Séries temporelles élémentaires et Mathematica

Le rôle du temps, de la durée, est essentiel dans de nombreuses questions géographiques.

Le géographe face à six questionnements

La première concerne la stationnarité de la série temporelle. Le concept de stationnarité signifie que le futur sera similaire au passé, que la moyenne et la variance ne changent pas. Une deuxième question concerne la dépendance temporelle, appelée aussi persistance ou inertie, des phénomènes géographiques. Souvent, la valeur d'une série temporelle au temps t est dépendante de celle prise au temps t-1. La population de la France en 2016 est dépendante de celle recensée en 2015. Dans tous les cas, cette question se décompose en trois sous-questions. D'abord quelle est la durée de cette persistance ? Puis, quelle est l'intensité de cette persistance ? Enfin, est-ce une persistance ou une anti-persistance, quand une forte valeur au temps t est suivie d'une faible valeur au temps t+k ? Un troisième groupe de questions concerne la décomposition d'une série en tendance, cycles, et variations aléatoires. Une quatrième question s'attache aux accidents bien réels, et qui ne sont pas une simple erreur commise lors du recueillement des données. Ces événements, qui correspondent à des valeurs maximales ou minimales de la série analysée, doivent être repérés. Puis, leur intensité sera mise en valeur. La cinquième question a pour objectif de déterminer le modèle sous-jacent aux données temporelles. Ce sont des processus stochastiques. Enfin, il est nécessaire d'étudier comment une première série temporelle est liée à une ou d'autres chroniques. Ce problème est plus compliqué qu'il n'y parait. En effet, une variable peut avoir un effet immédiat sur une autre variable. Mais, cette situation est exceptionnelle. Le plus souvent, on observe un décalage dans le temps, et ignorer ce déphasage conduit à des raisonnements infondés. Ce type d'évolution, avec retard, est la règle.

Tester la stationnarité et la présence de cycles d'une série temporelle

Repérer la stationnarité ou la non-stationnarité d'une chronique est une opération préalable à toute étude d'une évolution. On en distingue deux. La première est qualifiée de Trend Déterministe (TD). La seconde est d'origine stochastique (DS pour Differency Stationarity). C'est la tendance qui varie au cours du temps, par exemple quand une hausse prolongée fait suite à une baisse tout aussi longue. Sur le plan statistique, la non-stationnarité a d'importantes conséquences. Elle conditionne en partie le choix du modèle à utiliser pour représenter l'évolution du phénomène étudié. De plus, l'estimation des paramètres du modèle est plus ou moins biaisée. Surtout, pour « stationariser » la série, il faut employer la bonne technique, soit retirer la composante tendancielle, à l'aide d'une régression, pour un processus de type TD, soit procéder à une différenciation pour un processus de type DS. Mais ces deux non-stationnarités ont aussi des effets en termes d'interprétation disciplinaire, car elles ont des comportements totalement différents face aux perturbations, qu'il s'agisse d'un événement physique, comme une éruption volcanique, ou de l'introduction d'une innovation économique ou culturelle dans une société. Quand une série non stationnaire de type TD est perturbée par un choc, les impacts de ce choc tendent à s'estomper puis à disparaître à mesure que le temps passe. Cela signifie que la trajectoire sur le long terme n'est pas déviée par les aléas conjoncturels. En revanche, pour un processus non-stationnaire de type DS les impacts des perturbations perdurent, s'atténuent ou s'amplifient lentement avec le temps.

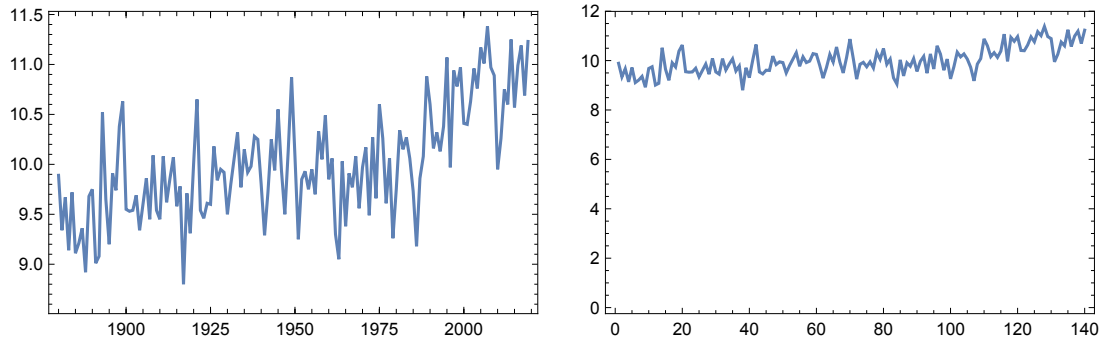
Pour déterminer si une série temporelle est stationnaire ou non-stationnaire, le géographe dispose de plusieurs outils. Cinq sont employés : la comparaison de moyennes et de variances, le corrélogramme, la technique des racines unitaires, l'exposant d'un spectre de puissance, et les graphiques de récurrence. Commençons par les représentations graphiques tout en sachant qu'elle peuvent tromper le lecteur.

Représenter la série brute avec DateListPlot[]

Voici une double représentation des températures annuelles pour la station de Valentia en Irlande.

In[1150]:=
```
ClearAll["Global`*"]
```
efface tout
```
data = ToExpression[Flatten[Import[SystemDialogInput["FileOpen"]]]];
```
convertis en une ⋯ aplatis importe entrée de dialogue de système
```
nom = {"Valentia"};
ntab = Length[data];
```
longueur
```
g1 = DateListPlot[data, {Automatic, "2019", "Year"}];
```
tracé de liste de dates automatique
```
g2 = ListLinePlot[data, Frame → True, AxesOrigin → {1, 0}];
```
tracé de liste de ligne cadre vrai origine des axes
```
GraphicsRow[{g1, g2}]
```
rangée de graphiques

Out[1156]=



Après avoir effacé la mémoire, le programme importe les données produites recueillies et pré-traitées par le Goddart Institute de la NASA. Le nom de la station, puis la longueur de la série sont transférées dans nom et ntab. Deux graphiques sont alors réalisés puis affichés conjointement sur une ligne graphique. La deuxième représentation graphique, la moins lisible, est cependant la plus exacte. En outre, le coefficient d'aplatissement, égal à 2,63, indique une distribution à traîne épaisse.

