Séries Temporelles

http ://maths.cnam.fr/Membres/ghorbanzadeh/

1 Processus Aléatoires ou Stochastiques

Un processus aléatoire est une famille de variables aléatoires $(X_t)_{t\in T}$ définies sur le même espace de probabilité. On distingue généralement les processus en temps discret et en temps continu, à valeurs discrètes et à valeurs continues. Si l'ensemble T est dénombrable on parle de processus discret ou de série temporelle, si l'ensemble est indénombrable on parle de processus continu.

1.1 Statistiques descriptives d'un Processus Aléatoire

Soit $(X_t)_{t\in T}$ un processus alèatoire. On définit

1. l'espérance mathématique de X_t par :
$$\mu_X(t) = \mathbb{E}[X_t] \tag{1}$$
2. la Fonction Autocorrélation de X_t par :
$$R_X(t,k) = \mathbb{E}[X_t X_k] \tag{2}$$
3. la Fonction Autocovariance de X_t par :
$$C_X(t,k) = \mathbb{E}\big[\big(X_t - \mu_X(t)\big)\big(X_k - \mu_X(k)\big)\big] \tag{3}$$
4. la Fonction variance de X_t par :
$$\sigma_X^2(t) = Var[X_t] = C_X(t,t) \tag{4}$$
5. les coefficients de corrélation de X_t par :
$$\rho_X(t,k) = \frac{C_X(t,k)}{\sqrt{C_X(t,t)C_X(k,k)}} \tag{5}$$

1.2 Processus Aléatoires stationnaires au second ordre

Un processus alèatoire $(X_t)_{t\in T}$ est dit stationnaire au second ordre ou stationnaire au sens large, si les trois conditions suivantes sont satisfaites :

$$\forall t \in T \ , \ \mathbb{E}[X_t^2] < \infty \tag{6a}$$
$$\forall t \in T \ , \ \mathbb{E}[X_t] = m \text{ indpendant de temps } t \tag{6b}$$
$$\forall (t,k) \in T^2 \ , \ C_X(t,k) = \mathbb{E}\big[\big(X_t - m\big)\big(X_k - m\big)\big] \text{ne dépend que de ladifférence } : t-k \tag{6c}$$

Par (6c), pour tout Processus Aléatoire stationnaire au second ordre, on désigne sa fonction Autocovariance par γ, définie par :
$$\gamma(h) = \mathbb{E}\big[\big(X_{t+h} - m\big)\big(X_t - m\big)\big] \tag{7}$$

Remarque 1. La fonction Autocovariance d'un Processus Aléatoire stationnaire au second ordre est paire, en effet,

$$\gamma(h) = \mathbb{E}\big[\big(X_t - \mu_X(t)\big)\big(X_{t-h} - \mu_X(t-h)\big)\big] = \mathbb{E}\big[\big(X_{t-h} - \mu_X(t-h)\big)\big(X_t - \mu_X(t)\big)\big] = \gamma(-h)$$

Remarque 2. La condition (6c) implique :

$$Var[X_t] = C_X(t,t) = \gamma(0) \text{ indpendant de temps } t$$

1.3 Le processus Bruit Blanc

Parmi la classe des processus stationnaires, il existe des processus particuliers que sont les processus bruit blanc (en anglais white noise). Ces processus sont très souvent utilisés en analyse des séries temporels car ils constituent en quelque sorte les "briques élémentaires" de l'ensemble des processus temporels.

Un bruit blanc est un processus stationnaire à accroissements indépendants. On parle aussi de processus i :i :d : (variables indépendantes et identiquement distribuées).
D'abord définisson ce qu'est un processus à accroissements indépendants.

Un processus $(X_t)_{t\in T}$ est un processus à accroissements indépendants si pour tout n−uplet du temps $t_1 < t_2 < \ldots < t_n$,, les variables aléatoires $X_{t_1}, X_{t_2} - X_{t_1}, \ldots, X_{t_n} - X_{t_{n-1}}$ sont indépendantes.

Le processus $(X_t)_{t\in T}$ est un processus stationnaire à accroissements indépendants, si, pour tout h, la loi de probabilité des accroissements $X_{t+h} - X_t$ est indépendante de t. C'est donc une classe particulière des processus stationnaires.

Un processus $(\varepsilon_t)_{t\in T}$ st un Bruit Blanc s'il satisfait les deux conditions suivantes : $\forall t \in T$,

$$\begin{aligned} &\mathbb{E}[\varepsilon_t] = 0 \\ &\mathbb{E}[\varepsilon_t\, \varepsilon_{t-h}] = \begin{cases} \sigma^2 & \text{si } \ h = 0 \\ 0 & \text{sinon} \end{cases} \end{aligned} \tag{8}$$

En outre, on parle de Bruit Blanc gaussien lorsque la loi de probabilité du processus est une loi normale $\mathcal{N}(0, \sigma^2)$.

Pour la simulation de la loi normale $\mathcal{N}(\mu, \sigma^2)$ avec les différents modules **numpy** et **scipy** de python, on a le code suivant :

code 1 – loi normale

```
import numpy
import scipy
import scipy.stats

######   fonction randn de numpy  ######

#v.a. N(0,1)
numpy.random.randn()
# vecteur de dim=2 suivant la loi  N(0,1)
numpy.random.randn(2,1)
# matrice de 5-lignes, 2-colonnes suivant la loi  N(0,1)
numpy.random.randn(5,2)
#v.a. N(2,5)
2.0+numpy.sqrt(5)*numpy.random.randn()
# matrice de 3-lignes, 5-colonnes suivant la loi  N(-2.4,11)
-2.4+numpy.sqrt(11)*numpy.random.randn(3,5)

######   fonction numpy.random.normal   ######
#v.a. N(0,1)
```

```
numpy.random.normal()
#v.a. N(8,12)
numpy.random.normal(8,numpy.sqrt(12))
## matrice de 7-lignes, 3-colonnes suivant la loi N(4,6)
numpy.random.normal(4,numpy.sqrt(6),[7,3])

######    fonction scipy.stats.norm  : import scipy.stats  ######

#v.a. N(0,1)
scipy.stats.norm.rvs(loc=0, scale=1, size=1)
## matrice de 8-lignes, 4-colonnes suivant la loi N(-3,2)
scipy.stats.norm.rvs(loc=-3.0, scale=numpy.sqrt(2), size=[8,4])
```

