Université de Poitiers
Département de Mathématiques

3M22–Séries temporelles
Master MMAS 2, année 2009–10

SÉRIES TEMPORELLES. — TRAVAUX PRATIQUES

1. **Un premier exemple**

On considère la série temporelle suivante :

t_i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y_i	7.5	4.4	3.3	7.6	3.9	2.4	6.9	4.5	2.7	8.2	4.1	3.0	7.5	3.5	2.8

1.1. Représentation graphique de la série

[1] Dans un premier temps nous saisissons la série. Avec Scilab cela peut se faire par

```
// code Scilab
```

```
Y = [7.5, 4.4, 3.3, 7.6, 3.9, 2.4, 6.9, .. // on continue..
    4.5, 2.7, 8.2, 4.1, 3.0, 7.5, 3.5, 2.8]; // vecteur ligne
Y = Y'; size(Y)
```

donc sous la forme d'un vecteur ligne, qu'on transpose ensuite pour obtenir un vecteur colonne (et on vérifie au passage que les dimensions de `Y` sont celles qu'on attend).

[2] La représentation graphique peut se faire en définissant le vecteur des abscisses comme étant le temps, et en utilisant `plot`

```
T = [1:15]; T = T';
clf(); plot(T, Y, "o-");
```

bien qu'ici

```
clf(); plot(Y, "o-");
```

aurait suffit puisqu'en présence d'un unique vecteur ligne celui-ci est considéré comme un vecteur d'ordonnées, les abscisses étant alors les indices de ce vecteur (mais il vaut mieux s'habituer dès le début à une situation plus générale.

[3] Quel modèle proposeriez-vous pour cette série ? Donnez des justifications.

1.2. Estimation

[1] Il est très facile de calculer la moyenne observée de la série

```
n = size(Y, 1); Ymean = sum(Y)/n;// ou Ymean = mean(Y);
```

et il peut sembler que la série s'écrive comme la somme de sa moyenne

```
F = Tmean*ones(n, 1),     F <- c(rep(Tmean, times=n))
```

d'une composante saisonnière et de bruit (composante irrégulière).

[2] Quelle période p donnerait-on à la composante saisonnière ? (on conservera cette valeur par la suite `p = ...;`. Proposer une méthode pour estimer, malgré le bruit, les différentes valeurs de la composante saisonnière. Justifier cette estimation lorsque le bruit est un bruit blanc.

[3] Mettre en œuvre cette estimation « à la main » si cela paraît plus facile (`S[1] = ...;` etc.).

[4] Calculer la série des fluctuation irrégulières :

```
E = Y-F-S;
```

[5] Afficher un tableau synthétique

```
RESUME = [Y,Y,S,E]';
```

de la décomposition $y_t = f_t + s_t + e_t$, et représenter sur le même graphique les 4 séries (on pourra utiliser des options de couleurs, chaque couleur correspondant à un type de série).

[6] Que pensez-vous de la composante irrégulière ? Présente-t-elle des corrélations ? (Il n'est pas demandé de les estimer, mais cela peut être envisagé pour plus tard.)

1.3. Ajustement affine

[1] On recherche une tendance affine. Calculer avec `Scilab` les coefficients $\hat{a}$ et $\hat{b}$ de la régression.

[2] Est-il envisageable de réaliser un test permettant de choisir entre ce nouveau modèle et le précédent ? Comment se formulerait-il ?

[3] Déduire avec la droite de régression les nouvelles composantes saisonnière s' et irrégulière e'.

[4] Comparer les tableaux numériques et les graphiques obtenus.

2. **De manière plus générale**

Nous souhaitons maintenant écrire une procédure qui automatisera ce qui a été fait précédemment. La seule quantité que nous nous dispenserons d'estimer est une éventuelle période qui pourra apparaître comme paramètre de la procédure, une valeur 0 ou 1 signifiant l'absence de période et l'initialisation de la série saisonnière à 0.

2.1. Ajustement par une fonction constante, affine, polynomiale

La régression peut généralement se présenter ainsi : on a une variable explicative t (ici le temps) et une variable expliquée y (ici les valeurs de la série). Nous désirons ajuster un modèle de la forme

$$y(t) = a_0\phi_0(t) + \cdots + a_d\phi_d(t)$$

à la série observée, où $\phi_i : \mathbf{R} \to \mathbf{R}$, $0 \leq i \leq d$ sont des fonctions données.

Exemples. — *a*) Ajustement d'une constante, $d = 0$ et $\phi_0(t) = 1$.

b) Ajustement d'une fonction affine, $d = 1$, $\phi_0(t) = 1$, $\phi_1(t) = t$.

c) Ajustement d'une fonction polynomiale, $\phi_0(t) = 1$, ..., $\phi_i(t) = t^i$, ..., $\phi_d(t) = t^d$.

d) etc.