Illustrer l'évolution des moyennes et des variances

Il est possible de visualiser l'évolution des moyennes et des variances au cours du temps. Le programme ci-dessous calcule tous les 20 ans, la moyenne et la variance des données, puis présente le résultat sous une forme graphique.

```
Print["Évolutions des moyennes et des variances"]
```
imprime

```
moy = TimeSeriesAggregate[data, 20, Mean];
```
ajoute série temporelle — valeur moyenne

```
f1 = ListLinePlot[moy, AxesOrigin → {1, 0}, PlotLegends → nom];
```
tracé de liste de ligne — origine des axes — légendes de tracé

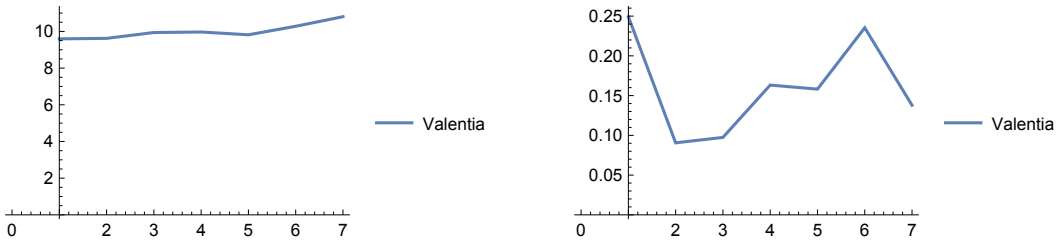
```
var = TimeSeriesAggregate[data, 20, Variance];
```
ajoute série temporelle — variance

```
f2 = ListLinePlot[var, AxesOrigin → {1, 0}, PlotLegends → nom];
```
tracé de liste de ligne — origine des axes — légendes de tracé

```
GraphicsRow[{f1, f2}]
```
rangée de graphiques

```
Évolutions des moyennes et variances
```

Out[•]=



Le lecteur remarquera la faible variation de la moyenne et, au contraire, une forte variabilité de la variance au cours du temps.

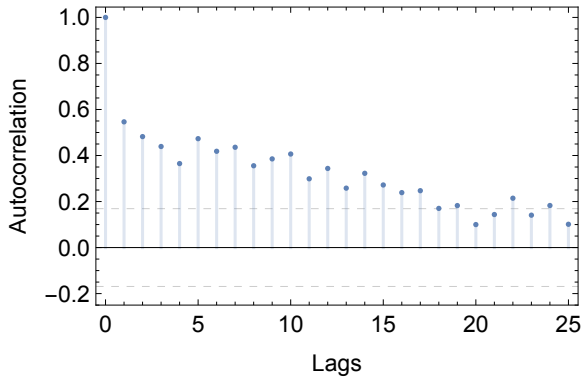
Le corrélogramme source d'information sur la non-stationnarité

Le corrélogramme, visualise les autocorrélations ou corrélations sérielles. Une décroissance très lente des autocorrélations, à mesure que le temps augmente, est le signe d'une non-stationnarité. Les instructions suivantes illustrent les autocorrélations, puis les autocorrélations partielles.

In[1054]:=
```
Print["graphique des autocorrelations"]
imprime
DiscretePlot[CorrelationFunction[data, i], {i, 0, 25},
tracé discret  fonction de corrélation
 PlotRange → {-0.25, 1.045}, Frame → True, Axes → True,
 zone de tracé  cadre  vrai  axes  vrai
 GridLines → {None, {{2 / Sqrt[ntab], Dashed}, 0, {-2 / Sqrt[ntab], Dashed}}},
 lignes de grille  aucun  racine carrée  rayé  racine carrée  rayé
 FrameLabel → {"Lags", "Autocorrelation"},
 titre de cadre
 ImageSize → 300, BaseStyle → {FontSize → 12}]
 taille d'image  style de base  taille de police de caractères
Print["graphique des autocorrelations partielles"]
imprime
DiscretePlot[PartialCorrelationFunction[data, i],
tracé discret  fonction de corrélation partielle
 {i, 1, 25}, PlotRange → {-0.3, 1.045}, Frame → True, Axes → True,
  zone de tracé  cadre  vrai  axes  vrai
 GridLines → {None, {{2 / Sqrt[ntab], Dashed}, 0, {-2 / Sqrt[ntab], Dashed}}},
 lignes de grille  aucun  racine carrée  rayé  racine carrée  rayé
 FrameLabel → {"Lags", "Partial Autocorrelation"},
 titre de cadre
 ImageSize → 300, BaseStyle → {FontSize → 12}]
 taille d'image  style de base  taille de police de caractères
```

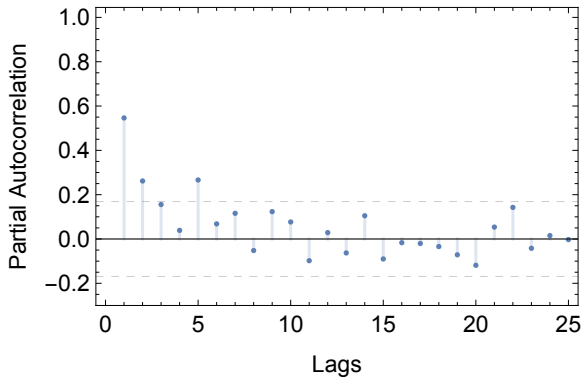
Out[1055]=

graphique des autocorrelations



Out[1057]=

graphique des autocorrelations partielles



La première figure montre que les autocorrélations persistent un certain temps. Ce qui est un indicateur de non stationnarité. Mais on ignore face à quel type de non stationnarité se trouve le géographe. Cepen-

dant, ce graphique indique la présence d'une traîne épaisse, donc des températures qui s'éloignent plus grandement de la moyenne que dans une série gaussienne.

Le modèle graphique de récurrence

Depuis quelques années, les diagrammes de récurrence et divers indicateurs, qui en dérivent, permettent d'observer les grands types d'évolutions temporelles.

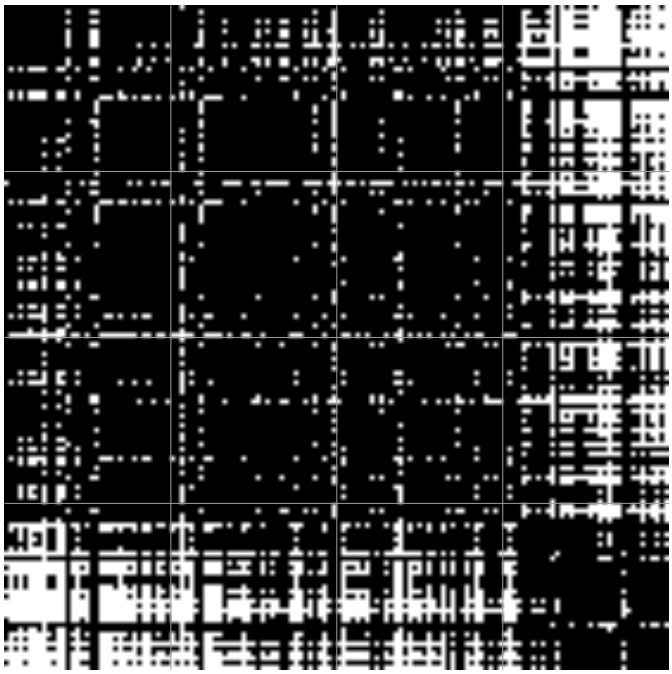
```
Print["diagramme  des proximités ou recurrenceplot"]
```
imprime

```
ResourceFunction["CrossRecurrencePlot"][
```
fonction ressource

```
 data, data, FrameTicks → Automatic, Mesh → {3, 3}]
```
graduations du ⋯ automatique maille

```
diagramme  des proximités ou recurrenceplot
```