Pour l'histogramme des données suivant une loi normale $\mathcal{N}(\mu, \sigma^2)$, on a le code suivant :

code 2 – histogramme des données suivant une loi normale

```
from matplotlib import pylab as plt
import numpy

#### fonction pour enlever les cardres haut , gauche droit de la figure
def Axe_simple(ax):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.spines['left'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.set_yticks([])

### simulation de la loi normale N(mu,sigma2)
mu=5
sigma2=7
taille =2000
data=numpy.random.normal(mu, numpy.sqrt(sigma2), size=taille)
#data : vecteur de taille=2000

### histogramme
nb_classe=15
min_x=numpy.min(data)
max_x=numpy.max(data)

#extremum_classe : extrémités des classes pour l'histogramme
extremum_classe=[min_x+(max_x-min_x)*i/(nb_classe) for i in range(nb_classe+1)]

fig, ax_h = plt.subplots(1, 1)
ax_h.hist(data,extremum_classe,facecolor=[0.50,0.50,0.80], edgecolor='white')

Axe_simple(ax_h)
plt.show()
```

Exemple 1.
Marches aléatoires. Soit $(U_n)_{n\geq 1}$ une suite de variables aléatoires indépendantes suivant la même loi : $\mathbb{P}(U_n=-1)=\mathbb{P}(U_n=1)=\frac{1}{2}$. On pose : $X_n=X_0+\sum_{i=1}^{n}U_i$.
$\mathbb{E}[X_n]=?$, $cov(X_n,X_m)=?$, $Var[X_n]=?$, X_n est-il stationnaire ? (**Démonstration en cours**).

Pour la simulation d'une marche aléatoire on a le code suivant :

code 3 – marches aléatoires

```
import random
import matplotlib.pylab as plt

def moinsUn_ou_Un():
    U=random.random()
    if U<= 0.5:
        return -1
    else:
        return 1

def marche(n,X_0=0):
    n1 = n if n > 0 else 1
    return X_0+sum([moinsUn_ou_Un() for _ in range(n1)])

##plt.figure(figsize=(10, 6), dpi=250)
plt.figure()
ax = plt.gca()

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.xaxis.set_ticks_position('bottom')

nb_trajectoire=5
nb_point=100
for _ in range(nb_trajectoire):
    ax.plot([n for n in range(1,nb_point+1)], [marche(n,X_0=3) for n in range
        (1,nb_point+1)])
ax.axhline(y=0,color='black',lw=1)
##plt.savefig('marche_alea.eps', format="eps", dpi=400)
##plt.savefig('marche_alea.png', format="png", dpi=400)
plt.show()
####plt.show() avant plt.close()
##plt.close()
```

Exemple 2.
Processus aléatoire à temps continu. Soit $(U_n)_{n\geq 1}$ une suite de variables aléatoires indépendantes suivant la même loi normale $\mathcal{N}(0,\sigma^2)$, pour $t \in [0,1]$, on définit le processus X_t par : $X_t = \sum_{i=1}^{[nt]} U_i$ où $[A]$ désigne la partie entière de A.
$\mathbb{E}[X_t] = ?$, $cov(X_t, X_s) = ?$, $Var[X_t] = ?$, X_t est-il stationnaire ? (**Démonstration en cours**).

Pour la simulation d'une marche aléatoire on a le code suivant :

code 4 – processus aléatoire à temps continu

```
from matplotlib import pylab as plt
import math
import numpy


def Processus_Continu(t,n=100,  sigma2=1):
        m=math.floor(n*t) # [nt]
        s=numpy.sqrt(sigma2)
        return sum([numpy.random.normal(0,s) for _ in range(m)])

##plt.figure(figsize=(10, 6), dpi=250)
plt.figure()
ax = plt.gca()

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.xaxis.set_ticks_position('bottom')

nb_trajectoire=[100, 1000, 1500, 5000, 6000]
nb_point=100
t=[0.01+(0.99-0.01)*i/(nb_point-1)  for i in range(nb_point)]
for j in range(len(nb_trajectoire)):
    ax.plot(t, [Processus_Continu(temps,n=nb_trajectoire[j],sigma2=1) for temps
        in t])
ax.axhline(y=0,color='black',lw=1)

ax.set_xlabel("t")
ax.set_ylabel("$X_{t}$")
ax.set_title("{:} trajectoires du processus".format(len(nb_trajectoire)))

##plt.savefig('Processus_alea.eps', format="eps", dpi=400)
##plt.savefig('Processus_alea.png', format="png", dpi=400)
plt.show()
####plt.show() avant plt.close()
##plt.close()
```

1.4 Le mouvement Brownien

Le processus $(W_t)_{t\in[0,T]}, T > 0$ est un mouvement Brownien (standard) si

Le mouvement Brownien est issu de l'origine :

$$\mathbb{P}(W_0 = 0) = 1 \tag{9a}$$

$$\forall t \leq s\ ,\ W_s - W_t \sim \mathcal{N}(0, s-t) \tag{9b}$$

Le mouvement Brownien est à accroissements indépendants :

$$\forall t_0 \leq t_1 \leq \ldots \leq t_k\ ,\ W_{t_0}, W_{t_1} - W_{t_0}, \ldots, W_{t_k} - W_{t_{k-1}} \text{ sont indépendantes} \tag{9c}$$