Pour cela, on minimise la fonction

$$F(a_0, \ldots, a_d) = \sum_{j=1}^{n} \bigl(a_0\phi_0(t_j) + \cdots + a_d\phi_d(t_j) - y_j\bigr)^2$$

qui est « parabolique », ou quadratique positive, et admet donc une valeur minimale atteinte en les solutions du système linéaire

$$\frac{\partial F}{\partial a_j}(a_0, \ldots, a_d) = 0, \qquad 0 \leq i \leq d$$

c'est-à-dire

$$\sum_{j=1}^{n} \phi_i(t_j) \times \big(a_0\phi_0(t_j) + \cdots + a_d\phi_d(t_j) - y_j\big) = 0, \qquad 0 \leq i \leq d,$$

ou encore

$$\sum_{j=1}^{n} \phi_i(t_j) \times \big(y(t_j) - y_j\big) = 0, \qquad 0 \leq i \leq d.$$

Ceci conduit à résoudre un premier système linéaire pour obtenir le vecteur $(y(t_1), \ldots, y(t_n))$, puis un second pour obtenir $(a_0, \ldots, a_d)$.

[1] Vérifier que ces explications sont cohérentes et les détailler. Est-il possible qu'il n'existe pas de solution au problème de la régression ? Dans le cas d'une régression polynomiale, se peut-il qu'il existe plusieurs solutions ? Comment s'assurer d'une solution unique raisonnable ?

[2] Préparer la procédure pour la suite :

```
// T est le temps, Y la s\'erie temporelle,
// p la p\'eriode, d le degr\'e de la r\'egression polynomiale

function TSanalysis(T, Y, p, d)
    // local n F S E a ...
    n = size(Y, 1);
    // tendance polynomiale
    a = zeros(d+1, 1);
    if d == 0 then a = mean(Y);
    elseif d == 1 then
        ... // coefficients de y=a(2)*t+a(1)
    else
        ... // d >= 2
    end
    // Calcul de la tendance F
    F = zeros(n, 1);
    ...
    // composante saisonni\`ere
    S = zeros(n, 1);
    if p > 1 then ... // (and p < n ?) estimer les coefficients
    end
    // composante irr\'eguli\`ere
    E = Y-F-S;
    // Synth\`ese des r\'esultats (tableaux, graphiques)
    ...
endfunction
```

Il est évident que les points de suspension correspondent à des morceaux de programmes à écrire et ne sont pas par eux-même exécutables !

[3] Se documenter sur l'inversion de système linéaire avec SCILAB (rapidement, pour résoudre $A \times x = b$, on exécute `x = A\b` ou encore `x = linsolve(A, -b)`).

[4] Compléter le programme et le tester sur les données du début.

3. Analyse élémentaire d'autres séries temporelles

On trouvera à

`http://www-math.univ-poitiers.fr/~phan/masterMMAS/documents/data/`

différents jeux de données.

[1] Pour trois d'entre eux, on présentera une analyse succinte de la série temporelle correspondante. Précisons que la commande `Y = read("monfichier.dat")` permet, ou devrait permettre de placer dans le vecteur (ligne) le contenu du fichier.

[2] Envisager des améliorations pour le programme (autres régressions, modèles multiplicatifs, estimation des covariances, tests statistiques).

Université de Poitiers
Département de Mathématiques

3M22–Séries temporelles
Master MMAS 2, année 2009–10

SÉRIES TEMPORELLES. — TRAVAUX PRATIQUES, CORRECTIONS

1. **Un premier exemple**

1.1. Représentation graphique de la série

[1] [2] Le code Scilab est élémentaire, nous le donnons pour mémoire.

```
// code Scilab

// la s\'erie
Y = [7.5, 4.4, 3.3, 7.6, 3.9, 2.4, 6.9, ..
    4.5, 2.7, 8.2, 4.1, 3.0, 7.5, 3.5, 2.8]';// vecteur colonne
n = size(Y, 1);
// le temps
T = [1:n]';
// un premier trac\'e
clf(); plot(T, Y, "o-");
halt("***taper return pour continuer***");
```

1.2. Estimation

[1] [2] [3] [4] Nous traitons les questions à la suite.

```
// \`a premi\`ere vue, une tendance constante semble raisonnable

Ymean = mean(Y);// Ymean = sum(Y)/n; d\'ej\`a impl\'ement\'e
F = Ymean*ones(n, 1);

// une composante saisonni\`ere de p\'eriode 3 semble indiqu\'ee

p = 3;

// S(i) est estim\'e par la moyenne des S(i+j*p), ce qui ici donnerait
// S(1) = ((Y(1)-F(1)+...+(Y(12)-F(12)))/5=(Y(1)+...+Y(12))/5-F(1);
// ...
// S(3) = ((Y(3)-F(3)+...+(Y(15)-F(15)))/5=(Y(3)+...+Y(15))/5-F(3);
// mais on peut l'\'ecrire de mani\`ere plus g\'en\'erale :

S = zeros(n, 1);
for i = 1:p
    for j = 0:ceil(n/p)-1
        S(i) = S(i)+(Y(i+j*p)-F(i+j*p));
        if i+(j+1)*p > n then break; end
    end
    S(i) = S(i)/(j+1);
end
```

```
for i = p+1:n
    S(i) = S(modulo(i-1, p)+1);
end

// sans oublier \`a la fin de r\'ep\'eter les valeurs par p\'eriodicit\'e.

// Le ``bruit'' restant dans la d\'ecomposition est

E = Y-F-S;

// enfin, on affiche grossi\`erement la d\'ecomposition

RESUME = [T, Y, S, E]'

// et on trace les diff\'erentes s\'eries sur un m\^eme graphique

clf(); plot(T, [Y, S, E], "o-");
halt("***taper return pour continuer***");

// ou sur des sous-graphiques s\'epar\'es.

clf();
subplot(3, 1, 1); plot(T, [Y, F], "o-");
subplot(3, 1, 2); plot(T, S, "o-");
subplot(3, 1, 3); plot(T, E, "o-");
halt("***taper return pour continuer***");
```