Out[ ]=



Comme les angles haut-droite et bas-gauche sont estompés, on peut encore conclure à une non stationnarité de la série, sans pouvoir en préciser l'origine. Malgré tout, l'ensemble de ces graphiques semblent indiquer une non stationnarité de la variance. Il convient alors de réaliser des tests probabilités pour conforter ou infirmer ces résultats.

Tester l'égalité des moyennes et des variances

La première solution consiste à calculer les moyennes et variances sur des plages de temps, par exemple au début et à la fin de la série de données, puis à tester si ces moyennes et variances sont égales. En cas d'égalité, la stationnarité est vérifiée. Il est possible de constater une stationnarité de la moyenne, mais une non stationnarité e la variance. Par contre, une non stationnarité de la moyenne entraîne une non stationnarité de la variance. Le programme ci-dessous partitionne la série initiale en 4 séries. Puis les tests pour l'égalité des moyennes est effectué sur les 4 séries, et cette approche est répétée pour les deux seuls quarts extrêmes de la série, celui du début et celui de la fin.

```
In[•]:= dataquart = Partition[data, Floor[Length[data] / 4]];
        partitionne   entier ··  longueur

LocationEquivalenceTest[{dataquart[[1]], dataquart[[2]],
test d'équivalence de position
  dataquart[[3]], dataquart[[4]]}, {"TestDataTable", All}]
                                                  tout

VarianceEquivalenceTest[{dataquart[[1]], dataquart[[2]],
test d'équivalence de variance
  dataquart[[3]], dataquart[[4]]}, {"TestDataTable", All}]
                                                  tout

LocationEquivalenceTest[{dataquart[[1]], dataquart[[4]]},
test d'équivalence de position
 {"TestDataTable", All}]
                  tout

VarianceEquivalenceTest[{dataquart[[1]], dataquart[[4]]},
test d'équivalence de variance
 {"TestDataTable", All}]
                  tout
```

Out[•]=

	Statistic	P-Value
Complete Block F	38.359119	1.1130745×10^{-16}
Friedman Rank	26.183113	1.2196339×10^{-12}
Kruskal-Wallis	54.506678	1.1790322×10^{-14}
K-Sample T	33.727711	2.3185833×10^{-16}

Out[•]=

	Statistic	P-Value
Bartlett	3.4388071	0.32878579
Brown-Forsythe	1.147425	0.3323739
Conover	2.7162344	0.43747547
Levene	1.2676958	0.28803068

Out[•]=

	Statistic	P-Value
Complete Block F	109.80181	3.4759181×10^{-12}
Friedman Rank	123.76515	7.1024892×10^{-13}
Kruskal-Wallis	37.746908	2.603563×10^{-13}
K-Sample T	78.669821	5.805789×10^{-13}

Out[•]=

	Statistic	P-Value
Bartlett	1.4227991	0.23294323
Brown-Forsythe	1.7184928	0.19429709
Conover	−1.329246	0.18376682
Fisher Ratio	0.66127324	0.23295542
Levene	2.0008108	0.1617778

Pour chaque analyse plusieurs tests sont mobilisés de façon automatique. Tous conduisent à la même conclusion. Ils confirment ce que les graphiques visualisaient, une non stationnarité due, non pas à l'évolution de la moyenne, mais à celle de la variance. Ce qui ouvre le champ à plusieurs hypothèses.

Le test des racines unitaires

Le principe du deuxième test, celui de la racine unitaire, est moins simple. Des livres entiers sont consacrés à ce test. Il s'agit de calculer la valeur d'une racine à partir de la série de données comparée à un modèle de processus qui est supposé être à l'origine de cette chronique. Quand cette valeur est élevée par rapport à une valeur probabiliste que donnent les tables de Dickey-Fuller, la série est non-stationnaire. En fait, on considère non pas un mais trois modèles hypothétiques, et on procède donc à trois tests successifs. Le premier porte sur un modèle de chemin aléatoire, le deuxième sur un modèle de chemin aléatoire avec une dérive constante, et enfin le troisième sur un modèle de chemin aléatoire avec une dérive autour d'une tendance aléatoire. Un chemin est la suite des différences premières de la série, c'est-à-dire les écarts successifs entre les valeurs t et t+1. Cette deuxième approche offre l'avantage de distinguer la non-

stationnarité de type TD de la non-stationnarité DS. Puis, ce test de Dickey Fuller fut amélioré pour l'adapter à diverses contraintes. Ainsi, le test de Philipps et Perron prend en compte les erreurs d'hétéroscédascité, c'est-à-dire quand les variances de la variable découpée en intervalles changent. Ce qui est vérifié avec l'exemple traité dans ce programme. La fonction UnitRootTest[] inclut ces tests.

```
In[•]:= Print["Test de la racine unitaire sur modèle
         imprime
    AR1 avec moyenne non nulle et trend deterministe "]
res1 = UnitRootTest[data, "Drift", "ShortTestConclusion"]
       test de racine unitaire
res2 = UnitRootTest[data, "Drift", "TestConclusion"]
       test de racine unitaire

Test de la racine unitaire sur modèle
  AR1 avec moyenne non nulle et trend deterministe
```

```
Out[•]= Reject
```

```
Out[•]= The null hypothesis that the model of order 1
    with a constant offset and a deterministic trend contains a unit root
 is rejected at the 5 percent level based on the Dickey-Fuller F test.
```

Ce premier test indique qu'aucune tendance n'est décelable sur la série pour un modèle autorégressif d'ordre 1. Mais qu'en est-il de modèles d'ordre 2, 3,... Pour le savoir, on utilise le programme ci-dessous :

```
In[•]:= pmax[n_] := IntegerPart[12 (n/100)^(1/4)]
                    partie entière

i = pmax[Length[data]];
          longueur

While[
tant que
  Abs[UnitRootTest[data, {"Constant", i}, {"TestStatistic", "DickeyFulleF"}]] <
  va··· test de racine unitaire      constante
   1.6, i--
 ];
i
UnitRootTest[data, {"Constant", i}, {"DickeyFullerF", "TestConclusion"}]
test de racine unitaire      constante
```

```
Out[•]= 13
```

```
Out[•]= {0.8100927,
 The null hypothesis that the model of order 13 with a constant offset and no
     deterministic trend contains a unit root and the zero offset
  is not rejected at the 5 percent level based on the Dickey-Fuller F test.}
```

Le programme détermine un ordre, AR13, où le test n'est plus rejeté. Il y a une racine unitaire, synonyme d'une tendance pour ce modèle.