Pour la simulation d'un mouvement Brownien standard, on a le code suivant :

code 5 – mouvement Brownien

```
import matplotlib.pyplot as plt

import numpy
import random

def Mouvement_Brownien(nb_points=1000):
    z=numpy.random.randn(nb_points)
    W=numpy.zeros(nb_points)
    #valeur initiale W_0=0
    for i in numpy.arange(1,nb_points,1):
        W[i] = W[i-1]+z[i]
    return W

fig = plt.figure(figsize=(8,5))
ax = fig.add_subplot(111)

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')

plt.plot(Mouvement_Brownien())
plt.xlabel('$t$', fontsize=16)
plt.ylabel('$W(t)$', fontsize=16)
plt.show()
```

Exercice 1. : Processus aléatoire à temps continu. Soit U une variable aléatoire suivant une loi uniforme sur l'intervalle $[0, 2\pi]$, on définit le processus à temps continu X_t par : $X_t = \cos\big(2\pi t + U\big)$.
$\mathbb{E}[X_t] =?$, $cov\big(X_t, X_s\big) =?$, $Var[X_t] =?$, X_t est-il stationnaire ?

Exercice 2. : Processus aléatoire à temps discret. Soit U et V deux variables aléatoires indépendantes suivant la même loi uniforme sur l'intervalle $[0, 2\pi]$, pour $t \in \mathbb{N}$, on définit le processus à temps discret X_t par : $X_t = \cos\big(Vt + U\big)$.
$\mathbb{E}[X_t] =?$, $cov\big(X_t, X_s\big) =?$, $Var[X_t] =?$, X_t est-il stationnaire ?

Exercice 3. : Processus aléatoire à temps discret. Soit $U_1, \ldots, U_n, V_1, \ldots, V_n$ des variables aléatoires indépendantes. On suppose que $U_1, \ldots, U_n$ suivent la même loi uniforme sur l'intervalle $[0, 2\pi]$ et $V_1, \ldots, V_n$ suivent la même exponentielle de paramètre 1. Pour $t \in \mathbb{N}$, on définit le processus à temps discret X_t par : $X_t = \sum_{i=1}^{n} V_i \cos\big(U_i t\big)$.
$\mathbb{E}[X_t] =?$, $cov\big(X_t, X_s\big) =?$, $Var[X_t] =?$, X_t est-il stationnaire ?

Exercice 4. : Processus stationnaire au second ordre. Soit $\big(Y_t\big)_t$ un processus stationnaire au second ordre, on définit le processus X_t par : $X_t = Y_t - Y_{t-1}$.
X_t est-il stationnaire ?

Exercice 5. : Mouvement Brownien. Soit $\big(W_t\big)_{t\in[0,T]}$, un mouvement Brownien, pour $s \geq 0$, on définit le processus X_t par : $X_t = W_{t+s} - W_s$.
Montrer que X_t est un mouvement Brownien.

2 Série temporelle ou chronologique

Les séries temporelles (ou séries chronologiques) sont des séries d'observations d'une (ou plusieurs) quantité(s) au cours du temps. On parle, selon les cas, de séries uni-variées, ou de séries multi-variées. Ce cours développe essentiellement les techniques relatives aux séries uni-variées, tout en faisant une incursion dans les séries multi-variées pour des problèmes particuliers. Pour les séries uni-variées, les observations appartiennent à l'ensemble des réels . Cependant, dans les cas où elles sont complexes (par exemple, le signal observé à la sortie d'un récepteur de radar), elles sont encore considérées comme uni-variées.

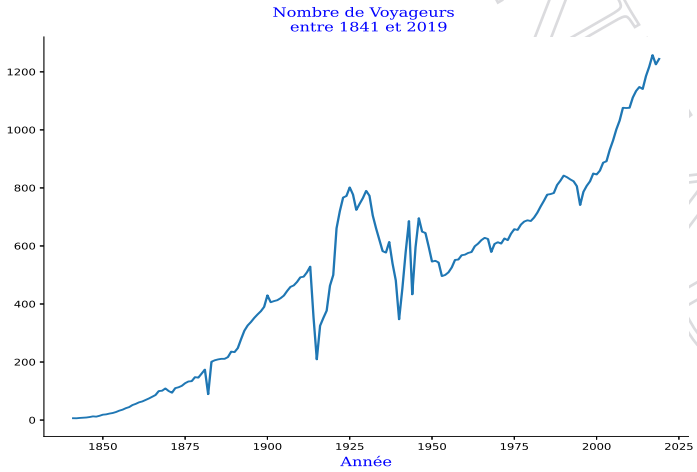
Les instants d'observations sont discrets, sans être nécessairement répartis de façon régulière.

Dans ce cours nous désignons par $(X_t)_{1\leq t\leq T}$ la série temporelle (ou encore une série chronologique) la suite finie de données indexées par le temps. L'indice temps peut être selon les cas la minute, l'heure, le jour, l'année etc.... Le nombre T est appelé la longueur de la série. Il est la plupart du temps bien utile de représenter la série temporelle sur un graphe construit de la manière suivante : en abscisse le temps, en ordonnée la valeur de l'observation à chaque instant. Pour des questions de lisibilité, les points ainsi obtenus sont reliés par des segments de droite. Le graphe apparaît donc comme une ligne brisée.

