Pratique R

Yves Aragon

Séries temporelles avec R

Méthodes et cas

Springer

Séries temporelles avec R
Méthodes et cas

Springer

Paris
Berlin
Heidelberg
New York
Hong Kong
Londres
Milan
Tokyo

Yves Aragon

Séries temporelles avec R
Méthodes et cas

Springer

Yves Aragon
Professeur émérite
Université Toulouse 1 – Capitole
2 rue du Doyen-Gabriel-Marty
31042 Toulouse Cedex 9

ISBN : 978-2-8178-0207-7 Springer Paris Berlin Heidelberg New York
ISSN : 2112-8294

Imprimé en France

Springer-Verlag est membre du groupe Springer Science + Business Media

Maquette de couverture: Jean-François Montmarché

Collection Pratique R

dirigée par Pierre-André Cornillon
et Eric Matzner-Løber

Département MASS
Université Rennes-2-Haute-Bretagne
France

Déjà paru dans la même collection :

Régression avec R
Pierre-André Cornillon, Eric Matzner-Løber, 2011

Méthodes de Monte-Carlo avec R
Christian P. Robert, George Casella, 2011

PREFACE

C'est un réel plaisir de vous inviter à entrer dans le monde des séries chronologiques en utilisant cet excellent livre, écrit par Yves Aragon, professeur émérite à l'université Toulouse-I-Capitole.

Le contenu est présenté de manière efficace et pragmatique, ce qui le rend très accessible non seulement aux chercheurs, mais aussi aux utilisateurs non universitaires. Ceci est très important, parce que de tous les côtés de presque tous les océans il y a une forte demande de praticiens en statistique appliquée (discipline également connue sous le nom d'*Analytics* dans le monde de l'entreprise).

L'esprit du volume est tout à fait celui de la revue *Case Studies in Business, Industry and Government Statistics* (CSBIGS), dont le professeur Aragon est membre de l'équipe éditoriale. Cette revue a été fondée il y a plusieurs années pour répondre à la demande des praticiens ainsi que des chercheurs, pour des cas qui privilégient les approches pratiques, et comportent des données permettant de reproduire les analyses. Ce même esprit anime également le programme exceptionnel de statistiques appliquées à l'université de Toulouse-I, dirigé par le professeur Christine Thomas-Agnan (coéditeur Europe pour CSBIGS), où le professeur Aragon a enseigné de nombreuses années.

Le choix du logiciel de statistique R, et la fourniture de code R et de données permettant aux lecteurs de s'entraîner ajoutent encore à l'intérêt de ce livre. R, déjà largement utilisé dans les milieux universitaires, est un outil de plus en plus important pour les praticiens dans le monde de l'entreprise. Dans la formation des étudiants à l'université de Bentley en *Analytics*, l'accent est mis sur le trio SAS®, IBM® SPSS® et R. Les étudiants sont encouragés à prendre un cours pratique de séries chronologiques basé sur R, non seulement parce que l'étude des séries chronologiques est importante, mais parce que les employeurs veulent embaucher des analystes maîtrisant R ; ce langage doit donc être ajouté au curriculum vitae.

L'ouvrage aborde l'étude de séries temporelles avec une approche claire et logique. Il commence par les moindres carrés ordinaires, en soulignant les limites de la méthode (car les erreurs sont corrélées dans de nombreux cas). Puis il conduit le lecteur vers le modèle ARIMA et ses extensions, y compris les modèles avec hétéroscédasticité conditionnelle. Un aspect particulièrement intéressant du livre est l'utilisation de la simulation comme outil de validation pour les modèles de séries chronologiques.

Le volume présente plusieurs cas fascinants, y compris une étude du trafic de passagers à l'aéroport de Toulouse-Blagnac avant et après les attaques terroristes du 11 Septembre 2001.

J'espère qu'il y aura bientôt une traduction en anglais de cet ouvrage, afin que les lecteurs qui n'ont pas de compétence en français puissent en bénéficier. Ce serait la cerise sur le gâteau.

Professeur Dominique Haughton,
Bentley University.

REMERCIEMENTS

Un certain nombre de collègues m'ont apporté une aide décisive dans l'élaboration de cet ouvrage.
Thibault Laurent a prêté attention de manière spontanée et désintéressée à mon travail, et relu minutieusement de larges pans du manuscrit. Il m'a fait mesurer toute la puissance de Sweave et exploiter son automatisation par Make. Plus largement j'ai bénéficié de sa connaissance étendue et précise de R. Il a amélioré très sensiblement le code des exemples et levé toutes les difficultés de programmation en expert. Il s'est chargé du package et du site du livre. Nos discussions m'ont toujours été très profitables.
Nadine Galy, professeur à l'ESC de Toulouse, m'a suggéré des séries financières avec leurs problématiques. Michel Simioni, directeur de recherche à l'INRA et Anne Vanhems, professeur à l'ESC de Toulouse, ont relu attentivement certains chapitres. Leurs questions m'ont amené à clarifier plusieurs points.
Cet ouvrage est issu d'un cours en master Statistique et économétrie, appuyé sur R pendant quelques années ; des étudiants de ce master, aussi bien en face à face qu'à distance, ainsi que des étudiants inconnus qui ont consulté des parties du cours sur Internet, m'ont signalé des points épineux. J'ai pu ainsi améliorer l'exposé.
Ce travail utilise tantôt des données classiques, ce qui permet au lecteur de comparer notre approche à d'autres démarches, tantôt des données nouvelles. Ces dernières, originales sans être confidentielles, ne sont pas faciles à obtenir. Le SRISE-DRAAF Champagne-Ardenne m'a facilité l'accès à la série sur le vin de Champagne. Le service de la statistique et de la prospective du ministère de l'Alimentation, de l'Agriculture et de la Pêche a mis à ma disposition ses données sur la collecte mensuelle de lait. La chambre régionale de commerce et d'industrie Midi-Pyrénées m'a fourni les données de trafic de l'aéroport de Toulouse-Blagnac.
Les relectures et conseils de Pierre-André Cornillon et Eric Matzner-Løber ont largement contribué à rendre le livre plus clair. Charles Ruelle de Springer-Verlag, d'emblée intéressé par cet ouvrage, présent à chaque étape du travail, m'a piloté de bonne grâce.
Que tous ces collaborateurs, connus ou inconnus, qui ont permis de transformer un travail assez solitaire en un travail d'équipe, veuillent bien accepter mes très vifs remerciements.
Bien entendu, je suis seul responsable des fautes et imprécisions qui subsisteraient dans la version finale.

AVANT-PROPOS

Panorama

Ce livre est bâti autour de l'étude de quelques séries temporelles régulières, c'est-à-dire de suites d'observations d'un phénomène à des dates régulièrement espacées. Avant d'étudier ces séries, une première partie, les chapitres 1 à 7, est consacrée à quelques rappels sur les méthodes, en particulier sur leur utilisation concrète, avec, souvent, des exemples de mise en pratique dans R.

Les outils de visualisation de série, nombreux dans R, permettent de comprendre la structure d'une série avant toute modélisation ; ils sont présentés dès le chapitre 1 et utilisés systématiquement dans l'étude des séries. Les graphiques sont importants à toutes les étapes du traitement. Avant modélisation, ils aident à saisir la structure de la série ; après modélisation, ils offrent une vision globale de l'ajustement, vision que ne peut donner un niveau de signification empirique considéré isolément.

Quelques éléments sur R pour les séries temporelles sont donnés au chapitre 2, qui concernent principalement les dates et les structures de séries. Mais la lecture de ce chapitre demande une connaissance préalable de R.

Le chapitre 3 est consacré à la régression linéaire. Elle est illustrée par la régression par Moindres Carrés Ordinaires, d'une consommation d'électricité sur des variables de température, sans considération de la nature temporelle des données. Or dans une régression sur séries temporelles, les erreurs sont habituellement autocorrélées...

Pour modéliser les erreurs autocorrélées ou toute série présentant une dynamique, il est indispensable d'avoir des notions sur les modèles ARMA, ARIMA et leurs versions saisonnières. Le chapitre 4 est précisément consacré à des rappels sur les modèles stationnaires, ARMA en particulier. Nous ne traitons que marginalement les méthodes d'estimation ou de prévision, mais insistons sur les concepts qui sont souvent source de confusion. Par exemple, régularité ne veut pas dire stationnarité, comme le montre la série des températures à Nottingham Castle. Nous présentons le test du Portemanteau et les principes d'identification du modèle d'une série temporelle. Nous définissons les modèles ARMAX, modèles de régression linéaire d'une série temporelle sur des séries prédéterminées où l'erreur présente une dynamique. Plusieurs séries étudiées dans l'ouvrage peuvent relever d'un tel modèle : le niveau du lac Huron, la température moyenne à Nottingham, la collecte de lait, la consommation d'électricité.

Les modèles non stationnaires, pour cause de tendance déterministe ou pour cause de racine unitaire, sont examinés au chapitre 5, et les tests classiques de l'une ou l'autre situation sont mis en pratique sur des séries classiques ou sur des séries simulées. Dans ces méthodes également, l'exploration de la série doit orienter le champ des questions théoriques qu'elle soulève.

Le lissage exponentiel, chapitre 6, qui s'intéresse à la prévision d'une série plus qu'à sa modélisation, n'est considéré que dans ses modèles les plus simples mais le traitement retenu passe par le filtre d'innovation ; le lissage échappe ainsi au traitement habituel par bricolage et intuition pure, et gagne une estimation par

maximum de vraisemblance : donc une précision d'estimation.
Le chapitre 7 est consacré à la simulation. Son importance n'est pas à démontrer. D'abord, elle reproduit un mécanisme aléatoire autant de fois qu'on le souhaite. La transposition informatique du modèle d'une série, même si elle se limite à l'utilisation d'une fonction de R, est déjà une façon de vérifier qu'on a compris ce modèle, et qu'on sait lire les résultats d'estimation fournis par le logiciel. Ensuite la simulation pousse un modèle dans ses limites : simuler de nombreuses trajectoires d'une série permet de voir comment il se comporte. Appliquée sur le résultat d'une estimation, elle permet de vérifier que des estimations apparemment raisonnables conviennent bien à la série étudiée. Etant donné une série simulée suivant un mécanisme particulier, il est toujours instructif d'estimer sur la série le modèle qui a servi à la simuler pour voir comment se retrouve le modèle initial après simulation et estimation. Autre démarche éclairante : estimer un modèle incorrect, afin de repérer quels mécanismes d'alerte offrent les méthodes, quand on les applique à des séries de manière inappropriée. La simulation est parfois indispensable, par exemple si la série a subi une transformation non linéaire vers une série plus normalement distribuée que l'original. L'effet du modèle doit s'examiner sur la série initiale. Mais la transformation réciproque est une opération le plus souvent très compliquée du point de vue théorique. La simulation, plus simple et plus sûre, permet de la contourner. On a la chance, en séries temporelles, de simuler facilement des modèles très variés sans grand effort d'imagination : ceci par le caractère limité des modèles et par la variété des fonctions disponibles pour la simulation.
La deuxième partie de l'ouvrage est consacrée à des études de séries. Pour chaque série étudiée, on se pose un certain nombre de questions et l'on essaie d'y répondre, d'abord par l'exploration graphique puis par la modélisation. Le trafic passager à l'aéroport de Toulouse-Blagnac, chapitre 8, avant et après le 11 septembre, se prête à différentes comparaisons : dynamique de la série, comparaison de l'évolution attendue du trafic annuel en l'absence d'attentats, avec sa réalisation après le 11 septembre. Une série peut relever de plusieurs traitements. La série classique des températures à Nottingham Castle, objet du chapitre 9, en est un exemple : traditionnellement, elle est modélisée par un SARIMA, mais en fait son exploration suggère de la régresser sur des fonctions trigonométriques et de modéliser la dynamique de l'erreur. La prévision de la consommation d'électricité, chapitre 10, expliquée par certaines fonctions de la température, passe par une modélisation ARMAX et requiert le concours de plusieurs méthodes, chacune présentée dans la première partie. La collecte mensuelle de lait, chapitre 11, a subi l'introduction de la politique des quota. On essaie d'apprécier les conséquences de cette politique et on compare deux modélisations de la série.
Le traitement des séries présentant une hétéroscédasticité conditionnelle, méthodes et exemples, fait l'objet du chapitre 12. La compréhension de la structure de ces séries permet incidemment de lever des confusions sur la notion de stationnarité. L'estimation et la prévision portent principalement sur la variance de la série et non sur la moyenne, au contraire des séries sans hétéroscédasticité conditionnelle. L'interprétation des résultats est donc plus délicate que lorsque c'est la moyenne

qui est modélisée.
Au cours de l'ouvrage on pourra constater qu'une série peut recevoir plusieurs modélisations, toutes satisfaisantes et pourtant contradictoires d'un point de vue théorique : tendance stochastique ou déterministe par exemple. Ceci peut s'expliquer par au moins deux raisons. D'une part, nous disposons d'un nombre limité de classes de modèles et il n'est pas certain qu'une série donnée ne serait pas mieux modélisée par une autre classe de modèles. D'autre part, nous étudions des séries finies, alors que les modèles concernent souvent des séries infinies. Observons enfin que la façon dont une série est étudiée dépend de nombreux paramètres : la compétence du statisticien, les outils théoriques et pratiques dont il dispose, le temps qu'il peut y consacrer, l'objectif de l'étude, académique, pratique...

Public
Ce livre pratique entre dans les détails concrets de l'analyse des séries temporelles. Il suppose un minimum de connaissances en statistique mathématique ou en économétrie. Il nécessite également quelques connaissances de base du logiciel R. Cet ouvrage s'adresse à tous les étudiants de masters stastistique, économétrie, école d'ingénieurs mais aussi aux ingénieurs et chercheurs désirant s'initier à l'analyse des séries temporelles.

Convention
Une fonction est écrite en style `télétype` et avec parenthèses (`mean()`), un package est écrit en **gras** (**forecast**), un objet est écrit en style `télétype` sans parenthèses. Du code R apparaît dans le texte. S'il s'agit de code après exécution, il commence par un chevron `>` et est en style *télétype italique*. Le résultat de ce code, s'il figure dans le livre, apparaît en style `télétype`. Dans ce même style, on rencontrera également du code non exécuté, mais donné comme exemple de syntaxe. On utilise le point et non la virgule dans la représentation des nombres décimaux.
Nous avons utilisé les fonctions `Stangle()` et `Sweave()` pour récupérer le code et les sorties de R ; les erreurs de copier-coller entre le logiciel et le livre devraient ainsi être limitées.

Ressources
Un package, **caschrono**, installé depuis R par `install.packages("caschrono")`, regroupe les données manipulées ici ainsi que quelques fonctions souvent appelées dans l'ouvrage. Son chargement provoque celui des packages dont une ou plusieurs fonctions, ou simplement des classes d'objet, peuvent être utilisées par **caschrono**. Les corrigés des exercices ainsi que des compléments sont contenus dans des vignettes associées à certains chapitres, ainsi `vignette("Anx2")` provoque l'affichage du fichier .pdf des compléments du chapitre 2. La liste des vignettes figure en tête de l'aide en ligne de **caschrono**. Le code R nécessaire à la rédaction de l'ouvrage, en particulier le code des graphiques, est disponible sur le site de l'ouvrage [1]. Les appels à la fonction `par()`, nécessaires pour la mise en page de l'ouvrage, ne sont pas reproduits dans le livre mais figurent sur le site. On trouve également sur le site du

1. http://seriestemporelles.com/

code R complémentaire signalé dans ce travail par le sigle SiteST. Les packages appelés hors de **caschrono** sont : **car**, **chron**, **dse**, **dynlm**, **expsmooth**, **fBasics**, **fGarch**, **FinTS**, **FitARMA**, **fSeries**, **nlme**, **polynom**, **TSA**, **urca**, ainsi que **xtable** qui ne sert qu'à l'édition en Latex. Il faut évidemment les installer pour traiter les exemples. Ceux chargés par **caschrono** sont : **forecast**, **Hmisc**, **its**, **timeSeries**.