Pour obtenir le résumé :

```
1.      2.      3.      4.      5.      6.      7.      8.      9.
7.5     4.4     3.3     7.6     3.9     2.4     6.9     4.5     2.7
2.72  - 0.74  - 1.98    2.72  - 0.74  - 1.98    2.72  - 0.74  - 1.98
- 0.04    0.32    0.46    0.06  - 0.18  - 0.44  - 0.64    0.42  - 0.14

10.     11.     12.     13.     14.     15.
8.2     4.1     3.      7.5     3.5     2.8
2.72  - 0.74  - 1.98    2.72  - 0.74  - 1.98
0.66    0.02    0.16  - 0.04  - 0.58  - 0.04
```

[5] Il n'est pas évident de dégager une structure avec si peu de bruit. On peut pourtant essayer (noter que par construction le bruit est de moyenne estimée nulle). En supposant le bruit stationnaire, on estimera les covariances par

$$\gamma(0) \approx \frac{1}{n}\sum_{j=1}^{n} e(j)^2, \qquad \gamma(1) \approx \frac{1}{n-1}\sum_{j=1}^{n-1} e(j)e(j+1), \ldots \quad \gamma(i) \approx \frac{1}{n-i}\sum_{j=1}^{n-i} e(j)e(j+i), \ldots$$

et pour finir $\gamma(n-1) \approx e(1)e(n)$. Nous ne conserverons que les corrélations estimées $\varrho(i) = \gamma(i)/\gamma(0)$. Ce qui donne avec SCILAB

```
// Analysons le bruit en le supposant stationnaire centr\'e.
// Nous n'avons gu\`ere le choix pour estimer la fonction
// de covariance. Seuls les tous premiers termes peuvent avoir
// un peu de sens.
```

```
rho = zeros(n, 1);
for i = 0:n-1
    for j = 1:n-i
        rho(i+1) = rho(i+1)+E(j)*E(j+i);
    end
    rho(i+1) = rho(i+1)/(n-i);
end
rho = rho/rho(1);
rho'
```

les corrélations étant définies comme $\varrho(i) = \gamma(i)/\gamma(0)$.

i	1	2	3	4	5
ϱ	0.0814549	0.1179855	−0.5094775	−0.6927161	−0.1493852
i	6	7	8	9	10
ϱ	−0.1290984	0.5632684	0.5747951	0.2218238	0.0067623
i	11	12	13	14	
ϱ	−0.5540471	−0.5184426	0.0399590	0.0122951	

Ces estimations ont peut-être un peu de sens pour i petit devant n, mais elles sont faites néanmoins avec une hypothèse de stationnarité (au moins).

1.3. Ajustement affine

Nous rappelons que les coefficients de la droite de régression $y = \hat{a} \times t + \hat{b}$ sont donnés par

$$\hat{a} = \operatorname{cov}(t, y)/\operatorname{var}(t) \qquad \text{et} \qquad \hat{b} = \bar{y} - \operatorname{cov}(t, y)/\operatorname{var}(t) \times \bar{t}$$

```
// proc\'edons \`a la r\'egression lin\'eaire
// en faisant les calculs pas \`a pas et assez na\"\i vement

Tmean = mean(T); // Ymean = mean(Y); d\'ej\`a calcul\'e

TYcov = mean(T.*Y)-Tmean*Ymean; // notons la multiplication terme \`a terme
Tvar = mean(T.*T)-Tmean^2;// idem
ahat = TYcov/Tvar; bhat = Ymean-ahat*Tmean;

// nous red\'efinissons la tendance

Fprime = ahat*T+bhat*ones(n, 1);

// et tra\c cons un premier graphique

clf(); plot(T, [Y, Fprime], "o-");
halt("***taper return pour continuer***");

// Puis, nous r\'ep\'etons les \'etapes pr\'ec\'edentes.

Sprime = zeros(n, 1);

for i = 1:p
    for j = 0:ceil(n/p)-1
        Sprime(i) = Sprime(i)+(Y(i+j*p)-Fprime(i+j*p));
        if i+(j+1)*p > n then break; end
    end
    Sprime(i) = Sprime(i)/(j+1);
end
```

```
for i = p+1:n
    Sprime(i) = Sprime(modulo(i-1, p)+1);
end

Eprime = Y-Fprime-Sprime;

RESUME = [T, Y, Sprime, Eprime]'

clf(); plot(T, [Y, S, E, Sprime, Eprime], "o-");
halt("***taper return pour continuer***");

clf();
subplot(3, 1, 1); plot(T, [Y, F, Fprime], "o-");
subplot(3, 1, 2); plot(T, [S, Sprime], "o-");
subplot(3, 1, 3); plot(T, [E, Eprime], "o-");
halt("***taper return pour continuer***");

rho = zeros(n, 1);
for i = 0:n-1
    for j = 1:n-i
        rho(i+1) = rho(i+1)+Eprime(j)*Eprime(j+i);
    end
    rho(i+1) = rho(i+1)/(n-i);
end
rho = rho/rho(1);
rho'
```