2.1 Exemples

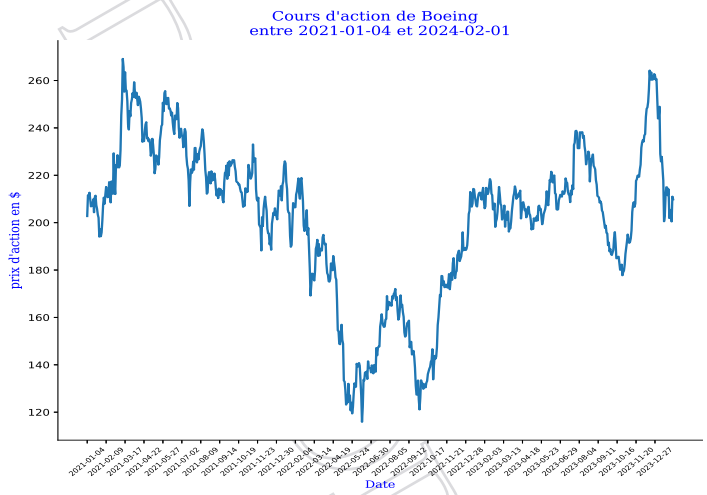
Exemple 3.
Évolution des trafics de voyageurs et marchandises depuis 1841, pour le transport ferroviaire en France (comprenant EPIC SNCF-iDTGV, Eurostar, Thalys sur la période récente).

(source : https://data.sncf.com/explore/)



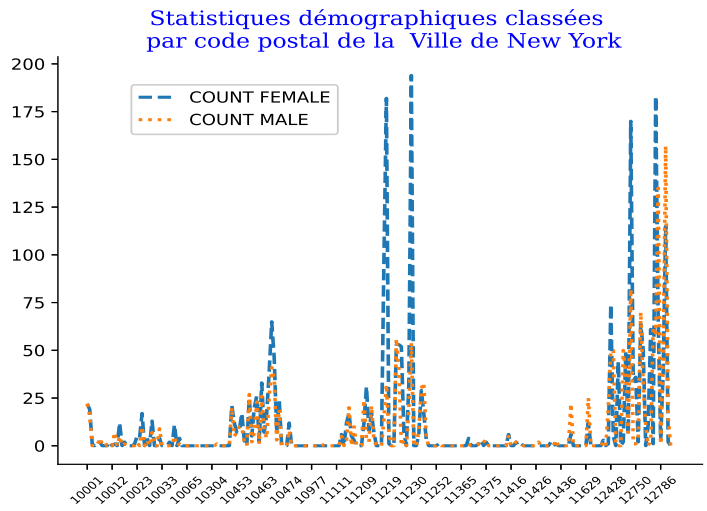
code source Python 1. voir : code_001.py

Exemple 4.
Finance. Cours de bourse quotidiens.
(source : https://fr.finance.yahoo.com/quote/BA/history?p=BA)



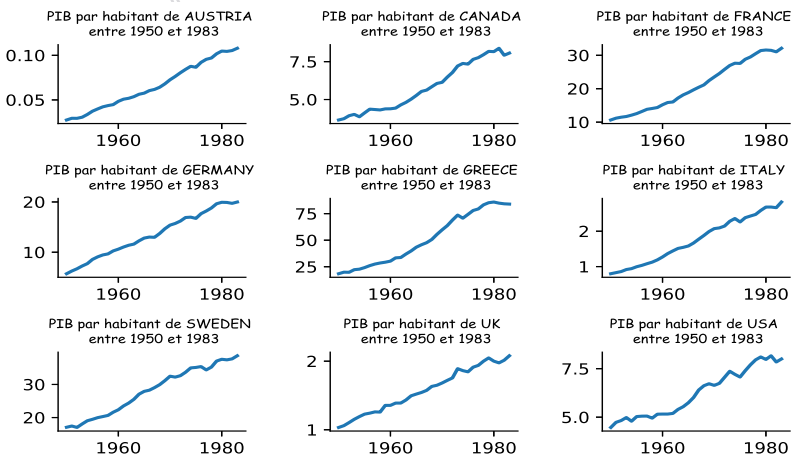
code source Python 2. voir : code_002.py

Exemple 5.
Statistiques démographiques classées par code postal de la Ville de New York.
(source : https://catalog.data.gov/dataset?res_format=CSV)



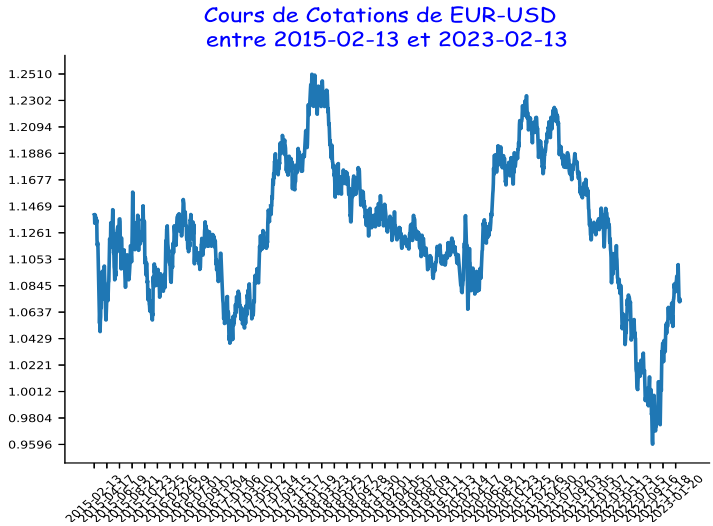
code source Python 3. voir : code_003.py

Exemple 6.
PIB. Séries chronologiques annuelles du PIB par habitant pour plusieurs pays entre 1950 et 1983.
(source : http://www2.stat.duke.edu/ mw/mwsoftware/moredata/ts_data_sets.html)



