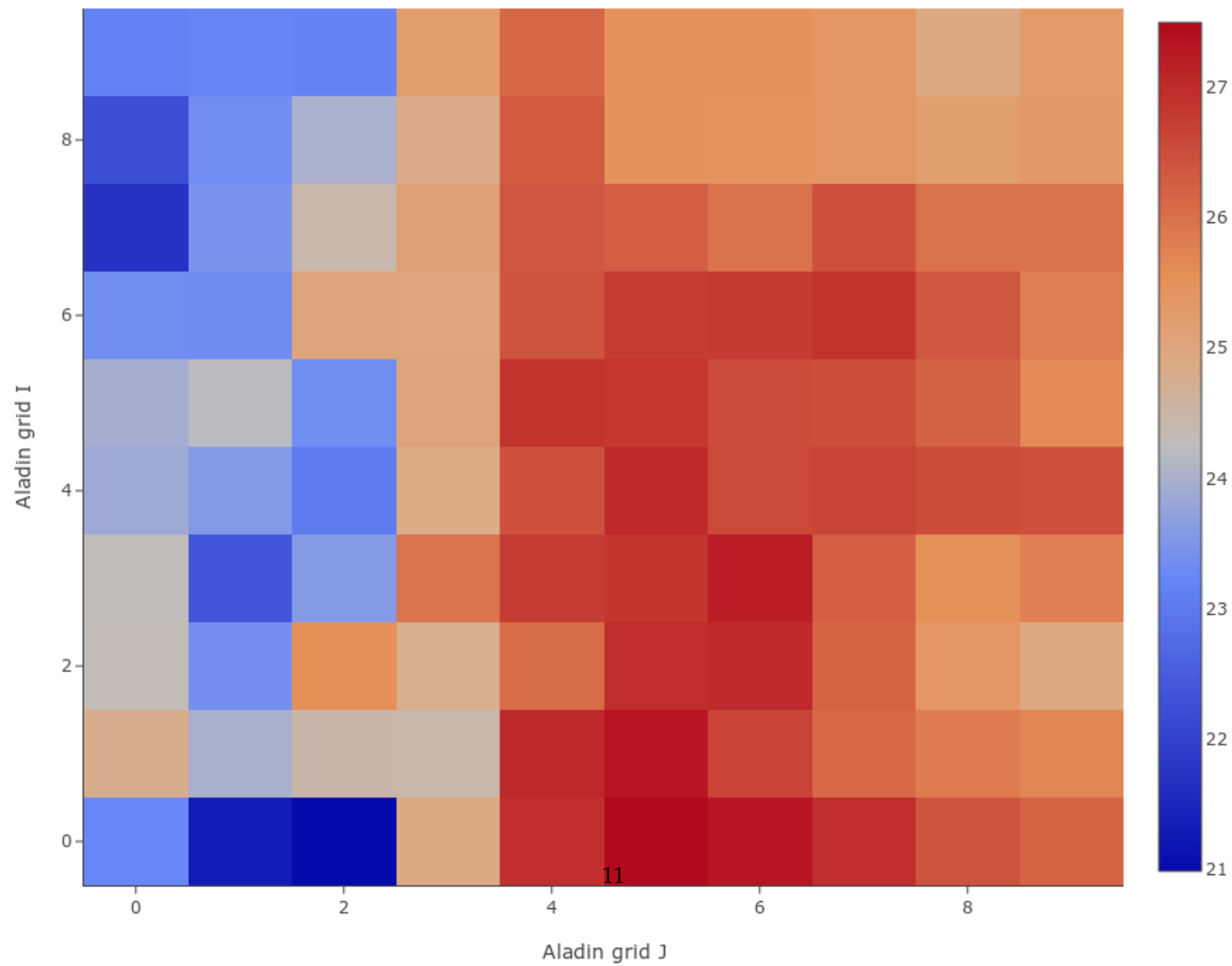
py.init_notebook_mode(connected=True)
```

Création d'une figure bidimensionnelle Valeurs moyennes de la température max sur toute la région lyonnaise au mois de juillet de l'horizon 2040