2. **De manière plus générale**

Donnons quelques précisions sur la régression polynomiale. Soient $\phi_0, \ldots, \phi_d$ une famille libre de fonctions, $(t_1, y_1), \ldots, (t_n, y_n)$ une famille de points. On cherche $a_0, \ldots, a_d \in \mathbf{R}$ tels que $y(t) = a_0\phi_0(t) + \cdots + a_d\phi_d(t)$ minimise

$$F(a_0, \ldots, a_d) = \sum_{j=1}^{n} \big(y_j - y(t_j)\big)^2.$$

Une condition nécessaire d'extremum est ici l'annulation de la différentielle de F : pour $i = 0, \ldots, d$,

$$\sum_{j=1}^{n} \phi_i(t_j)\big(y_j - y(t_j)\big) = 0 \qquad \text{soit} \qquad \sum_{j=1}^{n} \phi_i(t_j) \times y(t_j) = \sum_{j=1}^{n} \phi_i(t_j) \times y_j$$

Notons Φ la matrice dont les $d+1$ colonnes correspondent aux fonctions $(\phi_i)_{i=0}^{d}$:

$$\Phi = \begin{pmatrix} \phi_0(t_1) & \ldots & \phi_i(t_1) & \ldots & \phi_d(t_1) \\ \vdots & & \vdots & & \vdots \\ \phi_0(t_j) & \ldots & \phi_i(t_j) & \ldots & \phi_d(t_j) \\ \vdots & & \vdots & & \vdots \\ \phi_0(t_n) & \ldots & \phi_i(t_n) & \ldots & \phi_d(t_n) \end{pmatrix}$$

qui est une matrice de dimensions $(n, d+1)$. Le système précédent s'écrit alors

$$\Phi' \times \begin{pmatrix} y(t_1) \\ \vdots \\ y(t_j) \\ \vdots \\ y(t_n) \end{pmatrix} = \Phi' \times \begin{pmatrix} y_1 \\ \vdots \\ y_j \\ \vdots \\ y_n \end{pmatrix} = \Phi' \times Y.$$

avec $\Phi' = {}^t\Phi$ la matrice transposée de Φ. Comme on a

$$\Phi \times a = \Phi \times \begin{pmatrix} a_0 \\ \vdots \\ a_i \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} a_0\phi_0(t_1) + \cdots + a_i\phi_i(t_1) + \cdots + a_d\phi_d(t_1) \\ \vdots \\ a_0\phi_0(t_j) + \cdots + a_i\phi_i(t_j) + \cdots + a_d\phi_d(t_j) \\ \vdots \\ a_0\phi_0(t_n) + \cdots + a_i\phi_i(t_n) + \cdots + a_d\phi_d(t_n) \end{pmatrix} = \begin{pmatrix} y(t_1) \\ \vdots \\ y(t_j) \\ \vdots \\ y(t_n) \end{pmatrix}$$

le système s'écrit finalement

$$\Phi' \times \Phi \times a = \Phi' \times \Phi \times \begin{pmatrix} a_0 \\ \vdots \\ a_i \\ \vdots \\ a_d \end{pmatrix} = \Phi' \times \begin{pmatrix} y_1 \\ \vdots \\ y_j \\ \vdots \\ y_n \end{pmatrix} = \Phi' \times Y.$$

Ce système peut paraître simple, il l'est, mais son inversion pose d'assez gros problèmes numériques lorsque de grands écarts se creusent dans la matrice Φ. Lorsque la régression est polynomiale, les matrices Φ' et Φ sont des matrices de Vandermonde. Les difficultés d'inversion de telles matrices sont bien connues :

> *Remember that Vandermonde systems can be quite ill-conditioned. In such a case,* no *numerical method is going to gives a very accurate answer.* [2]