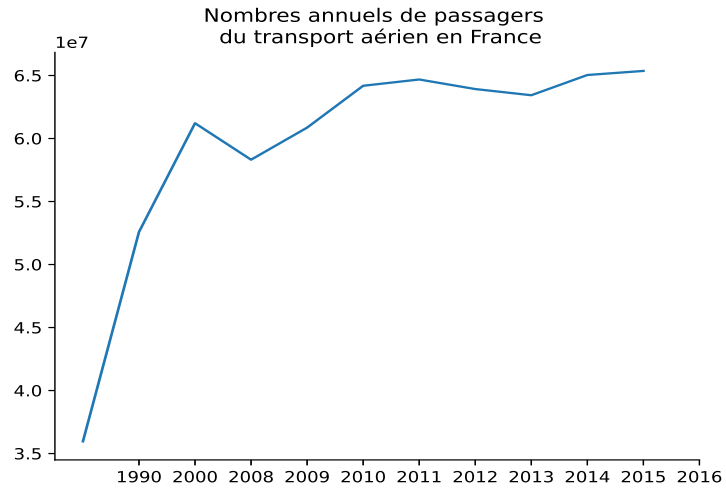
code source Python 4. voir : code_004.py

Exemple 7.
Cotations de EUR-USD du 13/02/2015 au 13/02/2023.
(source : https://fr.finance.yahoo.com/quote/EURUSD%3DX/history?p=EURUSD%3DX.
Le fichier de données contient : l'identifiant de la valeur, la date, le cours d'ouverture, le plus haut, le plus bas, le cours de clôture et le volume de titres échangés.)



code source Python 5. voir : code_005.py

Exemple 8.
Nombres de passagers du transport aérien transportés.
(source : http://databank.worldbank.org/data/)



code source Python 6. voir : code_006.py

2.2 Description d'une série chronologique

Une série $(X_t)_{1\leq t\leq T}$ est souvent décomposée en plusieurs parties :

- **Tendance (trend en anglais)** représente l'évolution à long terme
- **Saison** correspond à de phénomènes répètes dans le temps à des intervalles réguliers
- **Cycle** composante de moyen terme avec un intervalle de répétition plus
- **Résidu ou Erreur** , fluctuations aléatoires de faible intensité.

2.3 Objectifs de l'étude d'une série chronologique

1. Décrire, expliquer un phénomène evoluant au cours du temps et modélisation. D'eterminer les différentes composantes d'une série, en particulier, obtenir la s'erie corrigée des variations saisonnières.

2. Prévision. Prévoir les valeurs futures. Sur la base d'observation $(X_1, \ldots, x_T)$ faire une prévision, à la date T, de la réalisation en date $T + \tau$.

3. Lissage. Le lissage consiste à transformer une série de façon à détecter (pour éliminer ou au contraire conserver) certaines caractérisques (composante saisonnière, points abérants).

3 Modèle linéaire

La série $(X_t)_{1\leq t\leq T}$ est la somme de deux composantes déterministes : une tendance f, d'une saisonnalité S et d'une composante aléatoire ε

$$X_t = \underbrace{f(t)}_{\text{Tendance}} + \underbrace{S(t)}_{\text{Saison}} + \underbrace{\varepsilon_t}_{\text{Erreur}} \tag{10}$$

où $\forall t\ \varepsilon_t \sim \mathcal{N}(0,\sigma^2)$, $\varepsilon_1,\ldots,\varepsilon_T$ sont indépendantes et $\mathbb{E}[\varepsilon_i\varepsilon_j] = \begin{cases} 0 \text{ si } i\neq j \\ \sigma^2 \text{ si } i=j \end{cases}$

On suppose que f et S sont des combinaisons linéaires de fonctions connues dans le temps :

$$\begin{cases} f(t) = \sum_{j=1}^{m} \alpha_j\, f_j(t) \\ S(t) = \sum_{k=1}^{r} \beta_k\, S_k(t) \end{cases} \tag{11}$$

L'objectif est d'estimer les paramètres $(\alpha_1,\ldots,\alpha_m,\beta_1,\ldots,\beta_r)$ en fonction des observations $(X_1,\ldots,X_T)$.
La tendance $(f_1,\ldots,f_m)$ représente l'évolution à long terme de la grandeur étudiée, et traduit l'aspect général de la série. C'est une fonction monotone, souvent polynomiale. Les variations saisonnières $(S_1,\ldots,S_r)$ ont liées au rythme imposé par les saisons météorologiques (production agricole, consommation de gaz,...), ou encore par des activités économiques et sociales (vacances, solde, etc).

On rencontre souvent aussi des phenomènes pour les quelles la pèriode peut elle même varier. On parle alors de **cycles** $(C_1,\ldots,C_r)$, qui regroupent des variations à période moins precise autour de la tendance, par exemple les phases économiques d'expansion et de recession. Ces phases durent généralement plusieurs années, mais n'ont pas de durée fixe. Sans informations spécifiques, il est généralement tr'es difficile de dissocier la tendance du cycle. Dans le cadre de ce cours, la composante appelée tendance regroupera pour la plupart du temps aussi les cycles.
Les résidues $(\varepsilon_1,\ldots,\varepsilon_T)$ sont des variations de faible intensité et de courte durée, et de nature aléatoire (ce qui signifie ici, dans un cadre purement descriptif, qu'elles ne sont pas compl'etement expliquables). En effet, elles ne sont pas clairement apercevables dans les graphiques, à cause de leur faible intensité par rapport aux autres composantes.

3.1 Composante tendancielle

Cette composante a généralement une forme simple, reflétant la croissance moyenne. Plusieurs types de composantes tendancielles existent :

1. linéaire (régression linéaire) :
$$f(t) = \alpha_1 + \alpha_2\, t \tag{12}$$
2. quadratique :
$$f(t) = \alpha_1 + \alpha_2\, t + \alpha_3\, t^2 \tag{13}$$
3. polynomiale :
$$f(t) = \sum_{j=1}^{m+1} \alpha_j\, t^{j-1} \tag{14}$$
4. logistique :
$$f(t) = \frac{1}{\alpha_1\, e^{-\alpha_2 t} + \alpha_3} \tag{15}$$