Orientations bibliographiques

Ruppert (2004) présente de façon très accessible et soignée des rappels de statistique mathématique et beaucoup des méthodes manipulées dans ce livre, avec une orientation financière. Brocklebank & Dickey (2003) ont une approche concrète et fine des séries temporelles et n'utilisent qu'un minimum de bases mathématiques ; cet ouvrage illustre l'emploi du logiciel SAS/ETS. Le présent livre ne dépasse pas leur niveau théorique. Zivot & Wang (2006), pour les séries financières, mêlent efficacement considérations théoriques et pratiques. L'approche de Tsay (2005) pour les modèles ARMA, surtout intuitive, correspond aussi à l'esprit du présent ouvrage. Pour les bases théoriques on peut se reporter à de nombreux ouvrages, notamment : Bourbonnais & Terraza (2008), Brockwell & Davis (2002), Franses (1998), Hamilton (1994), Wei (2006). Les travaux de Cryer & Chan (2008), Pankratz (1991) et Wei, déjà cité, contiennent, en plus d'un exposé théorique, des exemples numériques fouillés. Shumway & Stoffer (2006) donnent un solide exposé théorique contemporain accompagné d'exemples réalistes tirés de domaines variés. Le manuel en ligne de méthodes statistiques NIST/SEMATECH (2006), pragmatique, s'avère utile pour réviser les bases de la statistique et même pour approfondir certaines questions.

Les méthodes manipulées ici sont classiques et largement pratiquées ; des travaux de recherche ont montré les limites de certaines d'entre elles. Le lecteur qui souhaite prendre du recul sur ces méthodes consultera avec profit Kennedy (2003) et sa bibliographie.

Comment utiliser ce livre ?

Il faut d'abord installer **caschrono**, les packages utilisés hors **caschrono** et récupérer le code sur le site. Ceci fait, la meilleure façon de travailler est d'exécuter le code pas à pas, et même ligne à ligne pour les graphiques, en vue de s'approprier les commentaires et conclusions, quitte à les contester ! Il est très formateur de simuler des séries puis de leur appliquer les techniques d'exploration et de modélisation.

Table des matières

Chapitre 1

Démarche de base en séries temporelles

Nous présentons ici quelques séries temporelles observées ou simulées, et envisageons différents objectifs que peut viser leur étude. Quel que soit l'objectif de l'étude, il faut commencer par l'exploration de la série. On examine donc différents types de graphiques offerts par R : ils permettent de comprendre la structure d'une série temporelle et de guider sa modélisation éventuelle. Le premier graphique à utiliser pour l'examen d'une série est son chronogramme, c'est-à-dire le diagramme des points (date, valeur de l'observation). Nous examinons à la section 1.2 le lag plot qui permet de visualiser la dépendance entre une série et la même à des dates antérieures, et le month plot qui aide à comprendre la saisonnalité d'une série.
Ces graphiques permettent de voir la nature de la tendance d'une série ou ses aspects saisonniers, ou encore l'existence d'une autocorrélation dans la série. Tendance et saisonnalité sont généralement organisées dans un schéma de décomposition saisonnière (section 1.3). Enfin, nous présentons l'opérateur retard qui permet de noter synthétiquement beaucoup des opérations faites sur une série.

1.1 Exemples de séries temporelles

Une série temporelle, $\{y_t, t = 1, 2, \cdots, T\}$, est une suite d'observations d'un phénomène, indicées par une date, un temps. Le moment auquel l'observation est faite est généralement une information importante sur le phénomène observé. Nous allons considérer le chronogramme de quelques séries et relever leurs caractéristiques les plus évidentes.
Pour commencer, nous chargeons les packages utilisés dans le chapitre.

```
> require(caschrono)
> require(fSeries)
```

Exemple 1.1 (Populations) Les populations de la France et des Etats-Unis, en millions d'habitants, (fig. 1.1) constituent des séries où le temps explique bien le niveau de la série.

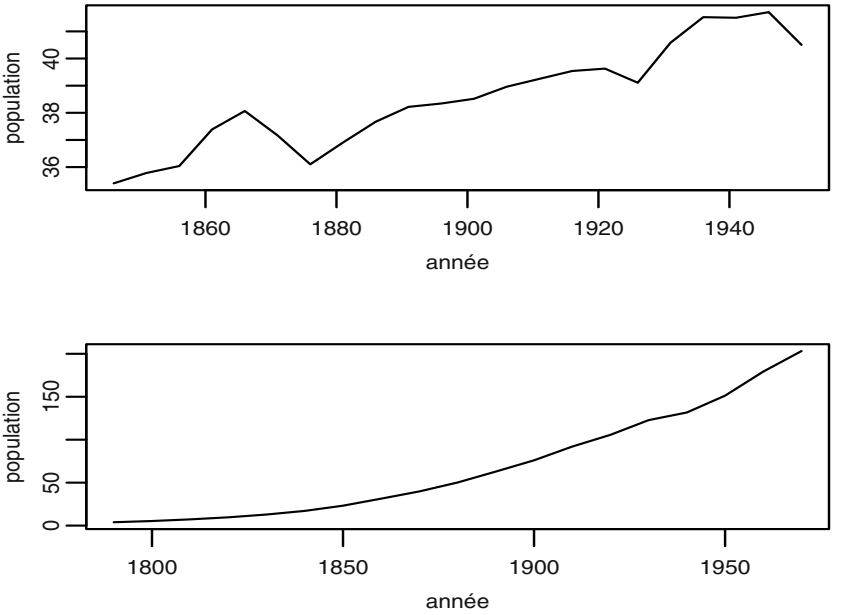

Fig. 1.1 – Population française (haut), population des Etats-Unis (bas) en millions d'habitants.

Leur graphique est obtenu par :

```
> data(popfr)
> plot.ts(popfr,xlab='année',ylab='population')
> plot.ts(uspop,xlab='année',ylab='population')
```

On précisera au chapitre suivant ce qu'est une série temporelle pour R. La commande `plot.ts()` dessine le chronogramme et il n'est pas nécessaire de préciser l'abscisse, une notion de temps étant intégrée à la série temporelle (cf. chap. 2.2). On peut noter qu'une fonction du temps assez lisse capte une grande part de la variabilité de la série, et en première approche rend bien compte de l'évolution de la série. Les démographes peuvent être intéressés par la prévision de la population à 10 ans, à 20 ans. Remarquons que la population française varie de 35.4 à 41.7 sur 110 ans, tandis que celle des Etats-Unis varie de 3.93 à 203.00 sur 180 ans.

Exemple 1.2 (Morts par accident) Le nombre mensuel de morts et blessés graves par accident de la route en France métropolitaine (fig. 1.2 haut) montre d'importantes fluctuations saisonnières. En vérité elles sont peu explicites sur la série complète, alors qu'un zoom de la série sur quelques années (fig. 1.2 bas)

montre que les mois de décembre et janvier présentent moins d'accidents. Sur la figure supérieure le zoom a été marqué à l'aide de la fonction `polygon()` :

```
> data(m30)
> plot.ts(m30,xlab="année",ylab="nombre de morts",las=1)
> polygon(c(1995,2005,2005,1995),c(340,340,910,910),lty=2)
> deb=c(1995,1); fin=c(2004,12) # zoom
> plot.ts(window(m30,start=deb,end=fin),xlab="année",ylab="nombre de morts")
```

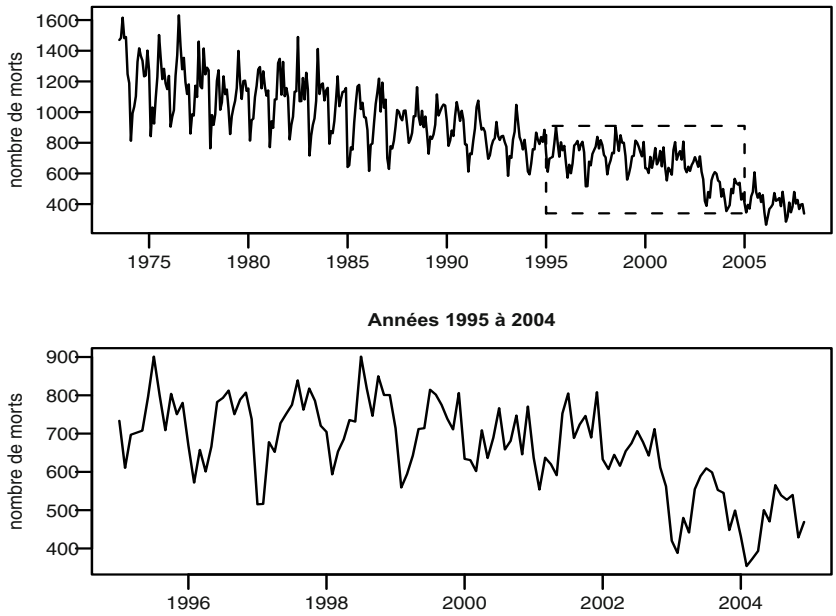

Fig. 1.2 – Accidents de la route - morts à 30 jours.

On note également une décroissance du niveau moyen de la série, qui s'accélère à partir de l'année 2000 ; cette décroissance est accompagnée d'une diminution de la variabilité [1]. Sans essayer d'expliquer cette évolution, on peut, à l'aide de ces seules données, essayer d'éclaircir un certain nombre de questions.

Par exemple, un service de santé publique peut souhaiter une vision synthétique de la situation, un aperçu de la tendance au cours des années. Un lissage des données répondrait à cette question (SiteST). On peut vouloir comparer un certain mois de janvier aux autres mois de janvier, ou plus généralement étudier la saisonnalité de

1. Un certain nombre de mesures ont été prises à partir de 1983 pour diminuer le nombre d'accidents sur les routes en France : introduction de limiteurs de vitesse sur les poids lourds, limitation de la vitesse à 50 km/h en ville... jusqu'à l'introduction du premier radar automatique en 2003.

la série, c'est-à-dire comparer les mois de janvier aux mois de février, de mars[2]... Le chronogramme n'est pas adapté à une telle étude. A la section suivante, nous présentons les month plots. Le month plot de cette série (SiteST) montre, sur la période considérée, de fortes diminutions des mois de juillet, août, septembre, octobre et novembre par rapport aux mêmes mois de l'année précédente, et une diminution plus faible des autres mois de l'année.

Exemple 1.3 (Champagne) Les expéditions mensuelles de champagne (fig. 1.3 haut) montrent une forte saisonnalité, avec un pic en novembre.

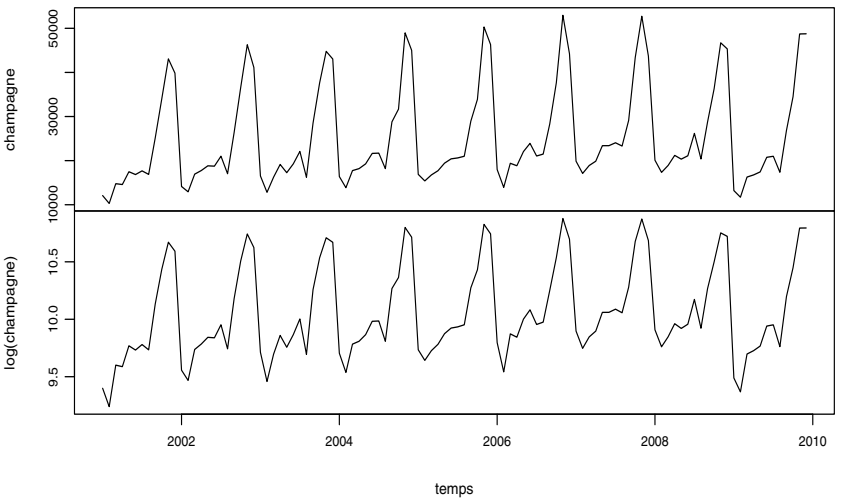

Fig. 1.3 – Expéditions de champagne : série et log(série).

On observe également une croissance de la série jusqu'à la crise de 2008 ; de plus, comme c'était le cas pour la série des nombres de morts sur la route, on voit que la variabilité de ces séries augmente avec le niveau moyen de la série. Ce sont des séries *hétéroscédastiques* (c'est-à-dire à variance non constante). Des transformations telles que log(.) (fig. 1.3 bas) $\sqrt{.}$, peuvent atténuer cette hétéroscédasticité. Elles sont présentées au chapitre 3, sous la rubrique « Stabilisation de la variance ». A l'aide de ces seules données, on peut essayer d'éclaircir un certain nombre de questions. Les données, nombre de bouteilles par mois, figurent dans un fichier texte, à raison d'une observation par enregistrement. Dans le code ci-dessous, on importe les données grâce à la fonction `scan()`. On les exprime en milliers de bouteilles et on forme la matrice de cette série et de son log ; on convertit cette matrice en série temporelle bidimensionnelle par `ts()`. Ensuite, `plot.ts()` appliqué à la série bidimensionnelle `ytr.ts` nous permet de surperposer les chronogrammes des composantes.

2. La saisonnalité dont il sera souvent question est à comprendre comme une périodicité plus ou moins marquée et régulière ; son sens sera précisé au cours du livre.

```
> aa=scan(system.file("/import/champagne_2001.txt",package="caschrono"))
> champa=aa/1000; #expéditions en milliers de bouteilles
> ytr=cbind(champa,log(champa))
> colnames(ytr)=c("champagne","log(champagne)")
> ytr.ts=ts(ytr,start=c(2001,1),frequency=12)
> plot.ts(ytr.ts,xlab='temps',main='',oma.multi=c(4.5,3,.2,0),
+   mar.multi=c(0,4,0,.5),las=0)
```

La chambre syndicale des producteurs de vin peut être intéressée par la tendance des ventes débarrassée des fluctuations de court terme. Elle a donc besoin d'une représentation lissée de la série. Pour une réflexion stratégique, une série longue est nécessaire. Par contre, un syndicat de transporteurs a besoin de savoir combien de bouteilles devront être livrées le mois prochain ou le suivant. Cette prévision-là devra évidemment incorporer les fluctuations saisonnières. La connaissance de la série plus de trois ans avant la période à prédire lui est de peu d'utilité. Evidemment, la série seule ne permet pas d'anticiper la crise de la fin 2008.
Les modèles de prévision à court terme : modèles ARIMA, lissage exponentiel sont rappelés dans les chapitres 4, 5 et 6, et utilisés dans les cas pratiques.

Exemple 1.4 (Danone)

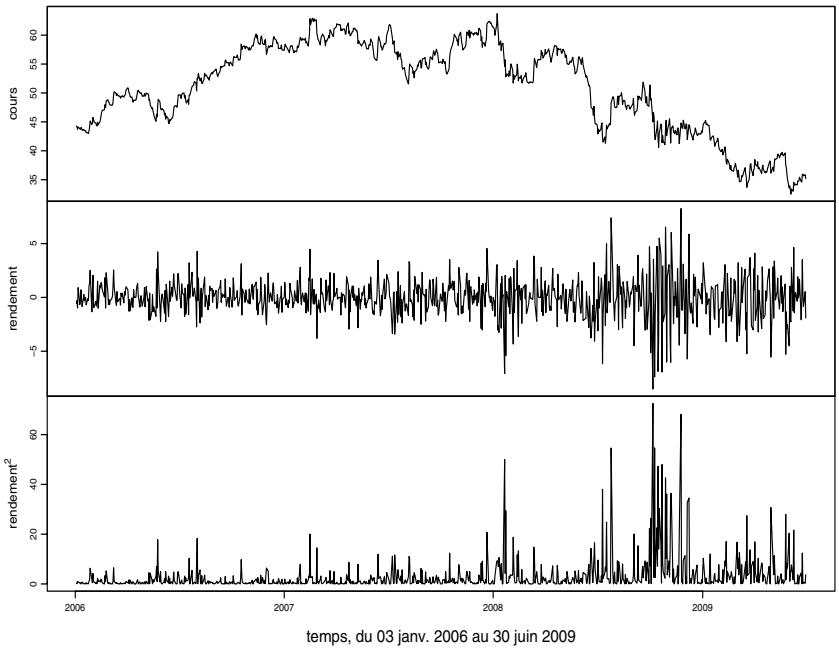

Fig. 1.4 – Danone - cours, rendement, rendement au carré.