Le test des spectres de puissance

Nous renvoyons le lecteur à notre ouvrage pour comprendre les décompositions de Fourier ou en ondelettes qui servent à calculer un spectre de puissance, parfois qualifié de spectre d'énergie ou de spectre de variance. Quand les points de ce graphique s'ordonnent suivant une droite pentue, la pente de cette droite, α, est déterminée par régression. Cette valeur α sert de test pour vérifier la stationnarité

d'une série, puis à départager les types de processus qui sont à l'origine de cette chronique. Si la valeur de la pente est comprise entre -1 et +1, la série est stationnaire. Les données matérialisent un bruit gaussien fractionnaire. Mais quand la valeur de la pente est comprise entre +1 et +3, la variance de cette série augmente avec le nombre d'observations. La chronique est non stationnaire. Pour surmonter diverses contraintes, il est préférable d'employer une décomposition en ondelettes. L'estimation de la pente permet encore de repérer la stationnarité d'une série, et de déterminer le type de processus stochastique correspondant aux données d'observation, en adoptant les règles déjà décrites quand on procède à une décomposition de Fourier. Seule différence, les deux bornes inférieures et supérieures élargissent la plage de validité du test. En effet, alors que des biais sont introduits avec une décomposition de Fourier, quand la pente, α, est inférieure à -1 et supérieure à +3, pour une décomposition en ondelettes, ce sont des valeurs de α inférieure à -3 et supérieure à +5 qui signalent la présence de biais dans les calculs.

In[•]:=
```
dwd = StationaryWaveletTransform[data, DaubechiesWavelet[4], 7];
      transformée d'ondelettes stationnaire   ondelette Daubechies
energie = Log[Values[dwd["EnergyFraction"]]];
          log··· valeurs
model = LinearModelFit[energie, x, x] // Normal;
        ajuste modèle linéaire                forme normale
da = model[[2, 1]];
Print["Valeur de la pente = ", da]
imprime
Print["Si la valeur de la pente est inférieure
imprime
   à - 1 ou supérieure à +1 la série est non stationnaire"]
Print[]
imprime
Print["Type de processus"]
imprime
Which[da < -3, Attention biais, -3 < da < -1, DfGn, -1 < da < 1, fGn,
quel
 da == 0, Bm, +1 < da < +3, fBm, +3 < da < +5, fBmI, +5 < da, Attention biais]
```

```
Valeur de la pente = 0.62190205

Si la valeur de la pente est inférieure
  à - 1 ou supérieure à +1 la série est non stationnaire


Type de processus
```

Out[•]=
```
fGn
```

Une fois encore, le test indique une série stationnaire, sans tendance.

Ajuster une chronique par un modèle de régression linéaire

Comme les tests donnent des résultats différents, il est possible de procéder directement, après avoir observé le chronogramme initial, à un ajustement linéaire ou non linéaire, et surtout d'en vérifier la pertinence. Rappelons en effet qu'il est toujours possible de tracer une droite dans un nuage de points. D'où la nécessité de bien étudier la qualité du modèle obtenu.

In[•]:=
```
modeldata = LinearModelFit[data, x, x];
```
ajuste modèle linéaire

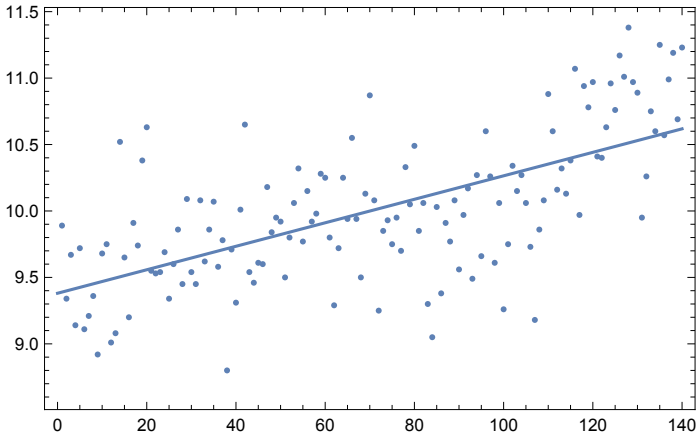
```
Show[ListPlot[data], Plot[modeldata[x], {x, 0, Length[data]}], Frame → True]
```
mon··· tracé de liste tracé de courbes longueur cadre vrai

```
Grid[
```
grille

```
 Transpose[{#, modeldata[#]} &[{"AdjustedRSquared", "AIC", "BIC", "RSquared"}]],
```
transposée

```
 Alignment → Left]
```
alignement gauche

Out[•]=



Out[•]=
```
AdjustedRSquared  0.41243548
AIC               162.15104
BIC               170.97597
RSquared          0.41666257
```

Les résultats obtenus montrent visuellement une tendance, mais avec des points très éloignés de la droite de régression, ce qui est le signe d'un modèle non significatif. Ceci est confirmé par les différents critères, qui tous indiquent un modèle non pertinent pour une série aussi longue. Ajoutons que cette démarche détruit l'organisation temporelle.

Tester la présence de cycles : périodogramme, spectogramme et corrélogramme

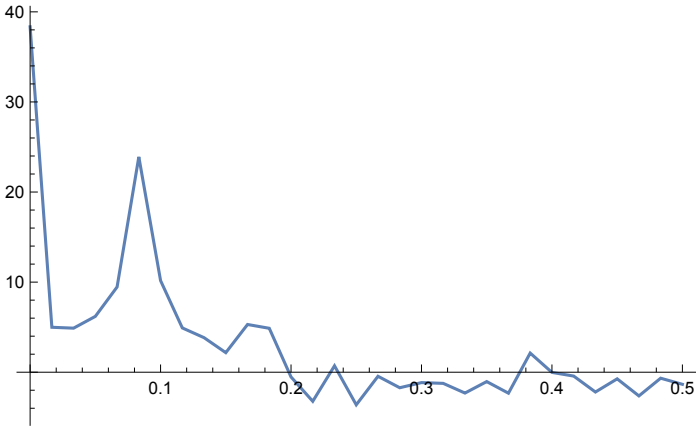
Le repérage de cycles est facile sur les corrélogrammes, qui illustrent soit les autocorrélations soit les autocorrélations partielles. Les cycles sont illustrés par des maximas et des minimas significatifs. Mais il est aussi possible de pratiquer une approche similaire à l'aide du spectre d'énergie ou de puissance déduit de l'analyse de Fourier. Sa représentation visuelle est parfois qualifiée de périodogramme de Schuster. Les cycles y sont figurés par des pics. Pour des cycles plus compliqués, qui ne s'observent qu'à certaines échelles, mais disparaissent à d'autres échelles, le même type de démarche portera sur le scalogramme issu d'une décomposition en ondelettes.