```
In [12]: trace0 = go.Heatmap(z=temp[0,6,:,:].tolist())
data=[trace0]
layout = dict (
    title = "Température moyenne maximale du mois de juillet horizon 2040",
    autosize=False,
    width=800,
    height=800,
    xaxis = dict(
        title = 'Aladin grid J',
        showline=True,
        showticklabels=True,
```

```
        ticklen=5
    ),
    yaxis = dict(
        title = 'Aladin grid I',
        showline=True,
        showticklabels=True
    ),
)
fig = dict(data=data, layout=layout)
py.ipplot(fig, filename='basic-heatmap')
```

Température moyenne maximale du mois de juillet horizon 2040



Comparaison de 3 horizons pour le mois de juillet en région lyonnaise

```
In [13]: trace0 = go.Heatmap(z=temp[0,6,:,:].tolist(), zauto=False, zmin=20,zmax=28)
         trace1 = go.Heatmap(z=temp[49,6,:,:].tolist(), zauto=False, zmin=20,zmax=28)
         trace2 = go.Heatmap(z=temp[89,6,:,:].tolist(), zauto=False, zmin=20,zmax=28)

         fig = tools.make_subplots(rows=1, cols=3, subplot_titles=('horizon 2030','horizon 2060','horizon 2090'))

         fig.append_trace(trace0, 1, 1)
         fig.append_trace(trace1, 1, 2)
         fig.append_trace(trace2, 1, 3)

         fig['layout']['xaxis1'].update(title='Aladin grid J')
         fig['layout']['xaxis2'].update(title='Aladin grid J')
         fig['layout']['xaxis3'].update(title='Aladin grid J')

         fig['layout']['yaxis1'].update(title='Aladin grid I')

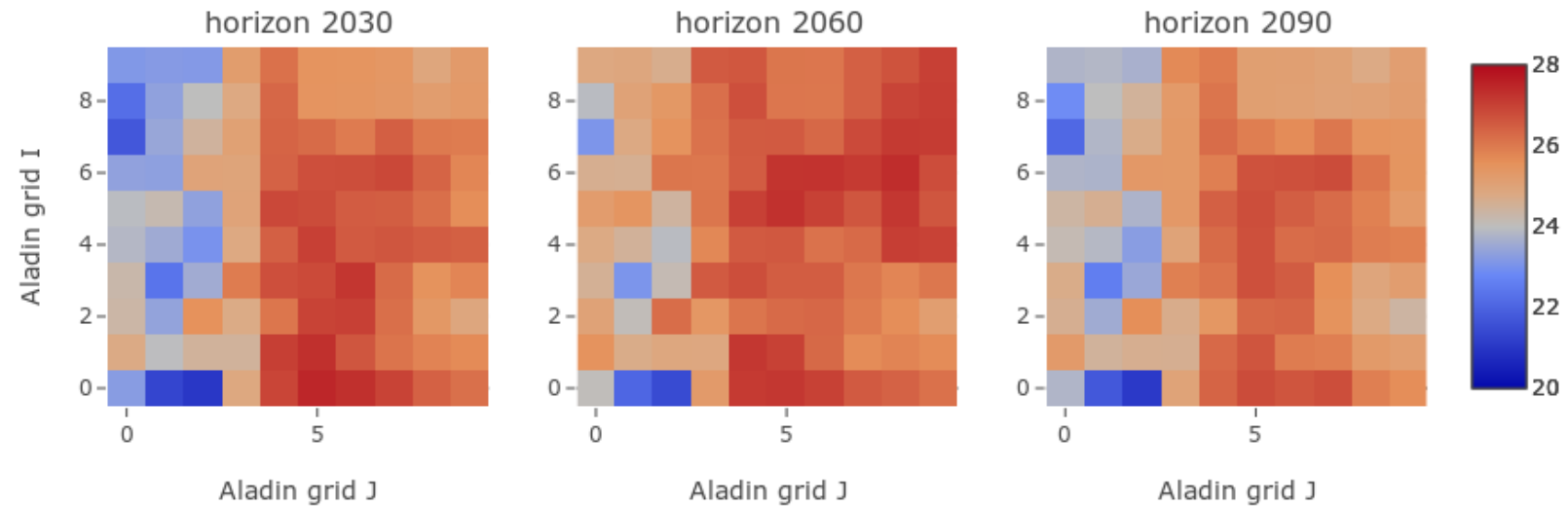
         fig['layout'].update(title='Température moyenne maximale du mois de juillet', autosize=False, width=800, height=380)

         py.iplot(fig, filename='basic-heatmap')
```

This is the format of your plot grid:

```
[ (1,1) x1,y1 ] [ (1,2) x2,y2 ] [ (1,3) x3,y3 ]
```

Température moyenne maximale du mois de juillet



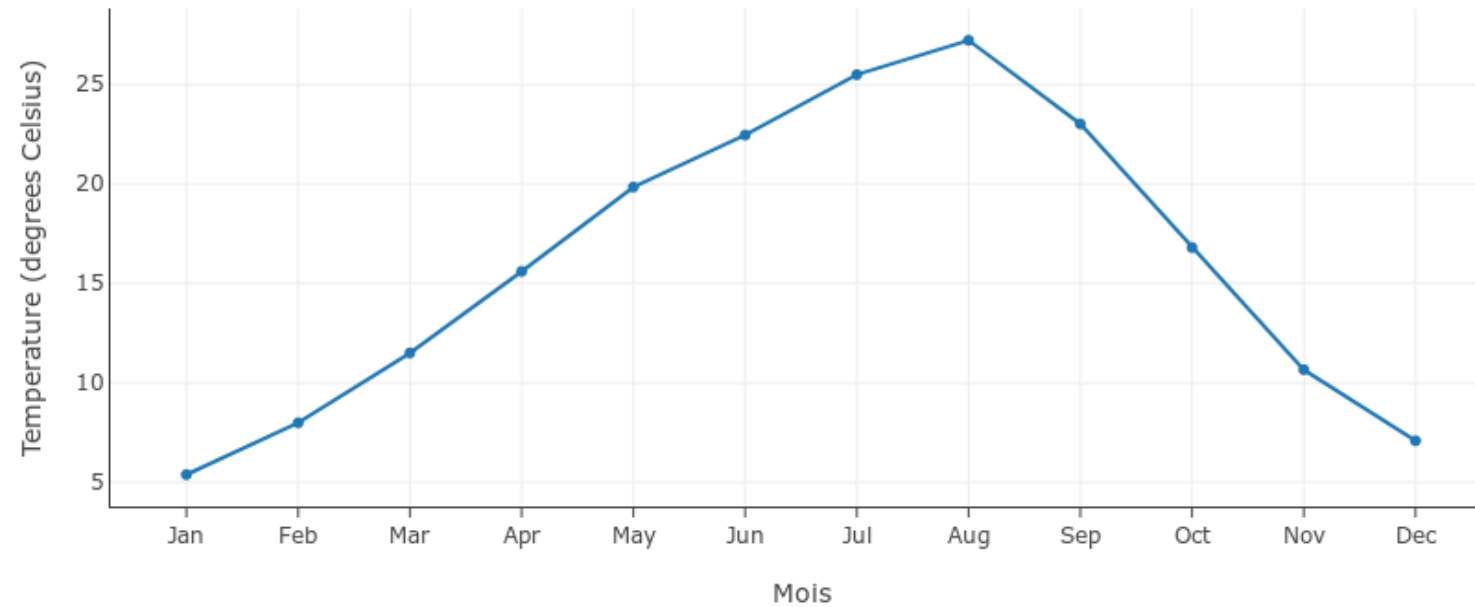
```
In [14]: trace0 = go.Scatter(
    x = months[0:slice],
    y = moyAreaInterval[0:slice],
    name = 'Avril 2040'
)

data = [trace0]

layout = dict (
    title = "Température moyenne maximale des mois de l'année horizon 2040",
```

```
axis = dict(  
    title = 'Mois',  
    showline=True,  
    showticklabels=True,  
    ticklen=5  
)  
yaxis = dict(  
    title = 'Temperature (degrees Celsius)',  
    showline=True,  
    showticklabels=True  
)  
)  
  
fig = dict(data=data, layout=layout)  
  
py.iplot(fig, filename='basic-line')
```

Température moyenne maximale des mois de l'année horizon 2040



n X n valeurs pour n² noeuds pour 12 mois et année Calcul de la moyenne de températures annuelle et de tous les mois sur yearInterval années à partir de l'année yearBegin sur chaque noeud allant de (startj,starti) de taille (intervalj,intervali). Les douze moyennes mensuelles sont suivies de la moyenne annuelle.