Nous ne pourrons donc utiliser l'interpolation polynomiale que pour des degrés très faibles (de 0 à 4 au mieux). C'est décevant. D'autres approches sans calcul des coefficients $a_0, \ldots, a_d$ devraient pouvoir être menées...

```
// T est le temps, Y la s\'erie temporelle,
// p la p\'eriode, d le degr\'e de la r\'egression polynomiale

function [RESUME, a, rho] = TSanalysis(T, Y, p, d)
    // local n powerT tPhi i j F S E
    n = size(Y, 1);
    //
    // tendance polynomiale
    //
    d = min(d, n-1); // le param\`etre d n'est pas prot\'eg\'e !!!
    a = zeros(d+1, 1);
    if d == 0 then
        a(1) = mean(Y);
        F = a(1)*ones(n, 1); // tendance
    elseif d == 1 then
        a(2) = (mean(T.*Y)-mean(T)*mean(Y))/(mean(T.*T)-mean(T)^2);
        a(1) = mean(Y)-a(2)*mean(T);
```

```
    F = a(1)*ones(n, 1)+a(2)*T; // tendance
else // d >= 2
    powerT = ones(n,1); tPhi = [];
    for i = 1:d+1; tPhi = [tPhi; powerT']; powerT = powerT.*T; end
    a = linsolve(tPhi*tPhi', -tPhi*Y);
    F = linsolve(tPhi, -tPhi*Y);// = tPhi'*a; tendance
end
//
// composante saisonni\`ere
//
S = zeros(n, 1);
if p >= n then S = Y-F;
elseif p > 1 then // estimer les coefficients
    for i = 1:p
        for j = 0:ceil(n/p)-1
            S(i) = S(i)+(Y(i+j*p)-F(i+j*p));
            if i+(j+1)*p > n then break; end
        end
        S(i) = S(i)/(j+1);
    end
    for i = p+1:n
        S(i) = S(modulo(i-1,p)+1);
    end
else // p <= 1 : ne rien faire, la composante saisonni\`ere est nulle
end
//
// composante irr\'eguli\`ere
//
E = Y-F-S;
//
// Synth\`ese des r\'esultats (tableaux, graphiques)
//
RESUME = [T, Y, F, S, E];
clf(); plot(T, [Y, F, S, E], "o-");
xtitle("Decomposition de la serie temporelle");
legend(["serie temporelle"; "tendance";
    "composante saisonniere"; "composante irreguliere"]);
halt("***taper return pour continuer***");
clf();
subplot(3, 1, 1); plot(T, [Y, F], "o-");
xtitle("Serie temporelle et tendance");
legend(["serie temporelle"; "tendance"]);
subplot(3, 1, 2); plot(T, [S, E], "o-");
xtitle("Composante saisonniere et bruit");
legend(["composante saisonniere"; "composante irreguliere"]);
//
// \'Etude sommaire du bruit
// Se rappeller que seules les premi\`eres corr\'elations
// ont peut-\^etre un peu de pertinence !
//
```

```
    rho = zeros(n, 1);
    for i = 0:n-1
        for j = 1:n-i
            rho(i+1) = rho(i+1)+E(j)*E(j+i);
        end
        rho(i+1) = rho(i+1)/(n-i);
    end
    rho = rho/rho(1);
    subplot(3, 1, 3); bar(T, rho);
    xtitle("Correlation estimee du bruit");
endfunction;

[RESUME, a, rho] = TSanalysis(T,Y,3,1);
```

3. **Avec R**

Ce qui suit est une tentative de traduction ligne à ligne du code SCILAB précédent en R. Le langage R a une structure proche du C comme nous allons le voir en chemin. Le code aura été saisi à l'aide d'un éditeur quelconque dans le fichier `3m22tp.R` et exécuté depuis la console R avec `source("3m22tp.R")`.

```
# code R

# la s\'erie
Y <- c(7.5, 4.4, 3.3, 7.6, 3.9, 2.4, 6.9,
    4.5, 2.7, 8.2, 4.1, 3.0, 7.5, 3.5, 2.8);# vecteur \'el\'ementaire
n <- length(Y);
```

Nous voyons ici une première différence notable entre R et de nombreux autres langages : l'affectation `Y <-` ⟨*objet*⟩ ou ⟨*objet*⟩ `-> Y`. Ceci est radicalement différent du C ou de SCILAB et même de SAS (=), mais n'est pas sans rappeler les commandes de redirection du shell (bourne ou bash). L'instruction `c` (pour *combine*) permet de former un vecteur à partir d'une liste qui possède une longueur mais pas réellement de dimensions ; en particulier, ça n'est ni vraiment un vecteur ligne, ni vraiment un vecteur colonne (contrairement à SCILAB).

```
# le temps
T <- c(1:n);
# un premier trac\'e
#clf(); ?
plot(T, Y, type = "o");# "o" pour "overplotted"
cat("***taper return pour continuer***"); readLines(n = 1) -> pause;
```

Il ne semble pas exister de commande du type *clear figure* ou *clear screen* en R. La commande `plot` ressemble jusqu'à présent à son homonyme SCILAB ou plus précisément MATLAB. L'option `"o"` demande que les points successifs soient marqués d'un cercle et joints par des segments. De simples cercles sont obtenus avec l'option `"p"` (pour `"point"` ou `"punkt"` ?) et le simples segments avec `"l"`. Nous n'avons pas trouvé de commande `halt` ou `pause`. La solution proposée dans un forum de discussion consiste à afficher une invite et entrer une ligne (n'importe quoi terminé par « entrée ») qui sera stockée dans un objet dépourvu d'intérêt et de sens particulier, ici `pause`.

```
# \`a premi\`ere vue, une tendance constante semble raisonnable
```

```
Ymean <- mean(Y);# Ymean <- sum(Y)/n; d\'ej\`a impl\'ement\'e
F <- rep(Ymean, times = n);
```

L'instruction `rep` permet de former un vecteur par répétition d'une expression, ici le nombre `Ymean`.

```
# une composante saisonni\`ere de p\'eriode 3 semble indiqu\'ee

p <- 3;

# S[i] est estim\'e par la moyenne des S[i+j*p], ce qui ici donnerait
# S[1] <- ((Y[1]-F[1])+...+(Y[12]-F[12]))/5 = (Y[1]+...+Y[12])/5-F[1];
# ...
# S[3] <- ((Y[3]-F[3]+...+(Y[15]-F[15]))/5 = (Y[3]+...+Y[15])/5-F[3];
# mais on peut l'\'ecrire de mani\`ere plus g\'en\'erale :
```