In[1162]:=
```
datamois = ToExpression[Flatten[Import[SystemDialogInput["FileOpen"]]]];
```
convertis en une ··· aplatis importe entrée de dialogue de système

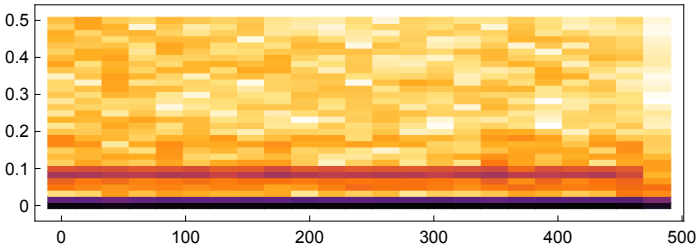
In[1121]:=
```
Periodogram[datamois, 62, PlotRange → All]
```
périodogramme zone de tracé tout

```
Spectrogram[datamois, 62]
```
spectrogramme

Out[1121]=



Out[1122]=



Le programme ci-dessus affiche le periodogramme et le spectogramme des données. On perçoit déjà un rythme cyclique, qui devient encore plus visible sur le corrélogramme calculé puis illustré sur la même série de températures mensuelles.

In[1077]:=
```
DiscretePlot[CorrelationFunction[datamois, i],
```
tracé discret fonction de corrélation

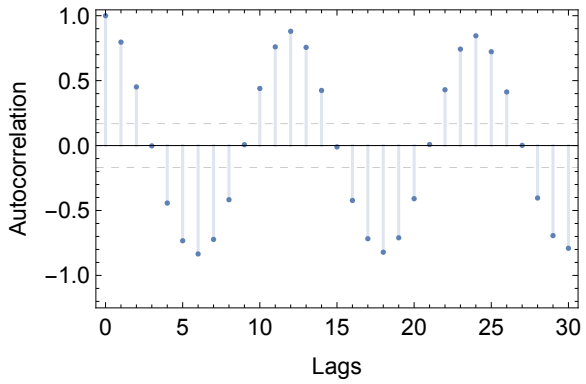
```
 {i, 0, 30}, PlotRange → {-1.25, 1.045}, Frame → True, Axes → True,
```
zone de tracé cadre vrai axes vrai

```
 GridLines → {None, {{2 / Sqrt[ntab], Dashed}, 0, {-2 / Sqrt[ntab], Dashed}}},
```
lignes de grille aucun racine carrée rayé racine carrée rayé

```
 FrameLabel → {"Lags", "Autocorrelation"},
```
titre de cadre

```
 ImageSize → 300, BaseStyle → {FontSize → 12}]
```
taille d'image style de base taille de police de caractères

Out[1077]=



Analyser la persistance avec le variogramme

Pour apprécier la persistance dans le temps d'un phénomène, il convient de construire un variogramme. Ce que réalise le petit programme suivant. Pour la série des températures annuelles on observe une persistance de 5 ans, avec un effet nuage très prononcé.

In[1244]:=
```
Print["Variogramme"]
```
imprime

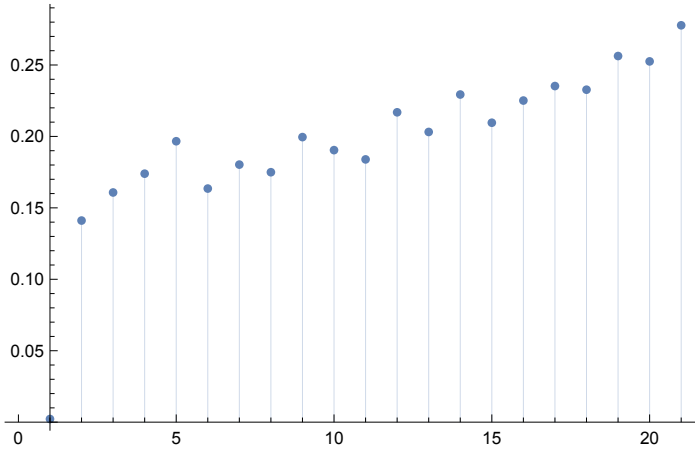
```
vario[data_, lmax_] :=
 ListPlot[{(Variance[data] - CovarianceFunction[data, {0, lmax}])},
```
tracé de liste | variance | fonction de covariance

```
  Filling → Axis, AxesOrigin → {1, 0}, PlotStyle → PointSize[Medium]]
```
remplissage | axe | origine des axes | style de tracé | taille des poi… | taille moyenr

```
vario[
 data,
 20]
```

```
Variogramme
```

Out[1246]=



Choisir un modèle de processus stochastique et en déterminer les paramètres

Après avoir réalisé ces tests, le géographe cherche un modèle de processus stochastique qui est à l'origine de la trajectoire constituée par les données. Trois approches sont possibles. D'abord, le géographe peut se servir d'un modèle reconnu par de nombreux auteurs. S'il a un doute deux solutions s'offrent à lui. La première consiste à tester trois ou quatre modèles, suite à l'examen de tous les traitements antérieurs. Dans notre exemple, le modèle sera plutôt du type stationnaire. De plus les autocorrélations sérielles et partielles, permettent d'envisager un modèle de type AR ou ARMA, voire un modèle ARCH pour tenir compte des tests réalisés sur les variances. Enfin, ne pas oublier que le spectre de la décomposition en ondelettes donne un modèle gaussien fractionnaire (fGn) stationnaire. La seconde solution utilise les ressources de l'apprentissage automatique. Cette approche compare plusieurs modèles et donne les plus significatifs. Enfin, un modèle de processus n'est pas un modèle explicatif, même s'il indique une voie pour envisager les causes sous-jacentes.