On a porté sur la figure 1.4, en haut le cours quotidien de l'action Danone, au milieu le rendement composé défini chap. 12, et en bas le carré de ce rendement [3]. En haut, le cours est assez imprévisible : il peut atteindre des valeurs très élevées aussi bien que très basses. Au milieu, le rendement est pratiquement de moyenne nulle. Il peut être élevé en valeur absolue pendant plusieurs périodes, puis rester faible un certain temps. En bas, le rendement étant donc pratiquement nul, la moyenne de son carré dans un intervalle de quelques jours consécutifs représente la variance quotidienne dans cet intervalle. On observe de faibles valeurs sur de longues périodes donc des variances faibles, puis de fortes valeurs sur d'autres périodes. Autrement dit, la variance à une date dépend des valeurs passées de la série. C'est un exemple de série à hétéroscédasticité conditionnelle. Le rendement d'une action est souvent de moyenne nulle et les rendements à 2 dates consécutives souvent non corrélés, on ne peut donc le prédire que par la valeur moyenne, 0. Mais la variabilité du rendement qui, elle, montre une corrélation entre dates consécutives peut être prédite. Les modèles de séries à hétéroscédasticité conditionnelle sont susceptibles d'accomplir cette prévision. Nous les présentons au chapitre 12.

Exemple 1.5 (Lac Huron) Considérons enfin la série, disponible dans R, des mesures annuelles en pieds du niveau du lac Huron de 1875 à 1972. Nous reprenons l'étude qu'en font Brockwell & Davis (2002). Le chronogramme (fig. 1.5) montre une tendance décroissante approximativement linéaire.
Commençons donc par régresser linéairement le niveau sur le temps (1875, 1876...) par MCO (moindres carrés ordinaires) ; on superposera la droite ajustée au chronogramme. Notons x_t le temps : $x_1 = 1875, x_2 = 1876, \cdots$ et y_t le niveau en t. On veut donc ajuster par MCO le modèle :

$$y_t = \beta_0 + \beta_1 x_t + u_t, \quad t = 1, \cdots, T. \tag{1.1}$$

Par la fonction `time()` nous extrayons le temps de la série, la fonction `lm()` accomplit la régression.

```
> temps=time(LakeHuron)
> reglin=lm(LakeHuron~temps)
> resi.mco=residuals(reglin)
> ychap=fitted(reglin)
> v12=c(var(LakeHuron),var(resi.mco))
```

Nous avons extrait les résidus de cet ajustement par `residuals()` et les valeurs ajustées par `fitted()`. Nous calculons également les variances totale et résiduelle. La variance de la série est : `v12[1]` = 1.7379 tandis que la variance non expliquée par la régression est : `v12[2]` = 1.2644. La variance résiduelle est environ 72% de la variance initiale.

3. Le graphique superpose les trois chronogrammes avec une même échelle de temps et sans marge entre les chronogrammes (SiteST).

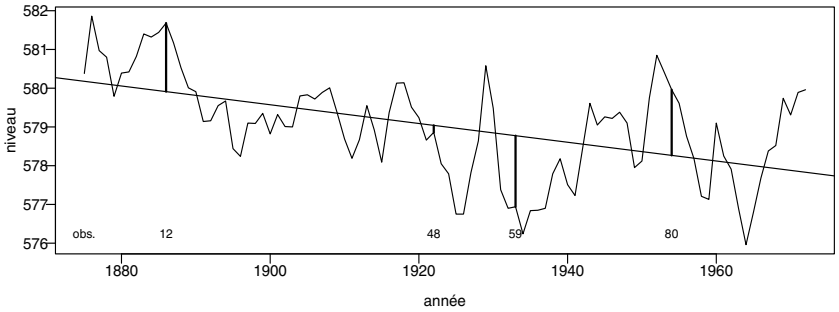

Fig. 1.5 – Niveau du lac Huron, ajustement MCO, résidus.

Nous représentons le chronogramme de la série, la droite ajustée et quelques résidus grâce aux commandes :

```
> plot.ts(LakeHuron,las=1,ylab="niveau",xlab="année")
> abline(coef=coef(reglin))
> s=c(12,48,59,80);
> segments(temps[s],ychap[s],temps[s],LakeHuron[s],lty=3,lwd=1)
> y.gra=1.0009*par()$usr[3]
> text(temps[1],y.gra,labels="obs.",cex=.8)
> text(temps[s],rep(y.gra,length(s)),labels=s,cex=.8)
```

Après avoir dessiné le chronogramme par `plot.ts()`, nous superposons la droite ajustée par `abline()`. Les résidus pour les observations n° 12, 48, 59 et 80 sont matérialisés par `segments()` et un commentaire ajouté par `text()`. La position des bords du graphique est récupérée dans `par()$usr`. Il est important de remarquer que les résidus consécutifs ont tendance à être de même signe (fig. 1.6).
Examinons d'un peu plus près ces résidus. Nous représentons leur chronogramme (fig. 1.6 haut) par le code :

```
> plot(as.vector(temps),resi.mco,type='l',xlab="année",ylab='résidu')
> abline(h=0)
> zero=rep(0,length(s))
> segments(temps[s],zero,temps[s],resi.mco[s],lty=3,lwd=1)
> y.gra=0.9*par()$usr[3]
> text(temps[1],y.gra,labels="obs.",cex=.8)
> text(temps[s],rep(y.gra,length(s)),labels=s,cex=.8)
```

La régression linéaire contenant une constante, les résidus sont de moyenne nulle et nous avons matérialisé cette moyenne par `abline(h=0)`. Nous dessinons également le diagramme de dispersion des points (résidu en $t-1$, résidu en t) (fig. 1.6 bas) :

```
> n=length(resi.mco)
> plot(resi.mco[-n],resi.mco[-1],xlab="résidu en t-1",asp=1,
+      ylab='résidu en t')
```

Ce diagramme s'appelle un *lag plot*, il sera longuement présenté à la section suivante.

On observe, plus clairement sur la figure 1.6 que sur la figure 1.5, que les résidus consécutifs ont tendance à être de même signe.

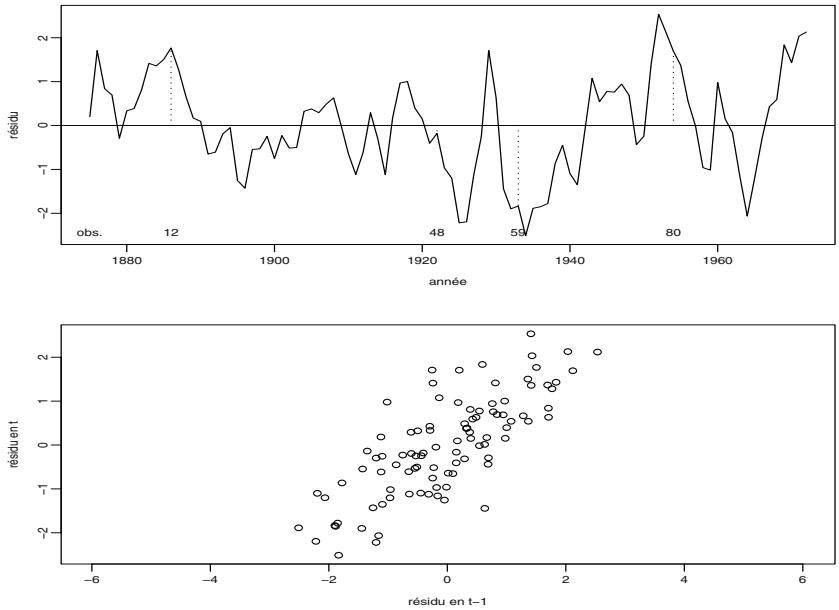

Fig. 1.6 – Niveau du lac Huron : chronogramme et diagramme de dispersion (résidu en t, résidu en $t-1$) des résidus MCO.

On voit aussi que le lag plot des résidus (fig. 1.6 bas) indique une assez forte corrélation entre la série et la série retardée d'un an. Les résidus sont autocorrélés. Si l'on arrive à modéliser la dynamique de l'erreur révélée par l'étude des résidus, on obtiendra un meilleur ajustement de la tendance. De plus, prendre en compte l'autocorrélation des $\widehat{u}_t$ permettra d'améliorer la prédiction. Par des techniques que nous rappellerons au chapitre 4, nous retenons le modèle suivant pour u_t :

$$u_t = \phi u_{t-1} + z_t, \qquad |\phi| < 1 \tag{1.2}$$

où les z_t sont indépendants identiquement normalement distribués. Ainsi, le modèle de y_t est composé de deux parties : un modèle pour la moyenne, $\mathsf{E}(y_t) = \beta_0 + \beta_1 x_t$, avec une erreur $u_t = y_t - \mathsf{E}(y_t)$ et un modèle pour l'erreur, $u_t = \phi u_{t-1} + z_t$. La prédiction de y_{t+1} est la somme de la prédiction du niveau moyen en $t+1$: $\widehat{\beta}_0 + \widehat{\beta}_1 x_{t+1}$ et de la prédiction de l'erreur en $t+1$ connaissant les erreurs passées. Nous avons là, un exemple de modèle ARMAX. On ajuste ce modèle par la fonction `Arima()` (cf. chap. 4).

```
> mod.lac=Arima(LakeHuron,order=c(1,0,0),xreg=temps,method='ML')
> ychap=fitted(mod.lac)
> resi.inno=residuals(mod.lac)
> v23=c(var(resi.mco),var(resi.inno))
```

Notons qu'il y a deux types de résidus dans ce modèle : (1) les $\widehat{u}_t$, écarts des y_t à la moyenne ajustée $\widehat{y}_t = \widehat{\beta}_0 + \widehat{\beta}_1 x_t$ et (2) les résidus $\widehat{z}_t$, stockés dans `resi.inno`, associés à la dynamique de l'erreur. La variance des $\widehat{z}_t$ est : 0.5016. Elle est sensiblement plus faible que la variance des $\widehat{u}_t$: 1.2644. Elle vaut 29% de la variance de la série. C'est la réduction de variance prise en compte par la tendance et la dynamique de l'erreur (1.2).
Pour évaluer l'intérêt de cette modélisation, dessinons le chronogramme de la prédiction ainsi que le lag plot des points $\widehat{z}_t$.

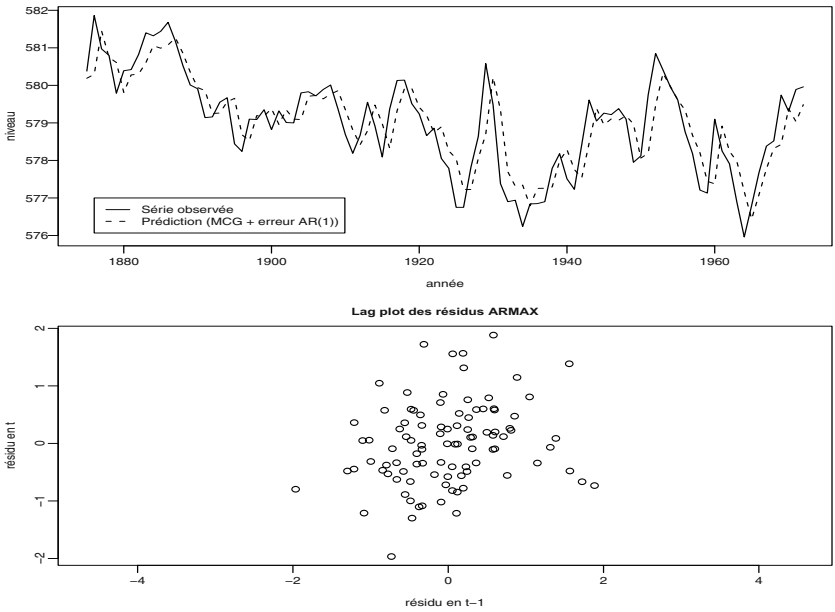

Fig. 1.7 – Niveau du lac Huron : niveau et ajustement MCG avec erreur AR(1) (haut), lag plot de `resi.inno` (bas).

```
> plot.ts(cbind(LakeHuron,ychap),lty=1:2,type="l",las=1,
+  plot.type="single",ylab="niveau",xlab="année")
> leg.txt=c('Série observée','Prédiction (MCG + erreur AR(1))')
> legend(x=1876,y= 577,leg.txt,lty=1:2,cex=.8)
> n=length(resi.inno)
> plot(resi.inno[-n],resi.inno[-1],xlab="résidu en t-1",asp=1,
+    ylab='résidu en t')
```

Le chronogramme (fig. 1.7 haut) de la prédiction prenant en compte l'autocorrélation des erreurs suit bien celui de la série observée. Le lag plot des $\widehat{z}_t$ (fig. 1.7 bas) ne montre plus de corrélation, au contraire de ce qu'on observait sur le diagramme de dispersion des résidus MCO (fig. 1.6 bas). Notons que `legend()` place une légende en un point qui s'exprime dans les unités de la série représentée.

Pour trouver le modèle susceptible de convenir à l'erreur d'ajustement (1.2), on a travaillé sur les résidus MCO, `resi.mco`. Pour voir si (1.2) convient, on a ajusté ce modèle à ces résidus par :

```
> (mod.res=Arima(resi.mco,order=c(1,0,0)))
```

et on a examiné si le résidu de ce modèle, qui correspond à z_t, pouvait être une suite d'observations i.i.d. ou au moins de même variance, et non corrélées. Les `()` qui encadrent une expression provoquent l'impression de la valeur de cette expression à la console. Ici, les résultats stockés dans `mod.res` sont volumineux et seul un extrait est imprimé à l'écran. Nous aurons l'occasion, dès le prochain chapitre, de fouiller dans les résultats d'une fonction statistique pour en extraire ce qui nous intéresse.

On peut retenir de ce survol qu'une même série peut être étudiée en vue de différents usages et peut donc recevoir des traitements variés.

1.2 Graphiques pour les séries temporelles

1.2.1 Chronogramme

Comme on l'a vu dans la section précédente, l'étude d'une série temporelle commence par l'examen de son chronogramme. Il en donne une vue d'ensemble, montre certains aspects, comme des valeurs atypiques, d'éventuelles ruptures, un changement dans la dynamique de la série... et il peut suggérer d'autres graphiques complémentaires que nous examinons ici.

1.2.2 Lag plot

Un lag plot ou diagramme retardé est le diagramme de dispersion des points ayant pour abscisse la série retardée de k instants et pour ordonnée la série non retardée. Si le diagramme retardé suggère une corrélation entre les deux séries, on dit que la série présente une autocorrélation d'ordre k. On a utilisé de tels graphiques (fig. 1.6 et 1.7) dans l'étude du lac Huron. Ce diagramme permet de comprendre la dépendance de la série par rapport à son passé. Il donne une vision locale de la série : y a-t-il une corrélation entre la série à un instant et la série 1, 2 ... instants avant ? Evidemment, si une telle dépendance apparaît, on doit en comprendre le mécanisme : s'agit-il d'une simple corrélation empirique sans pendant inférentiel ou bien d'une autocorrélation empirique qui traduit une corrélation entre variables aléatoires ?
L'extension de cette représentation à plusieurs décalages consécutifs s'obtient à l'aide de la fonction `lag.plot()`. Illustrons l'intérêt de ce graphique sur deux séries présentant des saisonnalités marquées mais structurellement différentes.
Considérons (fig. 1.8) d'une part, la série des températures mensuelles moyennes à Nottingham Castle analysée par Anderson (1976), fournie avec R ; d'autre part,

une série de même longueur, de saisonnalité 12, simulée [4] suivant le modèle :

$$y_t - 50 = 0.9(y_{t-12} - 50) + z_t - 0.7z_{t-1}, \tag{1.3}$$

où les z_t sont i.i.d. $\mathcal{N}(0, 17.5)$. Voyons d'abord ce que décrit ce modèle. La valeur 50 retranchée à toutes les dates semble fonctionner comme une moyenne et effectivement, comme ce sera rappelé au chapitre 4, c'est la moyenne de la série. Ensuite l'équation nous dit que l'écart à la moyenne à une date t dépend (a) fortement de l'écart à la moyenne 12 mois avant et (b) de la variation de la série entre $t-2$ et $t-1$ par l'intermédiaire du terme $-0.7z_{t-1}$. Les z_t sont une suite de v.a. normales indépendantes de moyenne 0, de variance 17.5 [5]. Anticipant encore sur le chapitre 4, signalons que y_t dans (1.3) suit ce qu'on appelle un modèle SARMA$(0,1)(1,0)_{12}$. La simulation de (1.3) fait l'objet d'un exercice du chapitre 7.
L'aspect saisonnier est manifeste sur le chronogramme de la série des températures (fig. 1.8 haut), alors qu'on le voit peu sur le chronogramme, très fluctuant, de la série simulée (fig. 1.8 bas).