Notons que les éléments d'un vecteur sont repérés par des indices accessibles via les crochets droits `[.]` ce qui est nettement moins original que les parenthèses `(.)` de SCILAB. La structure des boucles

$$\texttt{for (}\langle\textit{variable}\rangle\texttt{ in }\langle\textit{liste}\rangle\texttt{) }\langle\textit{expression}\rangle$$

ou

$$\texttt{for (}\langle\textit{variable}\rangle\texttt{ in }\langle\textit{liste}\rangle\texttt{) \{}\langle\textit{expression}_1\rangle\texttt{; ...; }\langle\textit{expression}_n\rangle\texttt{\}}$$

rappelle le C avec la liberté d'utiliser une liste quelconque pour l'indexation.

```
S <- rep(0, times = n);
for (i in 1:p) {
    for (j in 0:(ceiling(n/p)-1)) {
        S[i] <- S[i]+(Y[i+j*p]-F[i+j*p]);
        if (i+(j+1)*p > n) break;
    }
    S[i] <- S[i]/(j+1);
}

for (i in (p+1):n) {S[i] <- S[(i-1)%%p+1]}# noter le modulo

# sans oublier \`a la fin de r\'ep\'eter les valeurs par p\'eriodicit\'e.
```

Il est important de noter que l'opérateur « : » est prioritaire par rapport aux opérateurs algébriques usuels. Il est donc nécessaire d'ajouter des parenthèses si on ne veut pas qu'une somme ou un produit affecte la liste en son entier plutôt qu'une seule des bornes seulement.

```
# Le ``bruit'' restant dans la d\'ecomposition est

E <- Y-F-S;

# enfin, on affiche grossi\`erement la d\'ecomposition

RESUME <- list(T, Y, S, E) # rbind(T, Y, S, E)
```

La commande `rbind` colle les différents objets en ligne (*row*). Il est préférable d'utiliser `list` ou `data.frame` qui permet d'empaqueter des données diverses sous une forme réellement structurée pour pouvoir ainsi facilement récupérer ses constituants.

```
# et on trace les diff\'erentes s\'eries sur un m\^eme graphique

#clf(); ?
plot(c(T, T, T), c(Y, S, E), type = "p", xlab = "T", ylab = "Y, S, E");
lines(T, Y); lines(T, S); lines(T, E);
cat("***taper return pour continuer***"); readLines(n = 1) -> pause;
```

La commande plot de R apparaît ici comme moins souple que son homologogue SCILAB/-MATLAB : on trace un nuage de points quitte à répéter les abscisses données par T. Il reste encore à joindre les points des bons sous-nuages et veiller à personnaliser la labelisation des axes.

```
# ou sur des sous-graphiques s\'epar\'es.

#clf(); ?
split.screen(c(3, 1));
screen(1); plot(T, Y, type = "o"); lines(T, F, type = "o");
screen(2); plot(T, S, type = "o");
screen(3); plot(T, E, type = "o");
close.screen(all = TRUE);
cat("***taper return pour continuer***"); readLines(n = 1) -> pause;
```

Nous avons ci-dessus enfin des commandes simples et compréhensibles ! L'écran est divisé en un tableau de dimensions $(3, 1)$ de sous-écrans numérotés par lecture de haut en bas des colonnes successives. Il n'y a ensuite que peu de choses nouvelles mis à part la multiplication terme à terme qui est privilégiée avec R plutôt que la multiplication matricielle comme en SCILAB.