```
In [15]: moyAreaInterval = np.zeros((lenmonths,intervalj,intervali))
        moyAreaInterval[:] = np.mean(temp[startYear:endYear,:,startj:endj,starti:endi],axis=(0))
```

```

print(moyAreaInterval.shape)
#print(moyAreaInterval[:,0,0])
nbnoeuds = intervalj * intervali
data = [] * nbnoeuds
trace = [] * nbnoeuds

```

(13, 4, 4)

```

In [16]: for p in range(intervalj) :
        for q in range(intervali) :
            order = p * intervalj + q
            trace = go.Scatter(
                x = months[0:12],
                y = moyAreaInterval[:,p,q],
                name = 'Année 2040 noeud : (%s'%(str(p)+'+', '+str(q)+''))'
            )
            data.append(trace)

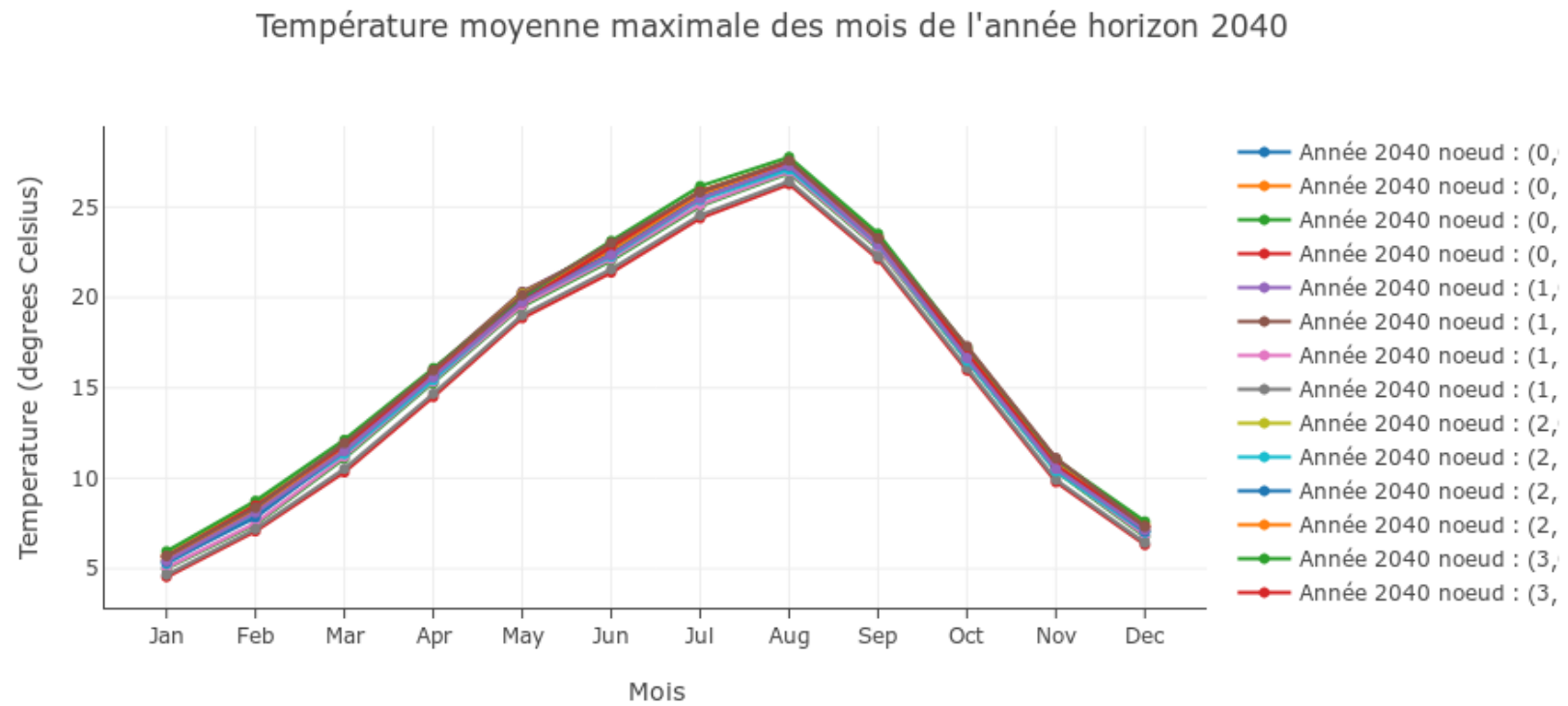
layout = dict (
    title = "Température moyenne maximale des mois de l'année horizon 2040",
    xaxis = dict(
        title = 'Mois',
        showline=True,
        showticklabels=True,
        ticklen=5
    ),
    yaxis = dict(
        title = 'Temperature (degrees Celsius)',
        showline=True,
        showticklabels=True
    ),
)

```