5. exponentielle type I :

$$f(t) = \alpha_1 + \alpha_2\, t_3^{\alpha} \tag{16}$$

6. exponentielle type II :

$$f(t) = \alpha_1 + \alpha_2\, (1+t)^{\alpha_3} \tag{17}$$

7. exponentielle type III :

$$f(t) = \alpha_1 + \alpha_2\, e^{\alpha_3 t} \tag{18}$$

8. modèles avec un point de rupture : $f(t) = \begin{cases} h_1(t) & \text{si} \quad t \leq t_k \\ h_2(t) & \text{si} \quad t > t_k \end{cases}$

3.2 Composante saisonnière

La forme de la composante saisonnière S du modèle dépend du type de données, et de la forme de la saisonnalité. Par exemple pour des données trimestrielles, on a :

$$S(t) = \sum_{k=1}^{4} \beta_k\, S_k(t) \quad \text{avec} \quad S_k(t) = \begin{cases} 1 & \text{si} \quad t = \text{trimestre } k \\ 0 & \text{si} \quad t \neq \text{trimestre } k \end{cases} \tag{19}$$

Écriture matriecielle de S_1, S_2, S_3, S_4 :

$$S_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix} \quad S_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix} \quad S_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix} \quad S_4 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \end{pmatrix}$$

4 Transformation de la série brute

L'intérêt d'une description linéaire de la série réside dans la simplicité du modèle. Ceci permet d'en faire une étude mathématique approfondie. Choisir une telle écriture peut sembler restrictif, cependant l'hypothèse de linéarité est moins forte qu'il n'y paraît. De nombreux modèles peuvent être rendus linéaires après transformation de la série brute.
Le modèle multiplicatif et le modèle logistique peuvent s'écrire comme des modèles linéaires.

4.1 Modèle trimestriel de Buys-Ballot

Le modèle trimestriel de Buys-Ballot (un modèle multiplicatif) s'écrit :

$$\log X_t = \alpha_1 + \alpha_2\, t + \sum_{k=1}^{4} \beta_k\, S_k(t) + \varepsilon_t \quad \forall X_t > 0 \tag{20}$$

où $S_k(t)$ désigne la fonction indicatrice du trimestre définie en (19) et les coefficients saisonniers satisfont $\beta_1 + \beta_2 + \beta_3 + \beta_4 = 0$.
Ce modèle est adapté, par exemple, à l'étude du trafic SNCF agrégée par trimestre.

Pour les composantes saisonnières définies dans (19), on a la fonction suivante :

code 6 – composantes saisonnières

```
def indi_saison(N,k=4):
    """
    N=nombre d'observations
    k=période
    retourne : k vecteurs de dim=N
    """
    S=[[] for _ in range(k)]
    for i in range(N):
        for j in range(k):
            if i%k==j:
                S[j].append(1)
            else:
                S[j].append(0)

    return S

## Exemples d'Appel :
print(indi_saison(15))
print(indi_saison(15,3))
```

Pour créer la matrice contenant les composantes saisonnières $S_1, \ldots, S_r$, on a le code suivant :

code 7 – matrice de composantes saisonnières

```
import numpy
#  on fait appel à la fonction : indi_saison
def Matrice_saison(N,k=4):
    """
    Matrice avec les colonnes  S_1,...,S_k
    retourne matrice à N: lignes et k: colonnes
    """
    return numpy.stack(indi_saison(N,k), axis=-1)
```

4.2 Modèle logistique

Le modèle logistique s'écrit :

$$\log\left(\frac{1-X_t}{X_t}\right) = \alpha_1 + \alpha_2\, t + \sum_{k=1}^{r} \beta_k\, S_k(t) + \varepsilon_t \quad \forall X_t \in]0,1[\tag{21}$$

Ce modèle est adapté à l'étude de la consommation de certains biens durables. X_t représente alors la proportion de ménages possédant ce bien à l'instant t.

4.3 Moyenne Mobile

On appelle moyenne mobile (en anglais, moving average) , une transformation de la série $(X_t)_{1\leq t\leq T}$ s'érivantnt comme combinaison linéaire finie des valeurs de la série correspondant à des dates entourant t . La série transformée s'écrit :

$$M_{n_1+n_2+1}(t) = \sum_{j=-n_1}^{n_2} \psi_j X_{j+t} \tag{22}$$

où $\psi_j, j = -n_1, \ldots, n_2$ sont des réels et $n_1 \in \mathbb{N}$, $n_2 \in \mathbb{N}$.

Remarque 3. L'équation (22) n'a de sens que pour les t vérifiant : $1 + n_1 \leq t \leq T - n_2$.

Un cas particulier de (22), est une moyenne mobile arithmétique, (moyenne mobile centrée, d'ordre (impair) $2n+1$ avec $n_1 = n_2 = n$) qui est de la forme :

$$\psi_j = \frac{1}{2n+1} \quad \forall j = -n, \ldots, n \tag{23a}$$

$$M_{2n+1}(t) = \frac{1}{2n+1} \sum_{j=-n}^{n} X_{j+t} \tag{23b}$$

Exemple 9.
Filtre de Hanning :

$$M_H(t) = \frac{1}{4} X_{t-1} + \frac{1}{2} X_t + \frac{1}{4} X_{t+1} \tag{24}$$

Pour le Filtre de Hanning définie dans (24), on a la fonction suivante :

code 8 – filtre de Hanning

```
import matplotlib.pyplot as plt
import numpy

def Filtre_Hanning(data):
    # M_H(t)=0.25*X_{t-1}+0.5*X_{t}+0.25*X_{t+1}   (2 <= t <=T-1)
    T=len(data)
    FH=[]
    for t in range(1,T-1):
        FH.append(0.25*data[t-1]+0.5*data[t]+0.25*data[t+1])

    fig = plt.figure(figsize=(10, 7))
    ax = fig.add_subplot(111)
    # cacher axes 'haut' et 'droit':
    for side in ['right','top']:
        ax.spines[side].set_visible(False)

    plt.plot(data,'-b')
    plt.plot(FH,':r')
    plt.legend(['données brutes', 'données filtrées'])
    plt.show()
    return FH
```

5 Statistiques descriptives d'une série temporelle

Soit $(X_t)_{1\leq t\leq T}$ une série temporelle.