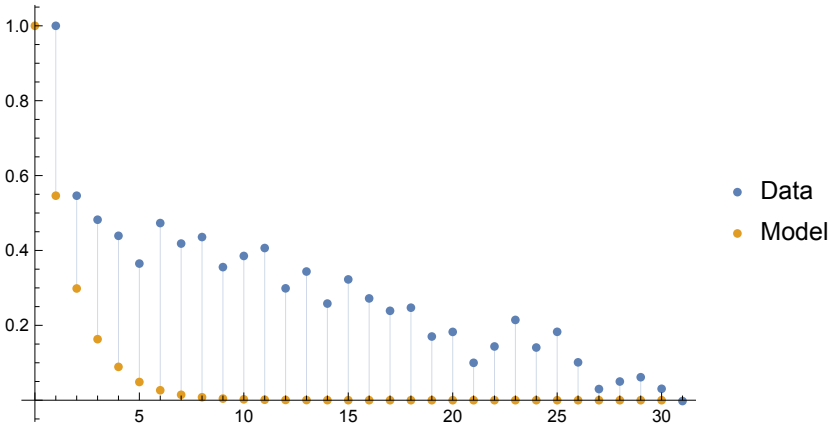
Comme l'autocorrélation sérielle baisse brutalement au pas de temps t + 1, puis de façon plus progressive après, il est possible de retenir comme premier essai le modèle AR d'ordre 1.

```
modelprocessus = EstimatedProcess[data, ARProcess[1]]
                 processus estimée      processus ARP

WeakStationarity[modelprocessus]
stationnarité faible

ListPlot[CorrelationFunction[#, {30}] & /@ {data, modelprocessus},
tracé de liste  fonction de corrélation

  Filling → {1 → {2}}, PlotStyle → PointSize[Medium],
  remplissage          style de tracé  taille des poi···  taille moyenne

  PlotLegends → {"Data", "Model"}]
  légendes de tracé
```

Out[•]= ARProcess[4.5390552, {0.54626949}, 0.21480328]

Out[•]= True

Out[•]=



Ce premier modèle obtenu, il est facile de tester s'il est bien stationnaire avec l'instruction **WeakStationarity[]**. En revanche, les deux corrélogrammes des données et du modèle sont très différents. Ce modèle n'est donc pas robuste. D'autres tests donneraient un résultat identique.

Nous pouvons alors essayer d'utiliser les techniques d'apprentissage automatique avec la fonction **TimeSeriesModelFit[]** :

```
mod = TimeSeriesModelFit[data];
      ajuste modèle de série temporelle

mod["CandidateSelectionTable"]
```

Le modèle proposé par l'apprentissage automatique est le modèle AR d'ordre 3. Nous pouvons en estimer les paramètres et observer avec le programme ci-dessous que les corrélogrammes sont plus ressemblants qu'avec le choix du modèle AR1 :

```
modelprocessus = EstimatedProcess[data, ARProcess[3]]
                 processus estimée      processus ARP

WeakStationarity[modelprocessus]
stationnarité faible

ListPlot[CorrelationFunction[#, {30}] & /@ {data, modelprocessus},
tracé de liste  fonction de corrélation

  Filling → {1 → {2}}, PlotStyle → PointSize[Medium],
  remplissage          style de tracé  taille des poi···  taille moyenne

  PlotLegends → {"Data", "Model"}]
  légendes de tracé
```

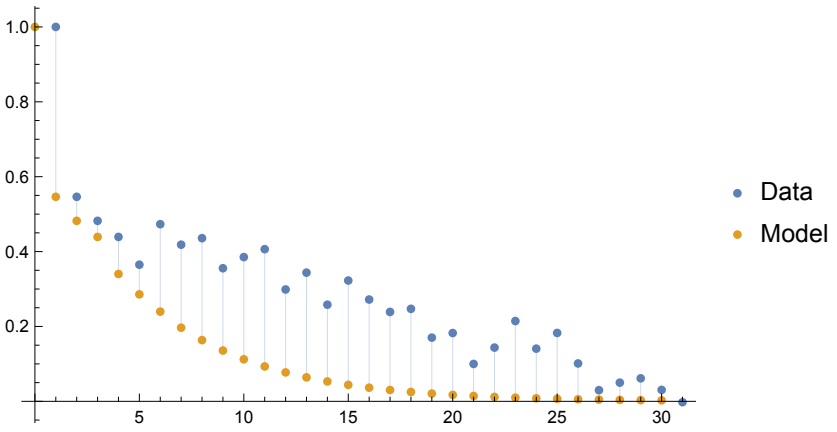
Out[•]=

	Candidate	AIC
1	**ARProcess[3]**	**−218.69866**
2	ARMAProcess[3, 1]	−217.80843
3	ARProcess[2]	−217.26708
4	ARProcess[4]	−216.91049
5	ARMAProcess[2, 1]	−215.03026
6	ARMAProcess[4, 1]	−211.453
7	ARMAProcess[1, 1]	−211.25259
8	ARMAProcess[3, 2]	−211.19003
9	ARMAProcess[2, 2]	−210.98008
10	ARProcess[1]	−209.32457

Out[•]=
```
ARProcess[2.829222, {0.36249753, 0.19908316, 0.15560619}, 0.19523295]
```

Out[•]=
```
True
```

Out[•]=



Out[•]=
```
ARMAProcess[33.17137,
 {-10.892066, 4.2174106, 2.3866932, 1.9721044}, {11.253481}, 0.19062964, {}]
```

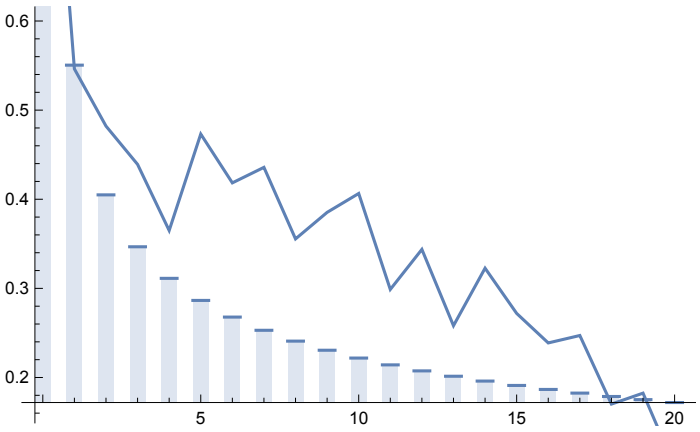
Il est possible de mobiliser un modèle gaussien fractionnaire, mais les résultats sont similaires.