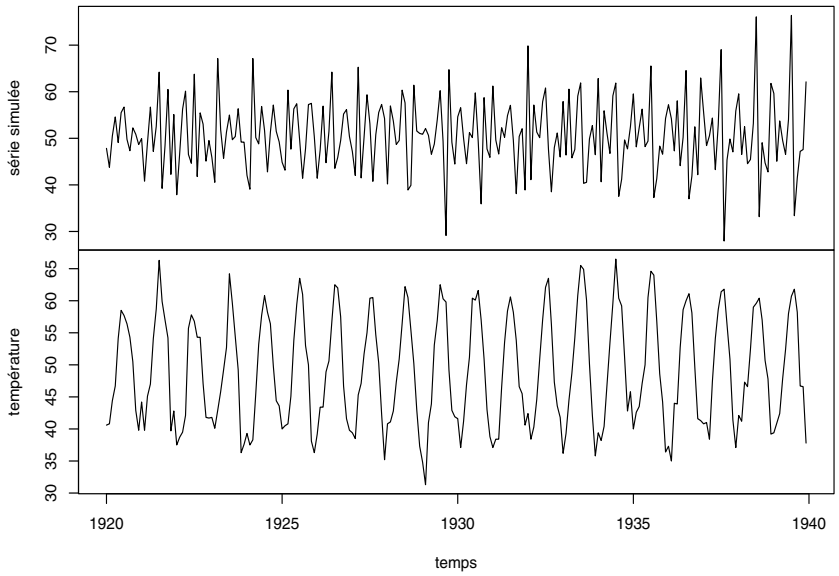

Fig. 1.8 – Chronogramme de deux séries avec saisonnalité de 12.

Examinons maintenant les diagrammes retardés de ces séries (fig. 1.9 et 1.10) (retards 1 à 12). Le graphique suivant est obtenu grâce à la commande :

```
> lag.plot(rev(nottem),12,layout=c(4,3),do.lines=FALSE,diag.col="red",
+     col.main="blue")
```

4. La simulation de séries temporelles est étudiée au chapitre 7.
5. Les paramètres ont été choisis pour que la série théorique ait approximativement les mêmes variance et moyenne que la série des températures à Nottingham Castle.

On notera dans le code ci-dessus la fabrication d'indices dans une légende, à l'aide de `expression()` qui permet d'une façon générale d'écrire des formules mathématiques dans les graphiques. Nous n'avons pas demandé le lag plot de `nottem` mais celui de `rev(nottem)` c'est-à-dire celui de la série *retournée* [6].

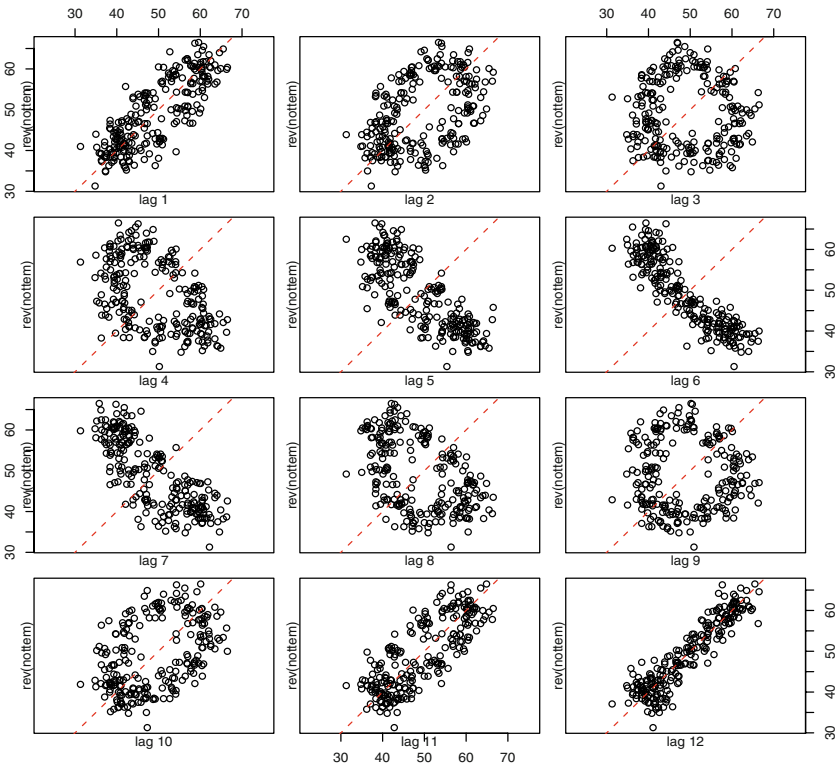

Fig. 1.9 – Lag plot de la série des températures à Nottingham Castle.

Ce passage par la série retournée est nécessaire car `lag.plot()` utilise la fonction `lag()` qui ne donne pas la série retardée mais la série avancée et, si l'on veut, en abscisse la valeur en une date t et en ordonnée la valeur en une date $t+k$, $k>0$, il faut retourner la série.
Ces lag plots montrent des aspects très typés, notamment aux décalages (ou retards) 1, 5, 6, 11, 12. Pour les deux séries on note que la valeur à une date est fortement liée à la valeur 12 périodes avant. La commande, pour la série simulée, est :

```
> lag.plot(rev(y),12,layout=c(4,3),ask=NA,do.lines=FALSE,diag.col="red")
```

6. Soit une série $y_t, t=1,\cdots,T$, la série retournée est $w_t = y_{T+1-t}$. Elle s'obtient par la fonction `rev()`.

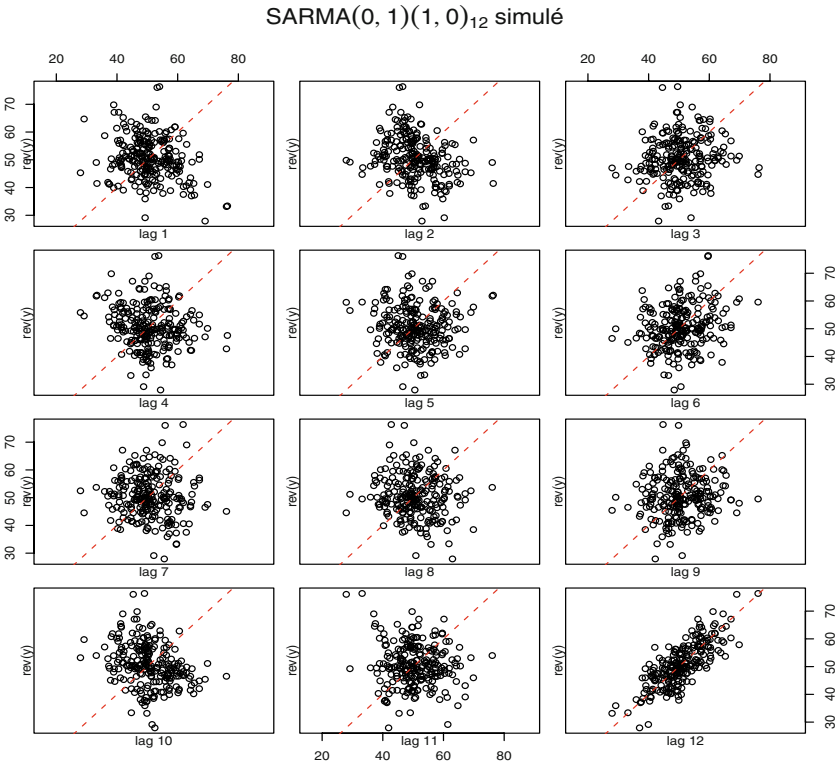

Fig. 1.10 – Lag plot de la série simulée : SARMA$(0,1)(1,0)_{12}$.

On observe une forte corrélation négative au retard 1, due en fait au terme $-0.7z_{t-1}$ dans (1.3). Pour celle des températures, le nuage des valeurs retardées a une forme d'anneau aux décalages 2, 3, 4 et 8, 9, 10, alors que le nuage de la série simulée est informe. La forme circulaire pour la série des températures s'explique assez facilement. Examinons par exemple le décalage 3. Le nuage est formé de points correspondant aux couples de mois donnés au tableau 1.1.

Tableau 1.1 – Température à Nottingham, décalage 3, couple de mois (abscisse, ordonnée) représentés sur le lag plot.

abscisse	jan	fev	mar	avr	mai	jun
ordonnée	avr	mai	jun	jul	aout	sept
abscisse	jul	aout	sept	oct	nov	dec
ordonnée	oct	nov	dec	jan	fev	mar

Les points dont l'abscisse est en février, mars, avril et mai ont des ordonnées plus chaudes que les abscisses, et ces points se trouvent donc au-dessus de la première

bissectrice. Les points proches de la bissectrice sont des couples de mois de températures proches et basses, ou proches et élevées. Enfin, les points en bas à droite sur l'anneau sont des points dont l'abscisse est plus chaude que l'ordonnée. Concluons que la forme allongée au retard 12 traduit une autocorrélation (c'est-à-dire une corrélation entre la série et la série décalée de 12 mois) de nature empirique, qui n'a pas de contrepartie théorique, mais qui exprime simplement le fait que le soleil chauffe la terre très régulièrement à Nottingham Castle : les températures d'un mois donné sont voisines d'une année à l'autre. La discussion de ces séries à travers leurs month plots sera poursuivie à propos de la notion de stationnarité (chap. 4).

1.2.3 Month plot

Un month plot (qu'on pourrait traduire approximativement par « chronogramme par mois ») est une représentation simultanée des chronogrammes des séries associées à chaque saison, appelée conventionnnellement *mois*. Il dessine un chronogramme de la sous-série correspondant à chaque niveau de la saison (mois de l'année, jour de la semaine...). Les points du chronogramme du mois m correspondent à la série du mois m, pour toutes les années. C'est une représentation pertinente pour étudier la saisonnalité d'une série, qu'elle soit mensuelle ou autre. Il faut évidemment indiquer la saisonnalité au logiciel. Quand on définit, chapitre 2, une série par `ts()`, on doit préciser sa saisonnalité : c'est elle que retient `monthplot()` ; mais si on applique `monthplot()` à un vecteur numérique, on indique la saisonnalité par l'option `frequency=`.
Examinons les month plots de la série `nottem` et de la série simulée (fig. 1.11).

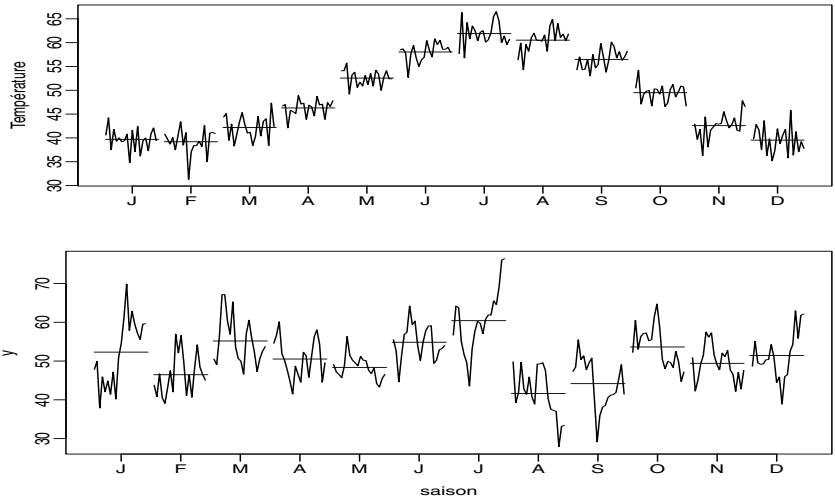

Fig. 1.11 – Month plots : de la température à Nottingham (haut), du SARMA simulé (bas).

Pour `nottem` le code est :

```
> monthplot(nottem,ylab="Température",main="",cex.main=1)
```

On peut observer pour le SARMA une forte fluctuation intra-mois autour de la moyenne du mois, ce qui suggère que les moyennes intra-mois ne sont pas significativement différentes. Alors que pour `nottem`, les moyennes intra-mois sont très différentes et les fluctuations intra-mois autour de la moyenne faibles, les moyennes intra-mois, si elles sont définies [7], sont significativment différentes.
Le trait horizontal dans chaque chronogramme de saison indique la moyenne de la saison. Les niveaux moyens par mois sont assez proches pour la série simulée, mais très différents pour la série des températures, ce qui recoupe les observations précédentes sur les lag plots. Pour cette série, le bruit autour de chaque niveau est faible, alors qu'il est élevé pour la série simulée. On retrouve la très forte variabilité des températures intersaison. Le month plot suggère que la température fluctue autour d'une fonction trigonométrique de période 12. On envisagera, chapitre 9, trois modélisations de cette série. L'une est la somme d'une moyenne exprimée par une fonction trigonométrique et d'un bruit résiduel ayant une dynamique également modélisée alors que la série simulée ne se prête à aucun ajustement par une fonction déterministe du temps. Pour aller plus loin dans la compréhension des différences entre ces deux séries, il faudra les examiner après avoir révisé la stationnarité (chap. 4). Nous rencontrerons un month plot au cours de l'étude du trafic mensuel à l'aéroport de Toulouse-Blagnac (chap. 8) et de celle de la collecte de lait (chap. 11). Un month plot permet d'avoir une vision simultanée des comportements de toutes les saisons.

Graphiques et observations manquantes

Dans beaucoup de séries, des valeurs manquent *régulièrement* à certaines dates. Ainsi il n'y pas de cotations le samedi et le dimanche à la Bourse. Il faut dans ce cas adopter une échelle de temps qui ne montre pas de manquants les samedis et dimanches. D'autre part, certains jours autres que samedi ou dimanche se trouvent exceptionnellement sans cotation ; ainsi à certaines dates on rencontre des manquants *occasionnels*. Deux attitudes sont alors possibles : (1) pour conserver la régularité du temps, remplacer les manquants occasionnels par une moyenne, même approximative, ou (2) ignorer les éventuels manquants irréguliers : c'est le choix de R pour la série `EuStockMarkets` examinée au chapitre 2. Ce peut être un choix maladroit. Par exemple, supposons que, pour une série quotidienne, il y ait un effet lundi, alors, dès le 1er manquant, la régularité est faussée et on trouve des lundis à des intervalles de temps irréguliers. Si l'on veut conserver la régularité des dates, il faut donc traiter ces manquants dès le début quitte à affiner leur traitement ultérieurement. Si ces manquants occasionnels sont peu nombreux, on peut envisager leur remplacement à la main. Nous verrons au chapitre 2 les outils proposés par R pour traiter les manquants.

7. C'est-à-dire si la saisonnalité est de nature déterministe et non stochastique.

1.3 Tendance, saisonnalité, résidus

Les exemples précédents l'ont montré : quand on étudie une série temporelle, on doit s'intéresser généralement à plusieurs aspects.

- On examine d'abord son comportement global : est-elle, d'un point de vue macroscopique, croissante, décroissante... ?
- Ensuite, on peut être amené à repérer un éventuel comportement saisonnier : y a-t-il dans cette série un élément qui revienne plus ou moins régulièrement toutes les k observations ?
- Enfin, une fois cernés les deux aspects précédents, on peut s'attacher au comportement à court terme, local de la série, notamment en vue de sa prévision.

Il est commode de se donner une présentation synthétique de ces trois aspects. C'est ce qu'on appelle *décomposition saisonnière* d'une série en une tendance, une composante saisonnière et une composante irrégulière. On peut décrire ainsi ces composantes :

Tendance. La tendance ou trend m_t capte l'orientation à long terme de la série.

Composante saisonnière. La composante saisonnière s_t capte un comportement qui se répète avec une certaine périodicité (toutes les 12 périodes pour des données mensuelles, toutes les 7 périodes pour des données quotidiennes...).

Composante irrégulière. C'est une composante d'erreur, u_t. Idéalement, elle est de faible variabilité par rapport aux autres composantes.

A ces trois composantes, on ajoute parfois un cycle.

Cycle. On appelle *cycle* un comportement qui se répète assez régulièrement mais avec une périodicité inconnue et changeante. Nous n'examinerons pas dans ce livre de séries présentant un cycle.