```
fig = dict(data=data, layout=layout)
```

```
py.iplot(fig, filename='basic-line')
```



1.4.2 Calculs pour UNE SÉRIE DE PÉRIODES de yearInterval années

Calcul de la moyenne de températures des mois de calcMonth (il peut y en avoir un seul ou un choix), sur yearInterval années à partir de l'année yearBegin jusqu'à l'année yearBegin + yearPeriod sur les noeuds à partir de (startj,starti) de taille (intervalj,intervali). La valeur trouvée est affectée à la dernière année de la plage de calcul.

```
In [17]: # ASéquence d'années pendant lesquelles les calculs sont effectués
```

```
yearBegin = 2010
yearInterval = 30
yearPeriod = 40
calcMonth = [4,5,6,7,8,9]
lencalcMonths = len(calcMonth)
# Grille i j
starti = 5
intervali = 4
startj = 2
intervalj = 4
# Variables de calcul
startYear = yearBegin - 2006
endYear = startYear + yearInterval
#print(yearPeriodInterval)
endi = starti + intervali
endj = startj + intervalj
loc_i = 2
loc_j = 3
locLon = lon[loc_j,loc_i]
locLat = lat[loc_j,loc_i]

if not (startYear >= 0) and (endYear <= lenyears) :
    print('starting year or finishing year out of bounds')
    sys.exit('giving up on year bounds')
#print(gridj[loc_j])
#print(gridi[loc_i])
```

```

#print(locLon)
#print(locLat)
#print(locMonth)
#print(startYear, ': ', endYear, ', ', calcMonth, ', ', startj, ': ', endj, ', ', starti, ': ', endi)
#print(temp.shape)
#print(np.mean(temp[startYear:endYear, calcMonth, startj:endj, starti:endi]))

```

1.4.3 Calcul pour n mois sur p périodes de q années pour la zone

Calcul de la moyenne des températures mensuelles sur yearInterval ans pendant une période de yearPeriod années successives pour une sélection de calcMonth mois.

```

In [18]: moyAreaInterval = np.zeros((lencalcMonths, yearPeriod))
        for p in range(yearPeriod) :
            a = startYear + p
            b = endYear + p
            moyAreaInterval[:,p] = np.mean(temp[a:b, calcMonth, startj:endj, starti:endi], axis=(0,2,3))
        #print(moyAreaInterval)
        firstPlotYear = startYear + yearInterval
        lastPlotYear = firstPlotYear + yearPeriod
        data = [] * lencalcMonths
        trace = [] * lencalcMonths

```

Le diagramme ci-dessous représente l'évolution sur yearPeriod années de la température dans la région Lyonnaise.