In[1144]:=
```
modelprocessus = EstimatedProcess[data, FractionalGaussianNoiseProcess[m, s, h]]
                 processus estimée      processus de bruit gaussien fractionnaire

WeakStationarity[modelprocessus]
stationnarité faible

dataCov = CorrelationFunction[data, {20}];
          fonction de corrélation

modelCov = CorrelationFunction[modelprocessus, k];
           fonction de corrélation

Show[DiscretePlot[modelCov, {k, 0, 20}, ExtentSize → 1/2],
mon··· tracé discret                    taille d'extension

 ListLinePlot[dataCov, DataRange → {0, 20}]]
 tracé de liste de ligne      plage de données
```

Out[1144]=
```
FractionalGaussianNoiseProcess[10.043764, 0.53317778, 0.81635739]
```

Out[1145]=
```
True
```

Out[1148]=



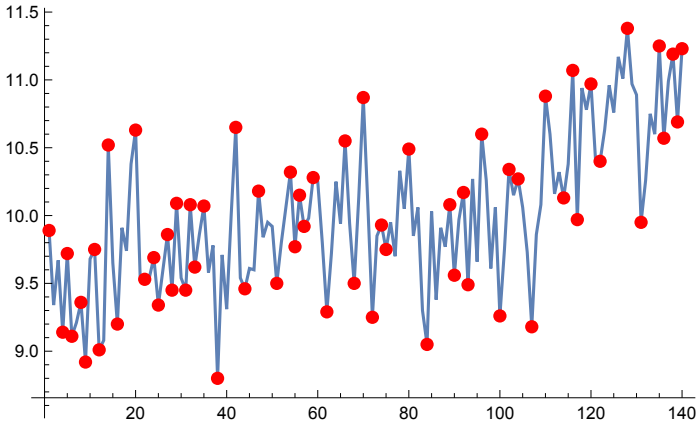
Signalons enfin que l'utilisateur de Mathematica dispose d'autres modèles plus spécifiques, tels les processus de Poisson, les modèles de file d'attente et les processus markoviens.

Repérer les événements exceptionnels, leur intensité et leur répétition

Les séries temporelles affichent des irrégularités plus ou moins significatives, comme un épisode neigeux de grande abondance, une crise financière ou un record du taux de chômage. Dans ce notebook, nous proposons seulement l'examen rapide d'un seul outils : la détection automatique des pics ou des fonds de vallée d'une chronique. L'apport des décompositions en ondelettes, les approches fractales et multifractales seront abordées dans d'autres notebook. L'instruction **FindPeaks**[], détecte les pics, les situe et donne leur intensité. Diverses options permettent de repérer les pics plus ou moins pointus et à différentes échelles. Il est facile de localiser les pics principaux et les pics secondaires. Ce qui favorise donc une première approche multiéchelles. Par simple inversion des données, on effectue un repérage des minimas. Le petit programme 2-10 montre un exemple sur une série test :

In[1141]:=
```
pics = FindPeaks[data];
```
trouve pointes
```
vallees = ({1, -1} #1 &) /@ FindPeaks[-data];
```
trouve pointes
```
ListLinePlot[data, Epilog → {Red, PointSize[0.02], Point[pics], Point[vallees]}]
```
tracé de liste de ligne | épilogue | rouge | taille des points | point | point

Out[1143]=

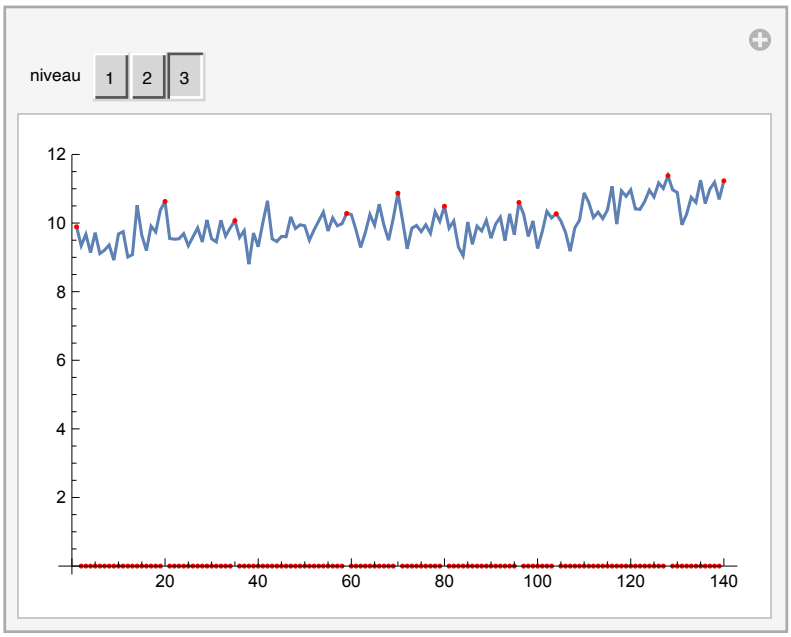


La première instruction repère les pics et donne leur valeur, tandis que la deuxième fait de même pour les vallées. Enfin, la dernière instruction réalise un graphique de la chronique qui fait ressortir les maximas et les minimas.

Une deuxième instruction, **PeakDetect**[], repère la position de chaque pic, leur attribue la valeur 1, et affecte la valeur 0 aux autres données. En sortie, le géographe dispose d'une nouvelle série, une séquence de 1 et de 0, sur laquelle il pratiquera diverses analyses, par exemple pour comprendre la répétition des maximas ou des minimas dans le temps. Les instructions emboîtées ci-dessous permettent de faire varier l'échelle de 1 à 3, et de repérer les pics qui correspondent à ces variations pour la série testList :

In[1160]:=
```
pics = Manipulate[ ListPlot[{data, data PeakDetect[data, niveau]},
```
manipule | tracé de liste | détecte pics
```
Joined -> {True, False}, PlotStyle -> {Automatic, Red}], {niveau, {1, 2, 3}}]
```
joint | vrai | faux | style de tracé | automatique | rouge

Out[1160]=



Il est aussi possible d'utiliser la fonction FindAnomalies[], un outil d'apprentissage automatique que nous aborderons plus en détail dans un autre notebook. En faisant varier le seuil (AcceptanceThreshold), on détecte plus ou moins d'anomalies. Appliquée à la série des températures mensuelles, nous obtenons deux anomalies froides et une chaude.