La décomposition peut être additive, multiplicative ou combiner les deux aspects.

$$y_t = m_t + s_t + u_t, \qquad \text{où } \mathsf{E}(u_t) = 0 \tag{1.4}$$

$$y_t = m_t \; s_t \; u_t, \qquad \text{où } \mathsf{E}(u_t) = 1 \tag{1.5}$$

$$y_t = (m_t + s_t) \times u_t, \qquad \text{où } \mathsf{E}(u_t) = 1 \tag{1.6}$$

Des séries montrant une saisonnalité qui s'accentue (cas des ventes de champagne par exemple) sont souvent mieux ajustées par un modèle multiplicatif que par un modèle additif.

Dans R, on peut obtenir des décompositions de série par les fonctions : `decompose()` et `stl()`. La fonction `decompose()` est une approche très rudimentaire décrite par exemple dans Brockwell & Davis (2002) au chapitre 1. Nous la retrouverons lors de l'étude du trafic passager à l'aéroport de Blagnac (chap. 8.2). La fonction `stl()`, décrite dans la référence de son aide en ligne, est basée sur la régression sur polynômes locaux. Les packages **pastecs** et **mFilter** offrent également de nombreuses possibilités.

1.4 Etapes et objectifs de l'analyse d'une série temporelle

On commence toujours par faire une étude descriptive de la série, en s'appuyant notamment sur des graphiques. Suivant l'objectif poursuivi, on est amené ensuite à lisser la série, la modéliser, la prédire. Parcourons maintenant ces étapes en envisageant quelques moyens de les traiter.

Décrire, préparer. Nous avons décrit un certain nombre de graphiques qui permettent d'explorer une série préalablement à toute modélisation ou encore de détecter les éventuelles anomalies, les aspects systématiques, phénomènes qu'il faut traiter avant de pouvoir utiliser des méthodes bien formalisées. Il est donc parfois nécessaire de préparer la série.

On commence par corriger la série des éventuels effets systématiques (par exemple, le nombre de jours ouvrables dans une série hebdomadaire d'une production), ainsi que des effets occasionnels (grèves, pannes de machine...). On impute des valeurs aux dates où manque l'observation de la série. Notons que les valeurs atypiques susceptibles de fausser la modélisation peuvent parfois être traitées comme des manquants.

On calcule aussi les statistiques descriptives usuelles : moyenne, variance, coefficients d'aplatissement et d'asymétrie.

Résumer. Dans certains cas, on cherche une vue synthétique débarrassée de détails de court terme ; c'est souvent un besoin des instituts de statistique officielle. Les méthodes qui donnent une vue d'ensemble de la série, nette de fluctuations de court terme ou saisonnières, sont appelées *méthodes de lissage*.

Modéliser. Expliquer la valeur en un instant, ou parfois la variance de la valeur, par des modèles à peu de paramètres.

– Modèle sans variable explicative. On explique le présent de la série par son propre passé :

$$y_t = f(y_{t-1}, y_{t-2}, \cdots) + u_t. \tag{1.7}$$

C'est la solution quand on ne dispose pas d'information sur les variables susceptibles d'agir sur la variable d'étude. Si les erreurs u_t sont de moyenne nulle, de variance constante et non corrélées deux à deux, on se trouve en présence d'un modèle autorégressif. Le modèle (1.3) est un exemple de modèle autorégressif, précisément un SARMA$(0,1)(1,0)_{12}$, suivant la terminologie du chapitre 4.

– Modèle avec variables explicatives

$$y_t = f(X_t) + u_t, \tag{1.8}$$

X_t est un vecteur de variables explicatives qui peut contenir des valeurs retardées de y_t, et u_t une erreur présentant généralement une autocorrélation, donc susceptible elle-même d'une modélisation du type (1.7). Dans tous les cas, la fonction $f(.)$ doit être estimée dans une classe de fonctions. On ne considère

ici que la classe des fonctions linéaires. Le modèle du niveau du lac Huron (1.1 + 1.2), les modèles de consommation d'électricité (chap.10) et le modèle (9.4) de la température à Nottingham Castle (chap. 9) entrent dans cette catégorie. Nous n'envisageons pas ici le cas où X_t dépendrait de valeurs $y_t, y_{t-1}, \cdots$.

Prévoir. La prévision de valeurs à des dates futures, le présent et le passé de la série étant connus, peut être (1) basée sur un modèle, ou bien (2) être construite sans ajustement préalable d'un modèle, cas du lissage exponentiel et de ses généralisations.

On distingue souvent la prévision de la prédiction. La prévision d'un phénomène mobilise un ensemble de moyens (statistiques, avis d'experts...), alors que la prédiction se limite à un aspect purement statistique. Ici, on emploie les deux termes indifféremment, mais ce livre ne concerne évidemment que la prédiction.
Retenons que l'expression *série temporelle* souvent abrégée en *série* peut désigner indifféremment : (1) une série finie de valeurs numériques indicées par le temps sans arrière-plan aléatoire, (2) une suite infinie de v.a. indicées par le temps (on parle alors également de *processus aléatoire* ou *processus stochastique*), (3) l'observation sur un intervalle de temps d'une telle suite de v.a. ; on appelle souvent *trajectoire* cette observation de longueur finie. Le contexte permet en général de lever toute ambiguïté sur la nature de la série étudiée. La série simulée considérée plus haut est une trajectoire : elle est l'observation, sur un intervalle de temps, d'une série obéissant à (1.3). Etudier une série temporelle peut consister à l'explorer, soit sans chercher à expliciter un mécanisme aléatoire qui pourrait la gouverner, soit en mettant ce mécanisme en évidence, pour prédire ou non son évolution. On peut considérer qu'une trajectoire est pour une série temporelle ce qu'un échantillon d'observations i.i.d. est pour une v.a..

Remarques (A propos de l'étude des séries)

1. *Variabilité*. La variabilité est une notion purement empirique. Une série a une certaine variabilité qu'on peut mesurer, par exemple, au moyen de sa variance empirique. Quand on décompose saisonnièrement une série, chaque partie de la décomposition a une variabilité, variabilité qui dépend de la technique de décomposition adoptée. Quand on modélise une série, on peut attacher une variabilité au résidu de la modélisation et, en comparant cette variabilité à celle de la série initiale, mesurer l'intérêt, le pouvoir explicatif de la modélisation. Ainsi pour les séries de population (fig. 1.1) on a vu qu'une grande part de variabilité s'explique par un modèle de régression sur le temps. Pour la série du champagne, une régression linéaire sur le temps ne capterait qu'une faible part de la variabilité de la série, alors qu'une modélisation de la dynamique de la série rendra compte d'une bonne part de sa variabilité. Dans l'exemple du lac Huron, on décompose la série en une tendance linéaire et une erreur dont on a identifié la dynamique et qu'on peut donc prédire. La variabilité de la série est décomposée en celle de sa moyenne et celle de l'erreur u_t qui elle-même peut être décomposée. Ces décompositions de variabilité sont rarement additives.

2. Le choix d'un modèle ou d'un autre, l'incorporation ou non d'une composante, peuvent s'apprécier d'après le graphique de la série et tirer leur validation de l'analyse elle-même.
3. Il n'y a pas une unique façon d'obtenir, pour une série donnée, une décomposition saisonnière telle que (1.4) ou (1.5). Plusieurs méthodes sont possibles pour réaliser ces étapes, accolées ou non à un modèle. Certaines, comme celle de Box-Jenkins, peuvent prendre en compte la saisonnalité, mais sans isoler la composante saisonnière.
4. Une même série temporelle peut être analysée de différentes façons, suivant l'objectif poursuivi, les capacités du statisticien, les outils informatiques à sa disposition, le temps dont il dispose pour faire le travail... Rien n'interdit de superposer plusieurs approches.

   Il arrive qu'une même série accepte de fait plusieurs traitements théoriquement incompatibles. Par exemple, la série de la température moyenne à Nottingham Castle (chap. 9), est modélisée avec une saisonnalité stochastique puis avec une saisonnalité déterministe, sans qu'un des choix apparaisse indiscutablement supérieur à l'autre. On avancera deux explications à cette ambiguïté : d'une part, les séries qu'on étudie sont nécessairement finies et des caractéristiques de long terme peuvent ne pas se manifester sur l'intervalle de temps étudié ; d'autre part, une série n'est pas nécessairement modélisée par un modèle de notre catalogue.
5. Si l'objectif est de prédire, on peut être amené à prédire chaque composante de la série, puis à rassembler ces résultats pour prédire la série. Brockwell & Davis (2002) donnent un exposé très accessible et concis de la décomposition d'une série. Gourieroux & Monfort (1995) décrivent les filtres de moyenne mobile et la méthode X11, méthode de référence en statistique officielle. Ladiray & Quenneville (2001) présentent en détail cette méthode et d'autres outils de désaisonnalisation.

1.5 Opérateur retard

La manipulation pratique ou théorique des séries temporelles se trouve considérablement simplifiée par l'usage de l'opérateur retard (*Lag operator*). On donne ici ses propriétés élémentaires et des exemples de sa manipulation dans R.

Opérateur retard. On note indifféremment B (backwards) ou L (lag), l'opérateur qui fait passer de x_t à x_{t-1} :

$$\mathrm{B}x_t = x_{t-1}.$$

On a :

$$\mathrm{B}^2 x_t = \mathrm{B}(\mathrm{B}x_t) = \mathrm{B}x_{t-1} = x_{t-2}.$$

Opérateur différence. La différence première est :

$$\Delta x_t = (1 - \mathrm{B})x_t = x_t - x_{t-1},$$

c'est la série des accroissements, alors que la différence seconde Δ^2 donne la série des « accroissements des accroissements ». On a :

$$\Delta^2 x_t = \Delta(\Delta x_t) = (1 - \mathrm{B})^2 x_t = (1 - 2\mathrm{B} + \mathrm{B}^2) x_t = x_t - 2x_{t-1} + x_{t-2}.$$

Opérateur différence saisonnière. Etant donné une série mensuelle, il peut être important d'en examiner les accroissements d'une année sur l'autre (janvier sur janvier...). L'opérateur différence saisonnière $\Delta_{12} = 1 - \mathrm{B}^{12}$ est utile dans ce cas.

$$\Delta_{12} x_t = (1 - \mathrm{B}^{12}) x_t = x_t - x_{t-12}.$$

L'opérateur retard simplifie grandement l'écriture des équations relatives aux séries. Il permet d'écrire une équation de récurrence comme un polynôme de l'opérateur retard appliqué à une série. Ses propriétés théoriques sont étudiées dans de nombreux ouvrages, par exemple : Hamilton (1994, chap. 2), Gourieroux & Monfort (1995).
Nous retrouverons des manipulations semblables au chapitre 4, dans des simulations au chapitre 7 ou dans l'estimation de modèles avec dérive ou avec tendance, comme au chapitre 8. Le calcul dans R de séries retardées et différenciées est examiné au chapitre 2.

Exercice 1.1
1 désigne la fonction constante, et t la fonction $t \mapsto t\ \forall t$, calculer :
(a) $(1 - 0.9\mathrm{B})1$,
(b) $(1 - 0.9\mathrm{B})t$,
(c) $\frac{1}{1-0.9\mathrm{B}} t$.

Exercice 1.2
Ecrire le modèle SARMA (1.3) à l'aide de l'opérateur retard.

Chapitre 2

R pour les séries temporelles

Nous présentons ici quelques notions de R indispensables dans la manipulation des séries temporelles. Nous examinons en particulier le traitement des dates ainsi que des structures classiques de séries temporelles régulières ou irrégulières. Quelques aspects concernant le traitement des manquants et des valeurs atypiques sont également abordés.

De nombreux et bons manuels pour l'apprentissage de R sont disponibles : Cornillon *et al.* (2010), Paradis (2005) ou Robinson & Schloesing (2008). Cette dernière référence insiste sur l'examen de la structure des objets [1].

On trouvera d'autres références sur le site de R, CRAN (2010a). Les présentations synoptiques de R et d'autres logiciels (Hiebeler 2010 ; Muenchen 2009) facilitent grandement l'apprentissage de l'un de ces logiciels par référence à un logiciel déjà connu.

Pour les séries temporelles, le site associé à Shumway & Stoffer (2006) contient une solide introduction aux fonctions de R ; la carte de référence de Ricci (2010) et les notes de Farnsworth (2008) offrent une présentation synthétique très pratique. Le panorama CRAN (2010b) donne un aperçu des packages consacrés aux séries temporelles.

La vignette `Anx2`, qu'on appelle par `vignette("Anx2")`, donne quelques notions élémentaires sur la recherche d'aide, l'examen et l'exploitation de la structure d'un objet.

1. Ces notes sont écrites dans un style très vivant ; le lecteur y est associé à l'examen des sorties du logiciel. De bonnes questions pratiques y sont soulevées.

2.1 Dates et heures

2.1.1 Considérations générales

Temps universel - Temps atomique. Deux mesures du temps coexistent. Le temps atomique international (TAI), qui est une échelle de temps basée sur la définition de la seconde, élaborée à l'aide d'horloges atomiques ; il est d'une très grande régularité. Le temps universel coordonné (UTC pour Universal Time Coordinated), qui est la base légale de l'heure dans le monde ; il définit le jour comme la durée moyenne de rotation de la Terre autour de son axe. Or, la rotation de la Terre n'est pas constante : elle ralentit sous l'effet des marées et d'irrégularités imprévisibles. La durée des jours UTC augmente donc très lentement en moyenne par rapport au TAI. Pour maintenir une certaine proximité entre les deux temps, on ajuste l'échelle de temps UTC en insérant des *secondes intercalaires* [2].

Mesure du temps. L'heure POSIX (*POSIX timestamp*) est une mesure utilisée principalement dans les systèmes qui respectent la norme POSIX (Portable Operating System Interface) [3]. Il s'agit du nombre de secondes écoulées depuis le 1er janvier 1970 à `00:00:00` UTC jusqu'à l'événement à dater, hors secondes intercalaires. Pour chaque seconde écoulée, l'heure POSIX s'accroît d'exactement une unité, et ce de façon invariable tout au long de l'année, sauf lorsque l'IERS décide d'ajouter (ou de supprimer) une seconde intercalaire ; des anomalies peuvent alors survenir. Si la date du 1er janvier 1970 à `00:00:00` UTC correspond à l'heure 0 de POSIX, comme cette heure augmente régulièrement toutes les secondes depuis cette date, il est clair que l'écriture d'une date quelconque est illisible dans ce repère [4]. Nous verrons les formats dont dispose R pour lire des dates qui ne sont pas données en heure POSIX ou pour écrire dans un format qui nous est familier des dates connues en heure POSIX.

Wapedia (2010) présente les variantes de temps universel utilisées, ainsi que quelques anomalies de l'heure POSIX dues aux secondes intercalaires. Notons enfin que GMT et UTC sont équivalents.

Dates et heures. La notation standard internationale des dates est `YYYY-MM-DD` où `YYYY` est l'année dans le calendrier grégorien habituel, `MM` le mois de l'année entre 01 (janvier) et 12 (décembre) et `DD` le jour du mois entre 01 et 31.

La notation standard internationale pour le temps dans un jour est `hh:mm:ss` où `hh` est le nombre d'heures complètes passées depuis minuit (00-24), `mm` le nombre de minutes complètes passées depuis le début de l'heure (00-59) et `ss` le nombre de secondes complètes depuis le début de la minute (00-60). Si la valeur de l'heure est 24, alors les valeurs des minutes et des secondes doivent être 0. Ainsi, l'heure PO-

2. Ces secondes sont insérées à l'initiative du service international de la rotation terrestre et des systèmes de référence (IERS).

3. POSIX est la norme des systèmes Unix pour les interfaces utilisateurs et logiciels, et R, langage développé sous Unix à l'origine, respecte cette norme.

4. On parle parfois de date-heure quand une date renvoie à un jour dans une année et à une heure dans ce jour.

SIX, une date-heure en réalité, se note `YYYY-MM-DD hh:mm:ss` avec éventuellement une indication de zone de temps.

La valeur 60 pour `ss` est nécessaire quand une seconde intercalaire doit être insérée dans l'échelle de temps universel coordonné. Sans autre précision, une date est supposée être mesurée en UTC, qui correspond à GMT, mais on utilise aussi des dates locales et on doit alors préciser la zone de temps du lieu. Le fuseau horaire, ou la zone de temps du territoire européen de la France, est UTC+1 en hiver, code de zone de temps CET (Central European Time), et UTC+2 pendant l'heure d'été, code CEST (Central European Summer Time).