```

In [19]: for q in range(lencalcMonths) :
        trace = go.Scatter(
            x = year[firstPlotYear:lastPlotYear],
            y = moyAreaInterval[q,:],
            name = 'Moyenne du mois %s'%months[calcMonth[q]]
        )
        data.append(trace)

```

```

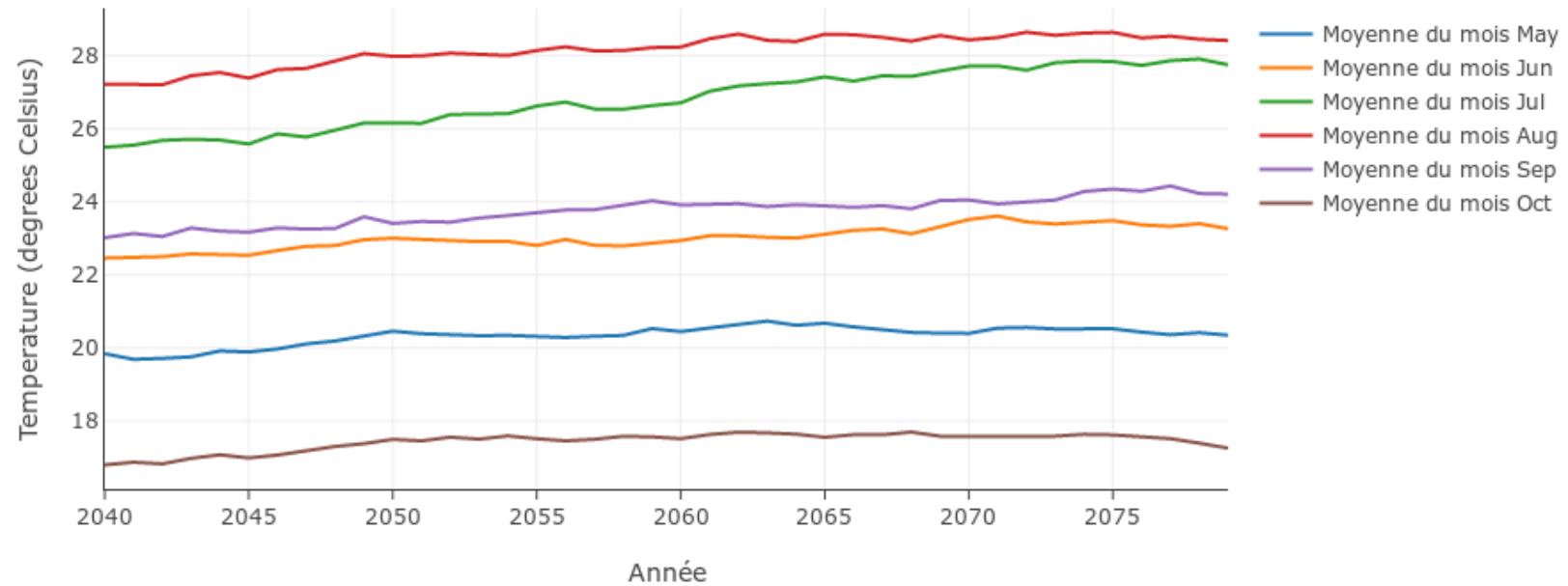
layout = dict (
    title = "Température moyenne maximale des mois de l'année moyennées sur 30 ans",
    xaxis = dict(
        title = 'Année',
        showline=True,
        showticklabels=True,
        ticklen=5
    ),
    yaxis = dict(
        title = 'Temperature (degrees Celsius)',
        showline=True,
        showticklabels=True
    ),
)

fig = dict(data=data, layout=layout)

py.ipplot(fig, filename='basic-line')

```

Température moyenne maximale des mois de l'année moyennées sur 30 ans



```
In [20]: #for dim in extractLyonTempYearMonth.dimensions.items():
#         print(dim[1])
#for var in extractLyonTempYearMonth.variables.keys() :
#         print (var, '\t\t', extractLyonTempYearMonth.variables[var].dimensions, '\t\t',
#               extractLyonTempYearMonth.variables[var].shape, '\t', extractLyonTempYearMonth.variables[var].dtype)
```

```
In [21]: extractLyonTempYearMonth.close()
```

```
In [ ]:
```