On définit le moment d'ordre r de la série par :

$$M_r = \frac{1}{T}\sum_{t=1}^{T} X_t^r \tag{25}$$

Le moment d'ordre 1, noté $\overline{X}_T$ s'appelle la moyenne empirique de la série.
On définit le moment centré d'ordre r de la série par :

$$MC_r = \frac{1}{T}\sum_{t=1}^{T} \left(X_t - \overline{X}_T\right)^r \tag{26}$$

Le moment centré d'ordre 1 est nul et le moment centré d'ordre 2 s'appelle la variance empirique de la série.

Le coefficient d'asymétrie (skewness en anglais) de la série est définit par :

$$\gamma_1 = \frac{1}{T}\sum_{t=1}^{T} \left(\frac{X_t - \mu}{\sigma}\right)^3 = \frac{MC_3}{(MC_2)^{3/2}} \tag{27}$$

où $\mu = \overline{X}_T$ et $\sigma = \sqrt{MC_2}$.

Le coefficient d'aplatissement de Fisher (kurtosis en anglais) de la série est définit par :

$$\gamma_2 = \frac{1}{T}\sum_{t=1}^{T} \left(\frac{X_t - \mu}{\sigma}\right)^4 - 3 = \frac{MC_4}{(MC_2)^2} - 3 \tag{28}$$

Pour $k \in \{2, ..., T-1\}$, on définit le moment d'ordre r des séries extraits $(X_t)_{1\leq t\leq k}$ et $(X_t)_{k+1\leq t\leq T}$ par :

$$\begin{cases} M_r(k) = \dfrac{1}{k}\displaystyle\sum_{t=1}^{k} X_t^r & \text{(29a)} \\ M_r^\star(k) = \dfrac{1}{T-k}\displaystyle\sum_{t=k+1}^{T} X_t^r & \text{(29b)} \end{cases}$$

Pour $k \in \{2, ..., T-1\}$, on définit le moment centré d'ordre r des séries extraits $(X_t)_{1\leq t\leq k}$ et $(X_t)_{k+1\leq t\leq T}$ par :

$$\begin{cases} MC_r(k) = \dfrac{1}{k}\displaystyle\sum_{t=1}^{k} \left(X_t - M_1(k)\right)^r & \text{(30a)} \\ MC_r^\star(k) = \dfrac{1}{T-k}\displaystyle\sum_{t=k+1}^{T} \left(X_t - M_1^\star(k)\right)^r & \text{(30b)} \end{cases}$$

Remarque 4. Dans le cadre des modèles de détection de Rupture, on prend $k \in \{4, ..., T-4\}$.

Pour les différents moments d'une série, définis dans (25), (26), (27), (28), (29), (30) , on a le code suivant :

code 9 – fonctions moments

```
import numpy

def Moment_r(data,r):
    return numpy.mean(numpy.power(data,r))

def Moment_cr(data,r):
    m1=Moment_r(data,1)
    X_C=numpy.array(data)-m1
    return Moment_r(X_C,r)

def Moment_partiel_r(data,k,r):
    N=len(data)
    ## 1 <= k <= N-1
    if k < 1 or k > N-1 :
        return []
    else:
        data1=data[:k]
        data2=data[k:]
    return Moment_r(data1,r), Moment_r(data2,r)


def Moment_partiel_cr(data,k,r):
    N=len(data)
    ## 1 <= k <= N-1
    if k < 1 or k > N-1 :
        return []
    else:
        data1=data[:k]
        data2=data[k:]
    return Moment_cr(data1,r), Moment_cr(data2,r)

## Exemples d'Appel :
print(Moment_r([1,2,7],2))
print(Moment_cr([1,2,7],1))
print(Moment_partiel_r([1,2,7,6,5,7],2,1)[0])
print(Moment_partiel_r([1,2,7,6,5,7],2,1)[1])
print(Moment_partiel_cr([1,2,7,6,5,7],2,2)[0])
print(Moment_partiel_cr([1,2,7,6,5,7],2,2)[1])
```

Fonctions d'auto-covariance et d'auto-corrélation

La fonction auto-covariance mesure la dépendance temporelle linéaire entre deux temps et est définie par :

$$C(h,k) = \frac{1}{T} \sum_{t=\max(1-h,1-k)}^{\min(T-h,T-k)} \left(X_{t+h} - \overline{X}_T\right)\left(X_{t+k} - \overline{X}_T\right) \tag{31}$$

La fonction d'auto-corrélation est définie par :

$$R(h,k) = \frac{1}{T} \sum_{t=\max(1-h,1-k)}^{\min(T-h,T-k)} X_{t+k}\, X_{t+k} \tag{32}$$

La coefficient d'auto-corrélation est définie par :

$$\rho(h,k) = \frac{C(h,k)}{C(0,0)} \tag{33}$$

6 Estimation des paramètres pour le modèle linéaire

Dans cette partie on suppose que la série $(X_t)_{1\leq t\leq T}$ est la somme d'une composante déterministe : tendance f et d'une composante aléatoire ε : $X_t = f(t,\theta)+\varepsilon$ où θ désigne le paramètre du modèle. $\hat{\theta}$ l'estimateur de θ est solution de :

$$\hat{\theta} = \underset{\theta}{\text{Min}} \sum_{t=1}^{T} \big(X_t - f(t,\theta)\big)^2 \tag{34}$$

L'estimateur de la variance résiduelle σ^2 est donné par :

$$\hat{S}^2 = \frac{1}{T-p} \sum_{t=1}^{T} \big(X_t - f(t,\hat{\theta})\big)^2 \quad , \; p = \dim\,\theta \tag{35}$$

6.1 Estimation des paramètres pour la tendance linéaire (régression linéaire)

Pour la tendance linéaire définie dans (12), les estimateurs de (α_1,α_2) sont solutions de :

$$(\hat{\alpha}_1,\hat{\alpha}_2) = \underset{(\alpha_1,\alpha_2)}{\text{Min}} \sum_{t=1}^{T} \big(X_t - \alpha_1 - \alpha_2 t\big)^2 \tag{36}$$

(**Démonstration en cours**).