Kuhn (2010) donne un résumé simple des représentations internationales du temps et des dates.

2.1.2 Dates et heures en R

R mesure le temps à l'aide de l'heure POSIX. La fonction `strptime()` assure la conversion entre représentation comme chaîne de caractères (*string*) et date-heure POSIX. Si on ne spécifie pas le format d'une date fournie à R, il considère qu'elle est donnée sous la forme internationale `%Y-%m-%d`, c'est-à-dire : année sur 4 positions–mois sur 2 positions–jour sur 2 positions, les champs étant séparés par des « - ».

Il existe deux classes de date POSIX dans R : `POSIXlt` (`l` pour *liste*) et `POSIXct` (`c` pour *caractère*). Commençons par examiner la date du système, fournie par `Sys.time()`.

```
> (z=Sys.time())

[1] "2011-01-31 11:00:38 CET"

> str(z)

 POSIXct[1:1], format: "2011-01-31 11:00:38"

> format(z,"%a %b %d %X %Y %Z")

[1] "lun. janv. 31 11:00:38 2011 Paris, Madrid"
```

Le jour et l'heure sont donnés dans la représentation internationale, la zone de temps est indiquée. Cette date est un objet de type `POSIXct`. La fonction `format()` permet de modifier la présentation de la date. Peut-on extraire les composantes : année, minute... de cette date ? On voit que c'est un objet à une seule composante. Voyons ce que donne `unlist()` sur cet objet.

```
> unlist(z)

[1] "2011-01-31 11:00:38 CET"
```

Les éléments restent groupés dans la même chaîne de caractères ; mais on peut accéder aux minutes... par `minutes()` et par les autres fonctions manipulées dans les exemples de la section. Exprimons maintenant `z` dans la classe `POSIXlt` et voyons l'effet de `unlist()` sur cette classe.

```
> ll=as.POSIXlt(z,"CET")
> str(ll)

 POSIXlt[1:9], format: "2011-01-31 11:00:38"

> unlist(ll)

    sec     min    hour    mday     mon    year    wday
 38.677   0.000  11.000  31.000   0.000 111.000   1.000
   yday   isdst
 30.000   0.000
```

On obtient maintenant un vecteur dont chaque élément est une composante de la date. Ce résultat demande quelques précisions et `?POSIXct` les complètera. La composante `mon` est le nombre de mois *après* le 1er mois de l'année, `mon` est donc compris entre 0 et 11. `year` est l'année depuis 1900, `isdt` est un indicateur de l'heure d'été, il vaut 0 car la mesure a lieu en hiver. Observons enfin que `sec` est donné au centième de seconde.
Une heure POSIX est stockée comme un entier. Par exemple 1268736919 correspond à une heure POSIX, pour une certaine origine et une zone de temps. Si l'on sait qu'on est dans la zone de temps UTC et que l'origine est le 1er janvier 1970, on écrira cette date en clair par

```
> as.POSIXct(1268736919,origin="1970-01-01",tz="GMT")

[1] "2010-03-16 10:55:19 GMT"
```

Il existe d'autres types de dates moins étroitement liées au calendrier : ainsi les séries mensuelles sont considérées comme des séries à temps régulier dont les observations sont espacées de 365/12 jours (section 2.2).
Examinons maintenant sur des exemples la notion de date-heure : comment importer, définir une date, comment extraire des éléments de date (mois, jour de la semaine...).

Exemple 2.1 Dans la vie quotidienne, nous lisons les dates comme des chaînes de caractères, mais dans un ordinateur une date (sans heure associée) est un entier. La fonction `as.Date()` lit une chaîne de caractères comme une date.

```
> cat(str("2008-05-16")," ",str(as.Date("2008-05-16")),"\n")

 chr "2008-05-16"
Class 'Date'  num 14015
```

On est passé d'une chaîne de caractères à un objet de type numérique. La fonction `as.Date()` a un argument optionnel pour préciser le format. On peut définir des dates régulièrement espacées par `seq.Date()` et même en remontant le temps :

```
> x="1970-01-01"
> as.numeric(as.Date(x))

[1] 0

> st=as.Date("1999-09-17")
> en=as.Date("2000-1-7")
> (ll=seq(en,st,by="-1 month"))
```

```
[1] "2000-01-07" "1999-12-07" "1999-11-07" "1999-10-07"
[5] "1999-09-07"

> str(ll)

Class 'Date'  num [1:5] 10963 10932 10902 10871 10841

> xs=strptime(x,"%Y-%m-%d")
> str(xs)

 POSIXlt[1:9], format: "1970-01-01"

> unlist(xs)

  sec   min  hour  mday   mon  year  wday  yday isdst
    0     0     0     1     0    70     4     0     0

> str(as.Date(x))

Class 'Date'  num 0
```

`x` est une chaîne de caractères, on la convertit en date, sans heure, par `as.Date()`. Ensuite on définit deux dates `st` et `en`, puis `ll` la suite décroissante de dates de mois en mois, de la plus grande à la plus petite. Par `strptime()`, nous exprimons `x` comme une heure POSIX en précisant le format de la date et éventuellement la zone de temps. Nous voyons, par `unlist()`, que cette heure POSIX revient bien à une date avec 0 minute, 0 seconde...
`help.search("date")` donne toutes les fonctions concernant les dates contenues dans les packages installés sur la machine.

Exemple 2.2 (Lecture de dates dans un fichier importé) Ces dates se présentent le plus souvent comme une chaîne de caractères. Examinons le fichier `Tel_extrait.csv` créé par un autocommutateur qui recense toutes les communications demandées par les postes téléphoniques d'une entreprise ; la date contient dans l'ordre le jour, le mois et l'année. Elle est suivie de l'heure de début d'appel, du code de destination de l'appel, de cette destination en clair, de la durée en secondes et du montant de l'appel en euros.

```
Date.app;H.deb.app;Code.Dest;Desti.Dét;Dur.app.sec;Mont.app.EU
01/06/2006;09:03:51;1393;Serbie & Monténégro;387;3,5475
02/06/2006;17:28:46;1485;Royaume-Uni;1081;0,990917
02/06/2006;18:50:54;1488;Suède;31;0,028417
03/06/2006;15:41:50;1491;Allemagne;8;0,007334
12/04/2006;15:50:41;315550;Haute-Garonne;42;0,0182
```

On le lit par :

```
> require(caschrono)
> don.mois1=read.csv2(file=system.file("/import/Tel_extrait.csv",
+                 package="caschrono"),skip=0,stringsAsFactors=FALSE)
```

Le séparateur de champs étant le ";", on utilise `read.csv2()`. En ne précisant pas un format pour la date et l'heure, on a choisi de les lire comme des chaînes de caractères. Le choix `stringsAsFactors = FALSE` évite que les entiers ou les chaînes ne soient transformés en facteurs. On a tout intérêt à examiner la structure de l'objet obtenu en sortie d'une fonction d'importation de données, notamment pour voir si les noms de variable ont été correctement pris et quels sont les types des différentes variables compris par R.

```
> str(don.mois1,width=60,strict.width="cut")

'data.frame':        5 obs. of  6 variables:
 $ Date.app   : chr  "01/06/2006" "02/06/2006" "02/06/2006..
 $ H.deb.app  : chr  "09:03:51" "17:28:46" "18:50:54" "15:..
 $ Code.Dest  : int  1393 1485 1488 1491 315550
 $ Desti.Dét  : chr  "Serbie & Monténégro" "Royaume-Uni" "..
 $ Dur.app.sec: int  387 1081 31 8 42
 $ Mont.app.EU: num  3.5475 0.99092 0.02842 0.00733 0.0182
```

On voit que la virgule décimale a été comprise correctement et que la date a été lue comme une chaîne de caractères. Il faut maintenant convertir cette chaîne en date, la chaîne qui décrit l'heure en heure et éventuellement combiner les deux en une date-heure.

Construction d'une date et d'une date-heure

Nous passons maintenant en revue quelques opérations sur les dates et heures qu'on vient de lire.

Récupération de la date. Nous convertissons la chaîne de caractères donnant la date en une date par `as.Date()` ; il faut donc préciser le format de la date dans le fichier .csv : jour, mois, année sur 4 chiffres et séparateur des champs `/` :

```
> date=as.Date(don.mois1$Date.app,"%d/%m/%Y")
> str(date)

Class 'Date'  num [1:5] 13300 13301 13301 13302 13250
```

Rappelons qu'une date est un nombre entier, la valeur 0 correspondant au 1er janvier 1970.

Récupération simultanée de la date et de l'heure par `chron()`. En consultant la page d'aide de `chron`, on voit que, fort logiquement, cette fonction a comme arguments le format des données en entrée, à transformer en date-heure, et le format choisi pour la représentation de ces dates ; on ne le précise pas s'il est identique au premier.

```
> require(chron)
> x=chron(dates=don.mois1$Date.app,times=don.mois1$H.deb.app,
+          format=c(dates="d/m/Y",times ="h:m:s"))
> x[1:2]

[1] (01/06/06 09:03:51) (02/06/06 17:28:46)
```

Récupération simultanée de la date et de l'heure comme heure POSIX. Fabriquons le vecteur des chaînes de caractères date-heure puis convertissons le par `strptime()` :

```
> dh=paste(don.mois1$Date.app,don.mois1$H.deb.app)
> xp=strptime(dh,"%d/%m/%Y %H:%M:%S")
> xp[1:2]

[1] "2006-06-01 09:03:51" "2006-06-02 17:28:46"
```

Extraction des composantes. Des fonctions permettent l'extraction des composantes d'un objet de type `chron` ou POSIX :

```
> options(digits=12)
> dates(x)[1:4]

[1] 01/06/06 02/06/06 02/06/06 03/06/06

> times(x)[1:4]

Time in days:
[1] 13300.3776736 13301.7283102 13301.7853472 13302.6540509

> hours(xp);minutes(xp);seconds(xp)

[1]  9 17 18 15 15

[1]  3 28 50 41 50

[1] 51 46 54 50 41

> quarters(xp);quarters(x)

[1] "Q2" "Q2" "Q2" "Q2" "Q2"

[1] 2Q 2Q 2Q 2Q 2Q
Levels: 1Q < 2Q < 3Q < 4Q

> months(xp)[1:3]; days(xp)[1:3]; weekdays(xp)[1:3]

[1] "juin" "juin" "juin"

[1] 1 2 2
31 Levels: 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < ... < 31

[1] "jeudi"    "vendredi" "vendredi"

> years(xp)

[1] 2006 2006 2006 2006 2006
Levels: 2006
```

`times()` extrait l'heure en fraction de jour depuis le 01/01/1970, `dates()` et `times()` ne s'appliquent pas à une heure POSIX, `hours()`, `minutes()` et `seconds()` permettent d'extraire des éléments de date. Notons que `quarters()`, `months()`, `days()`, `weekdays()` appliquées à un objet de type `chron` en donnent le trimestre, le mois, le jour dans le mois et le jour dans la semaine, *comme facteurs*, alors qu'appliqués à une heure POSIX, ils donnent ces éléments comme *chaînes de caractères*.

Extraction naïve des éléments de date. On peut alternativement décomposer l'heure à partir de la chaîne de caractères correspondante, récupérer l'année, le mois... sans considération de structures de date ou d'heure,

```
> heure0=don.mois1$H.deb.app
> heure=substr(heure0,1,2)
> minute=substr(heure0,4,5)
> seconde=substr(heure0,7,8)
> an=substr(don.mois1$Date.app,7,10)
> an.num=as.numeric(an)
```

mais on se prive ainsi de toute structure de date ou de date-heure.

Les packages **fCalendar**, **Hmisc**, **pastecs**, **timeSeries**, **zoo**, entre autres, contiennent des fonctions pour manipuler des dates.

2.2 Les structures de séries temporelles dans R

Une série temporelle unidimensionnelle est un vecteur à chaque élément duquel est associée une date ou une date-heure, ou si l'on préfère, une série temporelle est une suite de couples (date, valeur), ordonnée sur la date. Une série temporelle multidimensionnelle est une matrice à chaque ligne de laquelle est associée une date ou une date-heure. Mais on peut envisager plusieurs définitions de la date. Schématiquement, on trouve des structures de séries temporelles comme `ts` et `mts`, où la date est approximativement celle du calendrier, et dans **its** et **timeSeries** notamment, des structures de séries temporelles étroitement associées à une date. Les exemples suivants illustrent ces aspects.

2.2.1 La fonction `ts()`

La fonction `ts()` fait partie de **stats**, chargé au lancement de R. Elle permet de créer des séries temporelles à temps régulier. `?ts` fournit la syntaxe et de très utiles exemples. Illustrons la fonction en fabriquant différentes séries temporelles. Des principes de simulation sont développés au chapitre 7.

Exemple 2.3 (Série temporelle mensuelle) Fabriquons une série mensuelle supposée commencer en février 2005 à partir d'un vecteur `x` obtenu par simulation de 30 valeurs indépendantes d'une normale de moyenne 2, de variance 1, et dont on ne conserve que 3 décimales. L'unité de temps est l'année et il y a 12

observations par unité de temps. Février est la deuxième observation dans une unité de temps, d'où la syntaxe :

```
> set.seed(493)
> x=2+round(rnorm(15),3)
> x.ts=ts(x,start=c(2005,2),frequency=12)
> x.ts
```

```
       Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep
2005       1.545 0.980 1.162 3.208 2.666 2.739 0.413 0.560
2006 0.886 1.354 3.141 2.948
       Oct   Nov   Dec
2005 1.115 3.298 0.830
2006
```

Examinons la structure de `x.ts` :

```
> str(x.ts)
```

```
 Time-Series [1:15] from 2005 to 2006: 1.54 0.98 1.16 3.21 2.67 ...
```

Augmentons maintenant le nombre de décimales de la représentation des nombres à l'écran, afin de voir comment le temps de `x.ts` est mesuré :

```
> options(digits=12)
> time(x.ts)
```

```
               Jan            Feb            Mar
2005                2005.08333333 2005.16666667
2006 2006.00000000 2006.08333333 2006.16666667
               Apr            May            Jun
2005 2005.25000000 2005.33333333 2005.41666667
2006 2006.25000000
               Jul            Aug            Sep
2005 2005.50000000 2005.58333333 2005.66666667
2006
               Oct            Nov            Dec
2005 2005.75000000 2005.83333333 2005.91666667
2006
```

```
> frequency(x.ts)
```

```
[1] 12
```

```
> cycle(x.ts)
```

```
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
2005       2   3   4   5   6   7   8   9  10  11  12
2006   1   2   3   4
```

Ainsi : (1) `time()` donne le temps en fraction d'unité de temps à partir du numéro de l'unité initiale ; l'accroissement du temps d'une fraction sur la suivante est ici $0.08333333 = 1/12$, c'est-à-dire l'inverse de la fréquence déclarée ; (2) `frequency()` donne évidemment le nombre d'observations par unité de temps ; et (3) `cycle()`

permet de repérer la position de chaque observation dans une unité de temps. Remarquons sur `time(x.ts)` que pour les mois de janvier, la partie décimale est nulle ; ainsi, 0.08333333 est le temps écoulé depuis le début de l'année, *avant* l'observation de février.

Exemple 2.4 (Série multidimensionnelle) On peut définir simultanément plusieurs séries sur le même intervalle de temps. Fabriquons `y` en simulant 30 valeurs indépendantes d'une normale de moyenne 10, de variance 0.25 et en ne conservant que 3 décimales, puis définissons la série mensuelle bidimensionnelle de composantes `x` et `y`, commençant en février 2005 :

```
> set.seed(8515)
> y=10+0.5*round(rnorm(30),3)
> xy.ts=ts(cbind(x,y),start=c(2005,2),frequency=12)
> str(xy.ts)

 mts [1:30, 1:2] 1.54 0.98 1.16 3.21 2.67 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:2] "x" "y"
 - attr(*, "tsp")= num [1:3] 2005 2008 12
 - attr(*, "class")= chr [1:2] "mts" "ts"

> xy.ts[1:4,]

         x       y
[1,] 1.545  9.7715
[2,] 0.980 11.0025
[3,] 1.162 10.1980
[4,] 3.208 10.4780
```

La fonction `ts()` crée un objet de type `ts` ou `mts` selon que la série est uni ou multidimensionnelle.