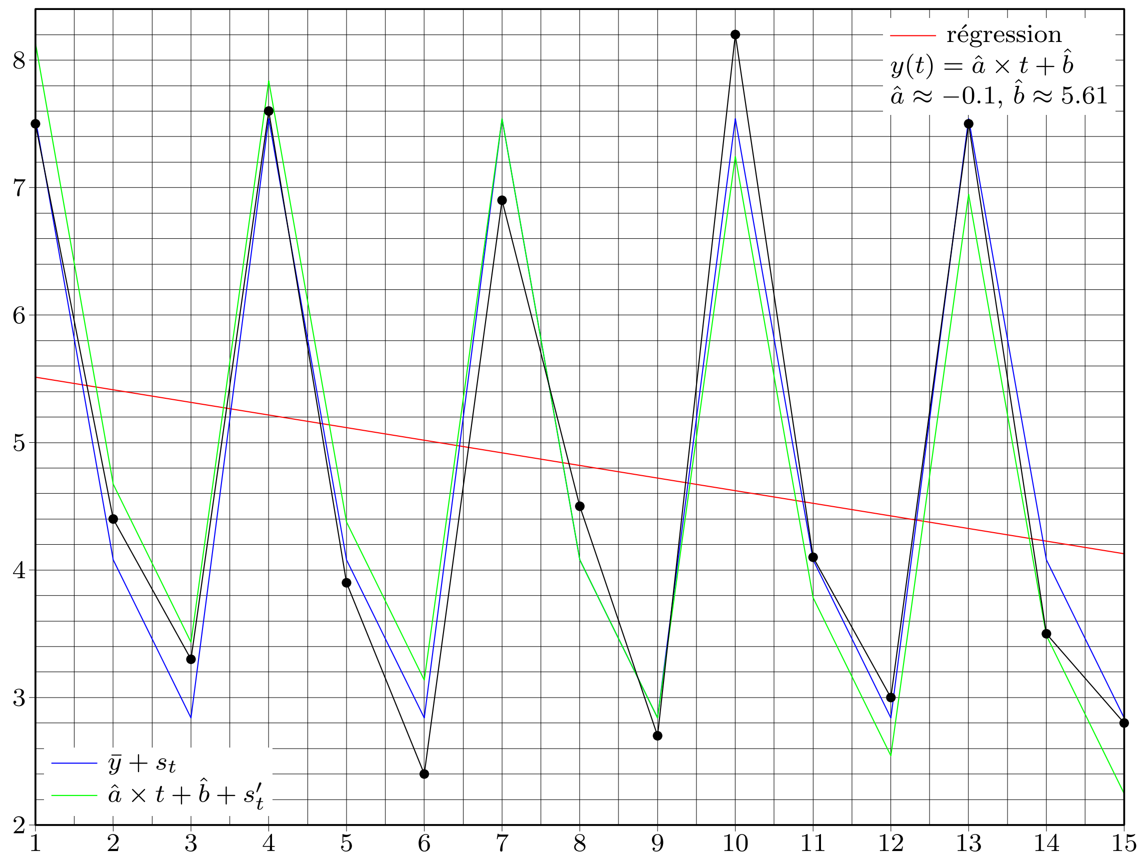
```
# Analysons le bruit en le supposant stationnaire centr\'e.
# Nous n'avons gu\`ere le choix pour estimer la fonction
# de covariance. Seuls les tous premiers termes peuvent avoir
# un peu de sens.

rho <- rep(0, times = n);
for (i in 0:(n-1)) {
    for (j in 1:(n-i)) {rho[i+1] <- rho[i+1]+E[j]*E[j+i]}
    rho[i+1] <- rho[i+1]/(n-i);
}
rho <- rho/rho[1];
rho

# proc\'edons \`a la r\'egression lin\'eaire
# en faisant les calculs pas \`a pas et assez na\"\i vement

Tmean <- mean(T); # Ymean <- mean(Y); d\'ej\`a calcul\'e

TYcov <- mean(T*Y)-Tmean*Ymean; # notons la multiplication terme \`a terme
Tvar <- mean(T*T)-Tmean^2;# idem
ahat <- TYcov/Tvar; bhat <- Ymean-ahat*Tmean;

# nous red\'efinissons la tendance

Fprime <- ahat*T+bhat;

# et tra\c cons un premier graphique

#clf(); ?
plot(T, Y, type = "o"); lines(T, Fprime, type = "o");
cat("***taper return pour continuer***"); readLines(n = 1) -> pause;

# Puis, nous r\'ep\'etons les \'etapes pr\'ec\'edentes.

Sprime <- rep(0, times = n);
```

```
for (i in 1:p) {
    for (j in 0:(ceiling(n/p)-1)) {
        Sprime[i] <- Sprime[i]+(Y[i+j*p]-Fprime[i+j*p]);
        if (i+(j+1)*p > n) break;
    }
    Sprime[i] <- Sprime[i]/(j+1);
}

for (i in (p+1):n) {Sprime[i] <- Sprime[(i-1)%%p+1]}# noter le modulo

Eprime <- Y-Fprime-Sprime;

RESUME <- list(T, Y, Sprime, Eprime)

#clf(); ?
plot(c(T, T, T, T, T, T, T), c(Y, F, S, E, Fprime, Sprime, Eprime),
    type = "p", xlab = "T", ylab = "Y, S, E, F', S', E'");
lines(T, Y); lines(T, F); lines(T, S); lines(T, E);
lines(T, Fprime); lines(T, Sprime); lines(T, Eprime);
cat("***taper return pour continuer***"); readLines(n = 1) -> pause;

#clf(); ?
split.screen(c(3, 1));
screen(1); plot(T, Y, type = "o");
lines(T, F, type = "o"); lines(T, Fprime, type = "o");
screen(2); plot(T, S, type = "o"); lines(T, Sprime, type = "o");
screen(3); plot(T, E, type = "o"); lines(T, Eprime, type = "o");
close.screen(all = TRUE);
cat("***taper return pour continuer***"); readLines(n = 1) -> pause;

rho <- rep(0, times = n);
for (i in 0:(n-1)) {
    for (j in 1:(n-i)) {rho[i+1] <- rho[i+1]+Eprime[j]*Eprime[j+i]}
    rho[i+1] <- rho[i+1]/(n-i);
}
rho <- rho/rho[1];
rho

# T est le temps, Y la s\'erie temporelle,
# p la p\'eriode, d le degr\'e de la r\'egression polynomiale

TSanalysis <- function(T, Y, p, d) {
    # local n powerT Phi i j F S E
    n <- length(Y);
    #
    # tendance polynomiale
    #
    d <- min(d, n-1); # le param\`etre d n'est pas prot\'eg\'e !!!
    a <- rep(0, times = d+1);
    if (d == 0) {a[1] <- mean(Y); F <- rep(a[1], times = n)} # tendance
    else {
        if (d == 1) {
            a[2] <- (mean(T*Y)-mean(T)*mean(Y))/(mean(T*T)-mean(T)^2);
            a[1] <- mean(Y)-a[2]*mean(T);
            F <- a[1]+a[2]*T;# tendance
```

```
    }
    else {# d >= 2
        powerT <- rep(1, times = n); Phi <- numeric();
        for (i in 1:(d+1)) {
            Phi <- c(Phi, powerT);
            powerT <- powerT*T;# produit termes \`a termes
        }
        Phi <- matrix(Phi, nrow = n, ncol = d+1);
        a <- qr.coef(qr(Phi), Y);
        F <- qr.fitted(qr(Phi), Y);# manuel, p. 47-48
    }
}
#
# composante saisonni\`ere
#
S <- rep(0, times = n);
if (p >= n) {S <- Y-F}
else {
    if (p > 1) {# estimer les coefficients
        for (i in 1:p) {
            for (j in 0:(ceiling(n/p)-1)) {
                S[i] <- S[i]+(Y[i+j*p]-F[i+j*p]);
                if (i+(j+1)*p > n) break;
            }
            S[i] <- S[i]/(j+1);
        }
        for (i in (p+1):n) {S[i] <- S[(i-1)%%p+1]}
    }
    else {}# p <= 1 : ne rien faire, la composante saisonni\`ere est nulle
}
#
# composante irr\'eguli\`ere
#
E <- Y-F-S;
#
# Synth\`ese des r\'esultats (tableaux, graphiques)
#
RESUME <- rbind(T, Y, F, S, E);
plot(c(T, T, T, T), c(Y, F, S, E), type = "p", xlab = "T");
lines(T, Y); lines(T, F); lines(T, S); lines(T, E);
title("Decomposition de la serie temporelle");
legend(T[1], Y[1], "serie temporelle", bty = "n");
legend(T[n], F[n], "tendance", bty = "n");
legend(T[1], S[1], "composante saisonniere", bty = "n");
legend(T[n], E[n], "composante irreguliere", bty = "n");
cat("***taper return pour continuer***"); readLines(n = 1) -> pause;
split.screen(c(3, 1));
screen(1);
plot(c(T,T), c(Y, F), type = "p", xlab = "T");
lines(T, Y); lines(T, F);
```

```
    title("Serie temporelle et tendance");
    #legend(T[1], Y[1], "serie temporelle", bty = "n");
    #legend(T[1], F[1], "tendance", bty = "n");
    screen(2);
    plot(c(T, T), c(S, E), type = "p", xlab = "T");
    lines(T, S); lines(T, E);
    title("Composante saisonniere et bruit");
    #legend(T[1], S[1], "composante saisonniere", bty = "n");
    #legend(T[1], E[1], "composante irreguliere", bty = "n");
    #
    # \'Etude sommaire du bruit
    # Se rappeller que seules les premi\`eres corr\'elations
    # ont peut-\^etre un peu de pertinence !
    #
    rho <- rep(0, times = n);
    for (i in 0:(n-1)) {
        for (j in 1:(n-i)) {rho[i+1] <- rho[i+1]+E[j]*E[j+i]}
        rho[i+1] <- rho[i+1]/(n-i);
    }
    rho <- rho/rho[1];
    screen(3);
    barplot(rho);
    title("Correlation estimee du bruit");
    close.screen(all = TRUE);
    list(RESUME, a, rho)
}

RESULT <- TSanalysis(T, Y, 3, 1);
```