Pour l'estimation des paramètres de ce modèle, on a le code suivant :

code 10 – tendance linéaire

```
import scipy.optimize as opt
import numpy
import random
### X=a1+a2*t
def tendance_Lin(data):
    ## sous fonction pour la partie d'optimisation
    def Objective_Lin(alpha, data,t):
        return data-alpha[0]-alpha[1]*t

    # valeur initiale
    A0= numpy.array([random.random() for _ in range(2) ])
    t=numpy.array([i for i in range(len(data))])
    R=opt.leastsq(Objective_Lin, A0, args = (data, t))[0]
    return R

### Exemple d'appel    ###
N=500
t=numpy.array([i for i in range(N)])
X = [numpy.random.normal(2.0+3*tt,1) for tt in t]
e = numpy.random.normal(0,1,N)
data = X + e
tendance_Lin(data)
> array([ 2.16006619,  2.99965803])
```

6.2 Estimation des paramètres pour la tendance quadratique

Pour la tendance quadratique définie dans (13), les estimateurs de $(\alpha_1, \alpha_2, \alpha_3)$ sont solutions de :

$$(\hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_3) = \underset{(\alpha_1,\alpha_2,\alpha_3)}{\text{Min}} \sum_{t=1}^{T} \left(X_t - \alpha_1 - \alpha_2 t - \alpha_3 t^2\right)^2 \tag{37}$$

(**Démonstration en cours**).

Pour l'estimation des paramètres de ce modèle, on a le code suivant :

code 11 – tendance quadratique

```
import scipy.optimize as opt
import numpy
import random
### X=a1+a2*t+a3*t^2
def tendance_quad(data):
    ## sous fonction pour la partie d'optimisation
    def Objective_quandratique(alpha, data,t):
        return data-alpha[0]-alpha[1]*t-alpha[2]*t**2

    # valeur initiale
    A0= numpy.array([random.random() for _ in range(3) ])
    t=numpy.array([i for i in range(1,1+len(data))])
    R=opt.leastsq(Objective_quandratique, A0, args = (data, t))[0]
    return R

### Exemple d'appel    ###
N=500
t=numpy.array([i for i in range(1,1+N)])
X = [numpy.random.normal(1.75-4*tt+7*tt**2,1) for tt in t]
e = numpy.random.normal(0,1,N)
data = X + e
tendance_quad(data)

> array([ 1.53717472, -3.99936998,  6.99999863])
```

6.3 Estimation des paramètres pour la tendance polynomiale

Pour la tendance polynomiale définie dans (14), les estimateurs de $(\alpha_1, \ldots, \alpha_{m+1})$ sont solutions de :

$$(\hat{\alpha}_1, \ldots, \hat{\alpha}_{m+1}) = \underset{(\alpha_1,\ldots,\alpha_{m+1})}{\text{Min}} \sum_{t=1}^{T} \left(X_t - \sum_{j=1}^{m+1} \alpha_j \, t^{j-1}\right)^2 \tag{38}$$

Pour l'estimation des paramètres de ce modèle, on a le code suivant :

code 12 – tendance polynomiale

```
from matplotlib import pylab as plt
import scipy.optimize as opt
import numpy
import random

def polynome_M(coeff,t):
        return numpy.sum([coeff[j]*numpy.power(t,j) for j in range(len(coeff))])

### X=a1+a2*t+a3*t^2+....+a_{m+1}*t^m
def tendance_polynomiale(data,m=2):
    ### m-1 degré du polynôme
    ### m=2 : régression linéaire
    ### m=3 : quadratique
    ### sous fonction pour la partie d'optimisation
    def Objective_polynomiale(alpha, data,t):
        return numpy.array([data[i] - polynome_M(alpha,t[i]) for i in range(len
            (data))])

    # valeur initiale
    A0= numpy.array([random.random() for _ in range(m) ])
    t=numpy.array([i for i in range(1,1+len(data))])
    R=opt.leastsq(Objective_polynomiale, A0, args = (data, t))[0]
    return R

N=100
t=numpy.array([i for i in range(1,1+N)])
X = [numpy.random.normal(polynome_M([2,-4,7],tt),1) for tt in t]
e = numpy.random.normal(0,1,N)
data = X +e
R=tendance_polynomiale(data,m=3)
##plt.figure(figsize=(10, 6), dpi=250)
plt.figure()
ax = plt.gca()
ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)
ax.xaxis.set_ticks_position('bottom')
ax.plot(X,'.r')
ax.plot([R[0]+R[1]*tt+R[2]*tt**2  for tt in t])
plt.title("$\\hat{\\alpha}_{1}=%.4f , \\hat{\\alpha}_{2}=%.4f , \\hat{\\alpha}_
    {3}=%.4f$"%(R[0],R[1],R[2]))
plt.show()
```

Exercice 6. : Écrire une fonction en langage python pour l'estimation des paramètres du modèle logistique défini dans (15)

Application aux données de l'exemple 6

Exercice 7. : Écrire une fonction en langage python pour l'estimation des paramètres du modèle exponentielle type I défini dans (16)

Application aux données de l'exemple 6

Exercice 8. : Écrire une fonction en langage python pour l'estimation des paramètres du modèle exponentielle type II défini dans (17)

Application aux données de l'exemple 6

Exercice 9. : Écrire une fonction en langage python pour l'estimation des paramètres du modèle exponentielle type III défini dans (18)

Application aux données de l'exemple 6

Liste des Codes