Exemple 2.5 (Série quotidienne) On tire un vecteur `z` de 900 observations indépendantes suivant une loi uniforme. Transformons-le en une série temporelle quotidienne commençant le 5 mars 2005 et couplons à cette série celle des dates calendaires exactes, objet `date.z` ci-dessous. Il y a, aux années bissextiles près, 365 jours dans une année, et le 5 mars 2005 est le 64^{e} jour de l'année. D'où la syntaxe :

```
> set.seed(571)
> z=round(runif(900),digits=4)+5
> z.ts=ts(z,start=c(2005,64),frequency=365)
> time(z.ts)[1:8]

[1] 2005.17260274 2005.17534247 2005.17808219 2005.18082192
[5] 2005.18356164 2005.18630137 2005.18904110 2005.19178082

> num=31+28+4
> (frac=num/365)
```

```
[1] 0.172602739726
```

L'objet `num` est le nombre de jours écoulés du début de l'année au commencement de la série (voir l'exemple 1) et `frac` la fraction d'unité de temps correspondante. On peut parallèlement dater la série quotidienne `z` :

```
> date.z=seq(as.Date("2005-03-05"),length=900,by="day")
> date.z[1:4]

[1] "2005-03-05" "2005-03-06" "2005-03-07" "2005-03-08"
```

En résumé, le temps de `ts()` ne s'exprime pas obligatoirement en dates calendaires. Si on exprime le temps en jours de calendrier, à cause des années bissextiles et des mois de longueurs inégales, on perd la notion de temps régulier. C'est pourtant une simplification très pratique pour beaucoup de séries, notamment pour les séries financières, comme on le voit sur l'exemple qui suit. Mais il est évidemment possible, comme on vient de le faire, de conserver une série de dates exactes parallèlement à une série de type `ts`.
Pour des données quotidiennes présentant une saisonnalité hebdomadaire, on pourra avoir intérêt à choisir `frequency=7` dans la définition de la série, tout en conservant la date exacte dans une autre variable. Rien n'empêche de faire démarrer une série à la date 1 et d'en définir une saisonnalité quelconque. Ainsi l'ordre `ts(z,start=c(1,1),frequency=9)` pourrait dater `z`, série du nombre d'appels téléphoniques envoyés à chaque heure de la journée par une administration travaillant de 8 h à 17 h.

Exemple 2.6 (Données boursières) La Bourse des valeurs est fermée les samedis et dimanches. Mais le « temps des affaires » présente une régularité que `ts()` peut prendre en compte. Examinons le jeu de données `EuStockMarkets` qui fait partie de **datasets**. Cet objet contient les valeurs à la fermeture des grands indices boursiers européens. Les données sont échantillonnées en *business time*, c'est-à-dire en omettant les samedis, dimanches et jours fériés.

```
> str(EuStockMarkets)

 mts [1:1860, 1:4] 1629 1614 1607 1621 1618 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:4] "DAX" "SMI" "CAC" "FTSE"
 - attr(*, "tsp")= num [1:3] 1991 1999 260
 - attr(*, "class")= chr [1:2] "mts" "ts"
```

Examinons la composante CAC :

```
> CAC40=EuStockMarkets[,"CAC"]
> str(CAC40)

Time-Series [1:1860] from 1991 to 1999: 1773 1750 1718 1708 1723 ...

> frequency(CAC40)

[1] 260
```

... et voyons quel temps y est associé :

```
> options(digits=12)
> time(CAC40)[c(1:3,1856:1860)]

[1] 1991.49615385 1991.50000000 1991.50384615 1998.63076923
[5] 1998.63461538 1998.63846154 1998.64230769 1998.64615385
```

Les cotations à la Bourse ont lieu à peu près 5 jours par semaine et approximativement 52 semaines dans l'année, soit 260 jours. L'unité de temps est donc ici l'année, la fréquence est le nombre d'observations par unité de temps, et l'accroissement de temps d'une observation à la suivante est égal à $0.00384615 = 1/260$ année. La première observation de la série se situe fin juin 1991.

2.2.2 Récupération de données boursières et séries de classe `its`

Nous voulons récupérer le CAC40 et les cours à la fermeture de trois sociétés cotées à Paris : la Société générale, Danone et L'Oréal, du 3 janvier 2006 au 30 juin 2009. Il faut d'abord trouver les codes de ces séries sur http://fr.finance.yahoo.com/ ; ce sont dans l'ordre :

```
^FCHI, GLE.PA, BN.PA, OR.PA
```

Ceci fait, on peut écrire le code pour récupérer les cours. Les cotations n'ayant pas lieu les samedis, dimanches et certains jours fériés, le temps associé n'est pas régulier. On utilise `priceIts()` de **its** (*irregular time series*). Le code ci-dessous récupère les données sur le site de Yahoo Finance et calcule les rendements composés de ces cours par `returns()` de **timeSeries**. `str(r.csdl)` nous montre que les dates sont conservées comme noms des lignes du fichier. Cours et rendements seront ensuite accessibles dans `csdl` du package **caschrono**.

```
> deb="2006-01-01"; fin="2009-06-30"
> instru=c("^FCHI","GLE.PA","BN.PA","OR.PA")
> csdl=priceIts(instrument=instru,start=deb,end=fin,quote="Close")
> colnames(csdl@.Data)=c("Cac40","Socgen","Danone","L_Oréal")
> aa=returns(csdl,percentage=TRUE)
> r.csdl=aa[complete.cases(aa)==TRUE,]
```

Examinons la structure d'une série récupérée par `priceIts()`.

```
> cac40=priceIts(instrument="^FCHI",start=deb,end=fin,quote="Close")
> str(cac40,width=60,strict.width="cut")

Formal class 'its' [package "its"] with 2 slots
  ..@ .Data: num [1:890, 1] 4755 4777 4839 4835 4867 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:890] "2006-01-02" "2006-01-03" "2006-..
  .. .. ..$ : chr "^FCHI Close"
  ..@ dates: POSIXct[1:890], format: "2006-01-02" ...
```

Elle est de classe `its` et on note que les dates de cotation font partie de la structure. Les composantes de l'objet `cac40` sont associées à des `@`; ce sont des encoches (*slots*) typiques d'un objet de classe S4. Le slot `cac40@.Data` contient les valeurs et le slot `cac40@dates`, les dates en heures POSIX. On trouve également les dates comme noms de ligne du slot `.Data`. La fonction `union()` appliquée à deux objets de classe `its` crée un objet de même classe qui contient autant de couples (date, série) qu'il y en a dans les objets réunis, et autant de dates/lignes qu'il y en a de distinctes dans les objets, alors que `intersect()` ne conserve que les sous-séries des dates communes.

```
> socgen=priceIts(instrument="GLE.PA",start=deb,end=fin,quote="Close")
> cacsoc=union(cac40,socgen)
> cs=intersect(cac40,socgen)
```

`rangeIts()` permet de sélectionner une sous-série entre deux dates :

```
> deb2="2008-02-01"
> wxx=rangeIts(cac40,start=deb2,end=fin)
```

A chaque observation d'une série temporelle correspond une date qui peut être celle du calendrier ou une date « sur mesure ».
En plus de **chron**, **base** et **its**, plusieurs packages proposent des structures de séries temporelles. **zoo**, comme **its** est dédié aux séries irrégulières. Nous retrouverons **its** au chapitre 12 consacré aux séries conditionnellement hétéroscédastiques. CRAN (2010b) donne une liste commentée de ces possibilités.

2.3 Série de classe `ts` : retard, différence, union, intersection, sous-série

On fabrique une série retardée à l'aide de `lag()` ou `Lag()` de **Hmisc**. Ces opérateurs sont la version numérique de l'opérateur retard B.
Attention ! Comme on l'a déjà indiqué (section 1.2) `lag()` fonctionne à l'inverse de ce que l'on attend : elle change la date de la série. De plus, elle ne s'applique qu'à des séries temporelles, alors que `Lag()` fonctionne comme on s'y attend et peut s'appliquer également à un vecteur non daté. L'action de retarder une série n'a de sens bien défini que pour des séries régulières.
Illustrons ces opérations sur la série des expéditions mensuelles de vin de Champagne stockée sous forme de fichier texte dans **caschrono**, à raison d'une année d'observation par ligne ; c'est une série à temps régulier, l'unité est la bouteille de 75 cl. Nous lisons les données

```
> aa=scan(system.file("/import/champagne_2001.txt",package="caschrono"))
```

et fabriquons la série temporelle de ces ventes, exprimées en milliers de bouteilles de 75 cl

```
> champa.ts=ts(aa/1000,start=c(2001,1),frequency=12)
```

Nous fabriquons maintenant des séries retardées de 1, de 12 et la série multidimensionnelle réunissant ces séries sur la date

```
> chL1=lag(champa.ts,-1)
> chL12=lag(champa.ts,-12)
> dd=ts.union(champa.ts,chL1,chL12)
> dd[c(1:2,12:13), ]
```

```
     champa.ts      chL1     chL12
[1,] 12083.682        NA        NA
[2,] 10295.468 12083.682        NA
[3,] 39829.010 43074.507        NA
[4,] 14140.749 39829.010 12083.682
```

Ainsi, `ts.union()` donne une série multidimensionnelle à temps régulier avec des manquants aux dates sans valeur associée dans l'une des séries. `window()` permet de zoomer sur une partie de la série

```
> window(dd,start=c(2002,10),end=c(2003,2))
```

```
         champa.ts      chL1     chL12
Oct 2002 36590.986 26369.199 34035.035
Nov 2002 46285.449 36590.986 43074.507
Dec 2002 41122.275 46285.449 39829.010
Jan 2003 16529.190 41122.275 14140.749
Feb 2003 12813.937 16529.190 12936.294
```

On peut faire l'intersection de deux séries par la date

```
> dd0=ts.intersect(champa.ts,chL1,chL12)
> dd0[1:3,]
```

```
     champa.ts      chL1     chL12
[1,] 14140.749 39829.010 12083.682
[2,] 12936.294 14140.749 10295.468
[3,] 16935.343 12936.294 14777.224
```

`ts.intersect()` ne conserve que les dates où les différentes séries sont observées simultanément.

2.4 Traitement des manquants

Une série temporelle régulière est une série de points (date, valeur) à des dates régulièrement espacées. Mais il peut arriver que les valeurs associées à des dates manquent ou que certaines soient aberrantes ou atypiques. Les valeurs exceptionnelles faussent les calculs qui font intervenir des moyennes sur les observations, c'est-à-dire à peu près toute la statistique. Les points dont la valeur est manquante ne peuvent être simplement ignorés ; on perdrait ainsi la régularité temporelle de la série, situation particulièrement ennuyeuse pour une série présentant une saisonnalité. Il est donc indispensable, pour réaliser le traitement statistique d'une série temporelle, d'identifier les points manquants ou exceptionnels et de leur attribuer (imputer) des valeurs plausibles. Il faut donc notamment trouver les dates de tels points. R offre différentes fonctions pour repérer et traiter les manquants.

Remarque
Beaucoup de fonctions n'acceptent pas de manquants dans la série à traiter, mais disposent de l'option `na.rm` (*remove not available*) pour les ignorer. Supposons que `x` soit un vecteur avec des manquants et qu'on veuille faire la moyenne des non-manquants : on la calcule par `mean(x,na.rm=TRUE)`.

Repérage des manquants

Les manquants, surtout dans des données saisies manuellement, sont parfois indiqués par des codes conventionnels comme `9999` ou `-1`. Il est prudent de remplacer de telles valeurs par des « manquants système » notés NA (Not Available) dans R, à l'aide de `is.na()`. Ainsi on ne risque pas de leur appliquer des opérations arithmétiques.

Exemple élémentaire

```
> (xx=c(0:3,99,7,99))

[1]  0  1  2  3 99  7 99

> which(xx==7)

[1] 6

> is.na(xx)=(xx==99)
> xx

[1]  0  1  2  3 NA  7 NA

> which(is.na(xx)==TRUE)

[1] 5 7
```

Par `which()` on repère les éléments qui prennent la valeur 7, le 6e ici, puis on remplace tous les éléments de `xx` valant 99 par une valeur manquante. D'une façon générale, `which()` fournit les numéros des observations vérifiant une certaine condition.
Illustration de l'usage de `na.omit()` et `na.action()` pour une matrice ou un data frame. On fabrique un data frame `DF` où il y a deux lignes avec manquants et on le convertit en une matrice `m`.

```
> DF=data.frame(x=c(1,2,3),y=c(0,10,NA),z=c(NA,1,2))
> (DF1=na.omit(DF))

  x  y z
2 2 10 1

> m=as.matrix(DF)
> (m1=na.omit(m))
```

```
     x  y z
[1,] 2 10 1
attr(,"na.action")
[1] 3 1
attr(,"class")
[1] "omit"
```

```
> (imq=na.action(m1))
```

```
[1] 3 1
attr(,"class")
[1] "omit"
```

Par `na.omit()`, on élimine les lignes avec manquant. Qu'elle soit appliquée au data frame ou à la matrice, `na.omit()` fournit ici l'objet formé de la seule ligne sans manquant, mais n'indique pas quelles lignes ont été éliminées. `na.omit()` appliquée à la matrice fournit également, par `na.action()`, les numéros des lignes éliminées. Il est clair qu'en séries temporelles, il faut éviter d'éliminer des observations manquantes.

Repérage d'une valeur manquante sur les cours de la Bourse. On a noté (section 2.2.2) qu'il y a des manquants dans certains cours. Pour repérer les dates où un cours (de type `its`) manque, nous utilisons `complete.cases()` et pour obtenir les dates où il y a un manquant, `which()` :

```
> manq=!complete.cases(csdl)
> i.manq=which(manq==TRUE)
> (date.manq=csdl@dates[i.manq][1:3])
```

```
[1] "2006-04-14 CEST" "2006-04-17 CEST" "2006-05-01 CEST"
```

La variable `manq` est une variable logique qui prend la valeur `TRUE` aux dates où manque au moins une des séries, donc `csdl[manq,]` est la sous-série où manque au moins un cours. `i.manq` est la liste des numéros d'observations où il y a au moins un manquant, `date.manq` celle des dates correspondantes. Notons que le package **its** ne considère pas les samedis et dimanches comme des manquants.

Repérage d'une valeur exceptionnelle. L'examen du chronogramme est le moyen de base pour repérer les éventuels points exceptionnels. On repère avec précision la date d'un tel événement à l'aide de `which()` et on peut ensuite affecter une valeur raisonnable à cette date, manuellement ou à l'aide d'une fonction de R. A titre d'illustration, examinons le cours de l'action Essilor à la Bourse de Paris, code EI.PA, de 2006 à 2009, inclus dans le package **caschrono** et téléchargé sur Yahoo Finance.

```
> data(essil)
> plot(essil,ylab="cours")
```

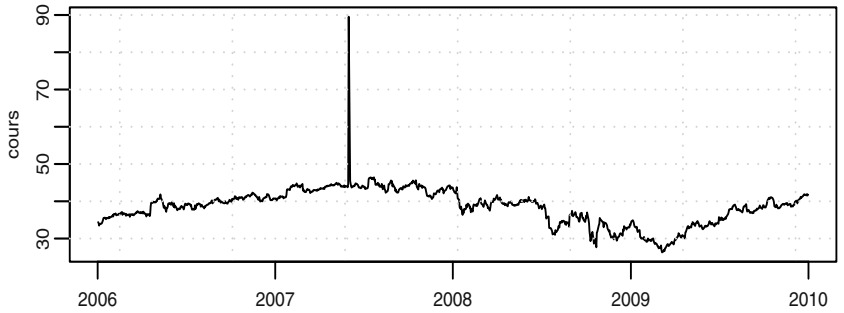

Fig. 2.1 – Action Essilor de janvier 2006 à décembre 2009.

Nous repérons une valeur aberrante, très supérieure à 50 (fig. 2.1). Il faut la remplacer par une valeur raisonnable. D'abord, cherchons la date de cette observation.

```
> i0=which(essil@.Data>60)
> essil@dates[i0]

[1] "2007-06-01 CEST"
```

Nous sommes maintenant en mesure d'imputer une valeur à cette observation. Cette imputation peut se réaliser de plusieurs manières.