Annexes



t_i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y_i	7.5	4.4	3.3	7.6	3.9	2.4	6.9	4.5	2.7	8.2	4.1	3	7.5	3.5	2.8
s_i	2.72	−0.74	−1.98	2.72	−0.74	−1.98	2.72	−0.74	−1.98	2.72	−0.74	−1.98	2.72	−0.74	−1.98
e_i	−0.04	0.32	0.46	0.06	−0.18	−0.44	−0.64	0.42	−0.14	0.66	0.02	0.16	−0.04	−0.58	−0.04
x_i	1.99	−1.01	−2.01	2.38	−1.22	−2.62	1.98	−0.32	−2.02	3.58	−0.42	−1.42	3.17	−0.73	−1.33
s'_i	2.62	−0.74	−1.88	2.62	−0.74	−1.88	2.62	−0.74	−1.88	2.62	−0.74	−1.88	2.62	−0.74	−1.88
e'_i	−0.63	−0.27	−0.13	−0.24	−0.48	−0.74	−0.64	0.42	−0.14	0.96	0.32	0.46	0.55	0.01	0.55

Références

[1] Chancelier (J.-P.), Delebecque (F.), Gomez (C.), Goursat (M.), Nikoukhah (R.), Steer (S.), *Introduction à Scilab, deuxième édition*, Springer-Verlag France (2007).

[2] Press (W.H.), Teukolsky (S.A.), Vetterling (W.T.), Flannery (B.P.), *Numerical Recipes in C*, second edition, Cambridge University Press (2002).

[3] Venables (W. N.), Smith (D. M.) and the R development Core Team, *An Introduction to R*, notes on R: A Programming Environment for Data Analysis and Graphics, Version 1.9.1 (2004-06-01).