In[1167]:=
```
anomalies = FindAnomalies[datamois, AcceptanceThreshold → 0.010]
```
trouve anomalies — seuil d'acceptance

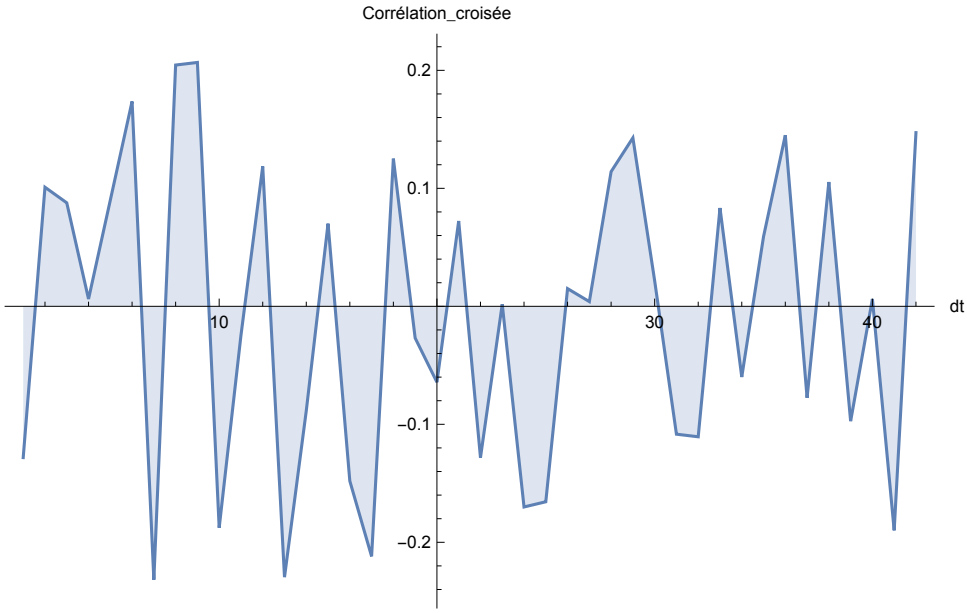
Out[1167]=
```
{2.48, 17.64, 3.24}
```

Première approche des séries temporelles bivariées : corrélation et variogramme croisés

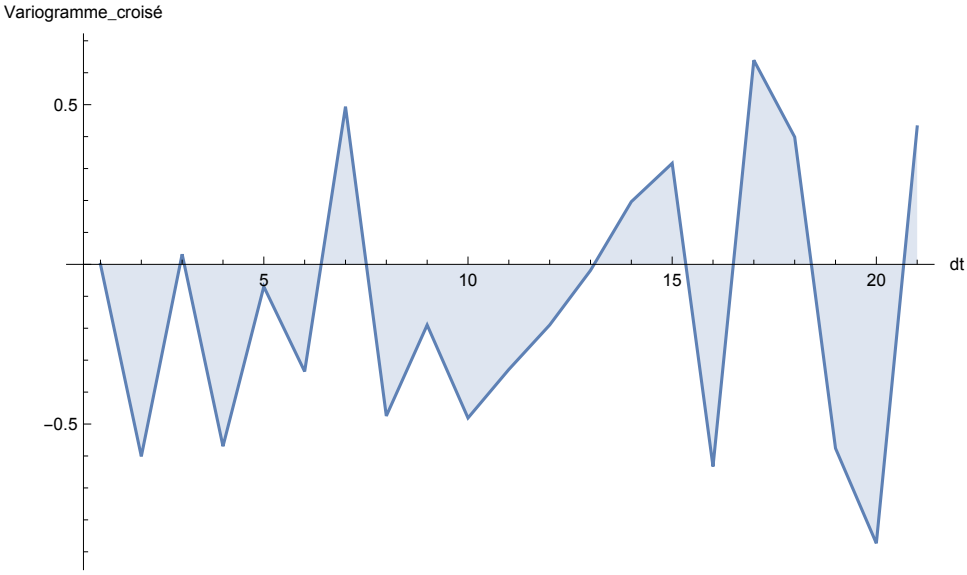
La géographie classique ayant l'ambition d'analyser les relations homme nature, les recherches de ce type doivent établir des liens entre au moins deux séries temporelles. Ainsi, le spécialiste de géographie ou d'économie agricole s'interrogera sur la relation entre l'évolution du temps, mesurée par des degrés jours, et le prix de telle ou telle denrée sur le marché. Pour analyser les relations entre deux chroniques il est possible de calculer le corrélogramme croisé ou le variogramme croisé entre deux séries temporelles. Le programme ci-dessous détermine le corrélogramme et le variogramme croisés pour deux séries de données aléatoires préalablement créées, d et d1 :

In[1219]:=
```
data1 = RandomInteger[5, 80];
```
entier aléatoire
```
data2 = RandomInteger[12, 80];
```
entier aléatoire
```
n1 = Length[data1]; n2 = Floor[n1/4];
```
longueur — entier inférieur
```
crosscovariance[y_, x_, dtemps_] := Return[Covariance[y, RotateLeft[x, dtemps]]/
```
reviens — covariance — tourne à gauche
```
    Sqrt[Variance[y]*Variance[x]]]
```
racin·· — variance — variance
```
cross1 = Table[crosscovariance[data1, data2, n1], {n1, 0, n2}];
```
table
```
cross2 = Table[crosscovariance[data2, data1, n1], {n1, 0, n2}];
```
table
```
crosstout = Flatten[Append[cross1, cross2]];
```
aplatis — appose
```
ListLinePlot[{crosstout}, ImageSize -> {500, 400}, AxesOrigin -> {n2, 0}, Filling
```
tracé de liste de ligne — taille d'image — origine des axes — remplissa
```
  AxesLabel -> {"dt", "Corrélation_croisée"}, PlotRange -> All, PlotStyle -> Poir
```
titre d'axe — zone de tracé — tout — style de tracé — taille
```
covar = Covariance[data1, data2];
```
covariance
```
crossvariogramme[y_, x_, dtemps_] := Return[Covariance[y, RotateLeft[x, dtemps]]]
```
reviens — covariance — tourne à gauche
```
crossvario1 = Table[crossvariogramme[data1, data2, n1], {n1, 0, n2}];
```
table
```
crossvario2 = Table[crossvariogramme[data2, data1, n1], {n1, 0, n2}];
```
table
```
crossvariotout = Flatten[covar - (crossvario1 + crossvario2)/2];
```
aplatis
```
Print[]
```
imprime
```
ListLinePlot[{d4}, ImageSize -> {500, 400}, Filling -> Axis,
```
tracé de liste de ligne — taille d'image — remplissage — axe
```
  AxesLabel -> {"dt", "Variogramme_croisé"}, PlotRange -> All, PlotStyle -> Point
```
titre d'axe — zone de tracé — tout — style de tracé — taille d

Out[1226]=



Out[1233]=



De la même façon, il est relativement facile de calculer un co-spectre entre deux séries à partir d'une décomposition de Fourier ou d'ondelettes. Cela sera exposé dans un autre notebook traitant des ondelettes.

Conclusion

Comme dans les autres notebook, il est possible d'enrichir et de généraliser ces petits programmes

en mobilisant les options des différentes fonctions. Par exemple choisir un nombre d'intervalles pour les autocorrélations, avec la fonction :

```
nb = Input["nombre d'intervalles"]
```
entrée

Un simple fenêtre apparaît sur l'écran, et il suffit de taper le nombre souhaité qui est sauvegardé dans nb

Ouvrage recommandé :

Mohamed Boutahar et Manuela Royer-Carenzi, 2019, *Méthodes en séries temporelles et applications avec R*, ellipses.