- En affectant une valeur raisonnable, par exemple 50, qui semble être de l'ordre de grandeur des valeurs voisines :

  ```
  > essil@.Data[i0]=50
  ```

  Nous pouvons vérifier notre imputation en imprimant les valeurs voisines :

  ```
  > essil[(i0-2):(i0+2),]

  An object of class "its"
             essilor
  2007-05-30   43.78
  2007-05-31   44.65
  2007-06-01   50.00
  2007-06-04   45.47
  2007-06-05   44.67
  Slot "dates":
  [1] "2007-05-30 CEST" "2007-05-31 CEST" "2007-06-01 CEST"
  [4] "2007-06-04 CEST" "2007-06-05 CEST"
  ```

- En faisant une interpolation linéaire. Utilisons pour cela `na.approx()` de **zoo** qui travaille sur une série à temps irrégulier et traite les manquants par interpolation linéaire ou spline. L'observation exceptionnelle d'indice `i0` est d'abord remplacée par NA :

  ```
  > essil@.Data[i0]=NA
  ```

Puis nous formons une série à temps irrégulier de type `zoo` en juxtaposant les vecteurs des données et des dates ; enfin `na.approx()` calcule l'interpolation linéaire.

```
> essil@.Data[i0]=NA
> z=zoo(essil@.Data,essil@dates)
> z.corr=na.approx(z)
> z.corr[(i0-2):(i0+2),]

2007-05-30 2007-05-31 2007-06-01 2007-06-04 2007-06-05
    43.780     44.650     44.855     45.470     44.670
```

On peut observer que l'approximation calculée est $44.855 = 44.650 + (45.470 - 44.650)/4$, interpolation basée sur l'espacement des dates : il y a en effet 4 jours entre les deux dates de cours non manquant. Or, les 2 et 3 juin 2007 correspondant à un week-end, l'interpolation pourrait tout aussi légitimement être calculée par $44.650 + (45.470 - 44.650)/2$.

L'approximation linéaire pour remplacer des manquants présente quelques limites. D'abord, il faut supposer que la série est approximativement linéaire autour de la valeur manquante, ensuite cette technique n'est pas valable pour des manquants au bord de la série. Alternativement, si l'on a identifié la dynamique de la série, il est possible d'imputer une valeur prédite d'après cette dynamique et non par interpolation linéaire.

Chapitre 3

Régression linéaire par la méthode des moindres carrés

Nous supposons que la méthode des moindres carrés ordinaires est connue et les rappels que nous donnons ici sont simplement destinés à fixer la terminologie. Nous illustrons la méthode par la régression d'une consommation mensuelle d'électricité sur des variables de température. Or les résidus obtenus ont une dynamique dont il faut tenir compte. C'est pourquoi nous compléterons le traitement de ces données, au chapitre 10 une fois rappelés les modèles ARMA.

3.1 Principe de la régression linéaire

La régression linéaire décompose une v.a. y en son espérance (ou moyenne), exprimée linéairement en fonction d'autres variables non aléatoires : 1, $x_2, x_3, \cdots, x_K$, plus une erreur aléatoire. K est donc le nombre de variables explicatives (ou covariables), constante comprise. Par exemple, si $K = 3$, nous avons le modèle linéaire suivant :

$$y_t = \beta_1 + \beta_2 x_{2t} + \beta_3 x_{3t} + u_t,\ t = 1, 2, \cdots, T. \tag{3.1}$$

Les erreurs u_t doivent vérifier un certain nombre de **présupposés** :

- **P1** être d'espérance nulle, $\mathsf{E}(u_t) = 0$;
- **P2** avoir la même variance pour tout t, $\mathsf{var}(u_t) = \sigma_u^2$;
- **P3** être non corrélées entre elles, $\mathrm{corr}(u_t, u_s) = 0$, $t \neq s$;
- **P4** être normalement distribuées.

Sous ces hypothèses, l'espérance de y_t est $\beta_1 + \beta_2 x_{2t} + \beta_3 x_{3t}$. Nous pouvons aussi écrire matriciellement le modèle et les présupposés :

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{bmatrix}, \qquad X = \begin{bmatrix} 1 & x_{21} & x_{31} \\ 1 & x_{22} & x_{32} \\ \vdots & \vdots & \vdots \\ 1 & x_{2T} & x_{3T} \end{bmatrix}, \qquad U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_T \end{bmatrix}, \qquad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}.$$

Avec ces notations (3.1) s'écrit :

$$\underset{T\times 1}{Y} = \underset{T\times K}{X}\ \underset{K\times 1}{\boldsymbol{\beta}} + \underset{T\times 1}{U} \tag{3.2}$$

et les présupposés s'expriment :

$$\mathbf{P1}: \qquad \mathsf{E}(Y) = X\boldsymbol{\beta}, \qquad\qquad \mathbf{P2}+\mathbf{P3}+\mathbf{P4}: \qquad U \sim \mathcal{N}(0, \sigma_u^2 \mathbf{I}_T),$$

où $\mathbf{I}_T$ est la matrice identité d'ordre T. On peut encore les formuler :

$$Y \sim \mathcal{N}(X\boldsymbol{\beta}, \sigma_u^2 \mathbf{I}_T). \tag{3.3}$$

La méthode des MCO (moindres carrés ordinaires) estime β par la valeur qui minimise

$$(Y - X\boldsymbol{\beta})'(Y - X\boldsymbol{\beta}),$$

où X' est la transposée de X. Le minimum est atteint pour

$$\widehat{\boldsymbol{\beta}} = (X'X)^{-1}X'Y.$$

Si les présupposés **P1**+**P2**+**P3** sont vérifiés, cet estimateur est sans biais et de variance minimum (théorème de Gauss-Markov). Il est normalement distribué de matrice de covariance $\sigma_u^2(X'X)^{-1}$:

$$\widehat{\boldsymbol{\beta}} \sim \mathcal{N}(\boldsymbol{\beta}, \sigma_u^2(X'X)^{-1}). \tag{3.4}$$

Les vecteurs des valeurs ajustées et des résidus sont respectivement définis par :

$$\widehat{Y} = \begin{bmatrix} \widehat{y}_1 \\ \widehat{y}_2 \\ \cdots \\ \widehat{y}_T \end{bmatrix} = X\widehat{\boldsymbol{\beta}}, \qquad\qquad \widehat{U} = \begin{bmatrix} \widehat{u}_1 \\ \widehat{u}_2 \\ \cdots \\ \widehat{u}_T \end{bmatrix} = Y - \widehat{Y}. \tag{3.5}$$

La variance de l'erreur est estimée sans biais par :

$$s_u^2 = \frac{1}{T-K}\sum_t (y_t - \widehat{y}_t)^2 = \frac{1}{T-K}\sum_t \widehat{u}_t^2. \tag{3.6}$$

On estime la matrice des covariances des paramètres en remplaçant dans $\sigma_u^2(X'X)^{-1}$ la variance inconnue σ_u^2 par son estimateur s_u^2. On note $s^2(\widehat{\beta}_i)$, l'estimation de la

variance de $\widehat{\beta}_i$ ainsi obtenue. C'est l'élément (i,i) de la matrice $s_u^2(X'X)^{-1}$. Rappelons que le coefficient $\widehat{\beta}_i$ est la quantité dont augmente y quand la variable x_i augmente de 1, *toutes choses égales par ailleurs* ; dans certains milieux scientifiques, on appelle d'ailleurs $\widehat{\beta}_i$, *coefficient de régression partielle* de x_i. On retrouvera ce vocabulaire dans les séries temporelles quand on étudiera la régression d'une série sur son passé (section 4.4.2). Une fois effectuée une régression, il faut s'assurer que les présupposés sur les erreurs sont vérifiés. On effectue cette vérification sur les résidus. Le diagramme de dispersion des résidus ou des résidus normalisés contre les valeurs ajustées permet de voir si la distribution de ces résidus est bien indépendante de la valeur ajustée. Le QQ-plot de normalité des résidus permet de vérifier l'hypothèse de normalité. L'examen des distances de Cook permet de voir si des observations modifient sensiblement l'équation. R offre de nombreux diagnostics, en particulier dans `lm()`, fonction de base de la régression linéaire. L'étude pratique de la régression est bien traitée dans de nombreux ouvrages, notamment : Cornillon & Matzner-Løber (2010), chapitre 4, Maindonald (2010) ou Sheather (2009).
Dans les cas que nous traiterons, **P4**, la normalité, sera systématiquement examinée, **P3**, la non-corrélation des erreurs, sera souvent rejetée. Il faudra donc comprendre le mécanisme de cette corrélation pour en tenir compte. Les outils développés aux chapitres 4 et 5 éclairent la nature de cette corrélation ; une fois la corrélation modélisée, la méthode des moindres carrés généralisés (MCG) permet de l'intégrer dans l'estimation de la régression. La connaissance du mécanisme de la corrélation des erreurs au cours du temps permet quant à elle de mieux prédire la série.

Moindres carrés généralisés

Dans la régression MCO d'une série temporelle telle que (3.1), on constate souvent que les résidus $\widehat{u}_t$ sont autocorrélés (si l'on calcule le coefficient de corrélation entre la série des résidus et la série des résidus retardés d'un instant, on obtient une valeur élevée). C'est signe que les u_t le sont eux-mêmes, donc **P3** ne tient pas. Dans d'autres cas, le chronogramme de la série y_t montre que les y_t ont une variance non constante (cas d'hétéroscédasticité), donc **P2** ne tient pas. Si les erreurs sont conjointement normalement distribuées $\mathcal{N}(0,\Omega)$, le modèle devient en notation matricielle :

$$Y \sim \mathcal{N}(X\boldsymbol{\beta}, \Omega) \tag{3.7}$$

où Ω est la matrice des covariances de l'erreur. La méthode des MCG fournit l'estimation de $\boldsymbol{\beta}$:

$$\widetilde{\boldsymbol{\beta}} = (X'\Omega^{-1}X)^{-1}X'\Omega^{-1}Y, \qquad \widetilde{\boldsymbol{\beta}} \sim \mathcal{N}(\boldsymbol{\beta}, (X'\Omega^{-1}X)^{-1}). \tag{3.8}$$

On ne connaît généralement pas la matrice Ω, mais l'étude des résidus de l'ajustement MCO permet d'en découvrir la structure. Dans ce livre consacré aux séries

temporelles, les résidus de la méthode des MCO montrent une dynamique dont on s'inspire pour définir la matrice Ω. Cette approche est illustrée au cours du chapitre 10 qui poursuivra l'exemple suivant, centré sur les MCO, et qui laisse de côté la corrélation manifeste entre les résidus.
Même si les présupposés sont vérifiés, il n'est pas sûr qu'une régression linéaire sur un certain ensemble de données soit pertinente, significative. On dispose heureusement d'outils pour apprécier l'intérêt d'une régression linéaire.

3.2 Significativité de la régression

Le *coefficient de détermination*, R^2, noté dans les sorties, `Multiple R-squared`, est défini de façon unique quand il y a une constante dans la régression par

$$R^2 = \frac{\sum_{i=1}^{T}(\widehat{y}_i - \overline{y})^2}{\sum_{i=1}^{T}(y_i - \overline{y})^2} = 1 - \frac{\sum_{i=1}^{T}\widehat{u}_i^2}{\sum_{i=1}^{T}(y_i - \overline{y})^2}. \tag{3.9}$$

C'est le rapport de la variabilité expliquée par la régression : $\sum_{i=1}^{T}(\widehat{y}_i - \overline{y})^2$ sur la variabilité totale des y_i : $\sum_{i=1}^{T}(y_i - \overline{y})^2$. Le R^2 augmente avec le nombre de variables explicatives. Ce n'est donc pas un indicateur pertinent de significativité de la régression. Il n'est intéressant que pour comparer des régressions portant sur un même nombre d'observations et de variables.
Le *coefficient de détermination ajusté*, `Adjusted R-squared`, est :

$$R^2_{adj} = 1 - (1 - R^2)\frac{T-1}{T-K}.$$

A T donné, c'est une fonction décroissante de K, le nombre de variables explicatives, constante comprise. L'effet de l'augmentation du nombre de variables sur le R^2 est pénalisant pour la valeur de ce coefficient.
On peut calculer un R^2 pour tout ajustement d'un modèle linéaire de l'espérance, qu'il soit obtenu par MCO ou par toute autre méthode. Les deux expressions de R^2 dans (3.9) ne coïncident généralement pas en dehors de la méthode des MCO avec constante dans la régression. Si l'on tient compte de l'autocorrélation des erreurs, et c'est un sujet majeur dans ce livre, on estime l'espérance de y par MCG ; la variabilité expliquée par la régression peut se calculer par $\sum_{i=1}^{T}(\widetilde{y}_i - \overline{\widetilde{y}})^2$ où $\widetilde{y}_i$ désigne l'estimation MCG de l'espérance mathématique de y_i, et $\overline{\widetilde{y}}$, la moyenne (empirique) de ces valeurs ajustées.

Significativité globale. La régression est dite significative, si au moins un des coefficients des variables autres que la constante est non nul. L'hypothèse nulle est : « la régression n'est pas significative », c'est-à-dire :

$H_0 : \beta_2 = \cdots = \beta_K = 0$ contre $H_1 :$ au moins un de ces coefficients est non nul.

Statistique de test. H_0 exprime une contrainte sur les paramètres. Notons $\text{SCR}_{\text{libre}}$ et $\text{SCR}_{\text{contrainte}}$ les sommes des carrés des résidus dans l'estimation libre et dans l'estimation contrainte, c'est-à-dire sous H_0. On a $\text{SCR}_{\text{contrainte}} = \sum_t (y_t - \overline{y})^2$ et $\text{SCR}_{\text{libre}} = \sum_t (y_t - \widehat{y}_t)^2$.
La statistique pour tester H_0 contre H_1 est

$$F = \frac{(\text{SCR}_{\text{contrainte}} - \text{SCR}_{libre})/(K-1)}{\text{SCR}_{\text{libre}}/(T-K)}. \tag{3.10}$$

Sous H_0, F suit une loi de Fisher à $(K-1, T-K)$ ddl : $F \sim \mathcal{F}(K-1, T-K)$. On rejette l'hypothèse nulle pour de grandes valeurs de la statistique de test.

Significativité d'un coefficient de régression. On veut tester qu'un coefficient β_i vaut une certaine valeur β_{i0} :

$$H_0 : \beta_i = \beta_{i0}$$

contre $H_1^A : \beta_i \neq \beta_{i0}$, ou contre $H_1^B : \beta_i < \beta_{i0}$, ou contre $H_1^C : \beta_i > \beta_{i0}$.
Statistique de test. Considérons l'expression de $\widehat{\beta}$, (3.4). Un sous-vecteur d'un vecteur gaussien est lui-même gaussien, donc sous H_0, $\widehat{\beta}_i \sim \mathcal{N}(\beta_{i0}, \sigma_0^2)$, où σ_0^2 est l'élément (i,i) de $\sigma_u^2 (X'X)^{-1}$, cf. (3.5).
Sous les quatre présupposés et si H_0 est vérifiée, $T_0 = (\widehat{\beta}_i - \beta_{i0})/s(\widehat{\beta}_i) \sim \mathcal{T}(T-K)$, loi de Student à $T-K$ ddl. Si $T-K > 10$, la loi de T_0 est très proche d'une loi $\mathcal{N}(0,1)$ et nous nous contenterons de cette approximation par une loi normale [1].
Sous H_0

$$T_0 = (\widehat{\beta}_i - \beta_{i0})/s(\widehat{\beta}_i) \sim AN(0,1). \tag{3.11}$$

Si l'on choisit $\beta_{i0} = 0$ et que H_0 est rejetée, on dit que β_i est significatif. En toute rigueur, T_0 suit sous H_0 une loi de Student à $T-k$ ddl. La statistique $\widehat{\beta}_i / s(\widehat{\beta}_i)$ est appelée t-statistique. Des statistiques semblables sont utilisées dans les modèles de séries temporelles, où les estimateurs suivent seulement des lois approximativement normales. Un exemple de test sur un vecteur de plusieurs paramètres figure au chapitre 12.
Nous traiterons la prédiction à la faveur de l'exemple de la section suivante : l'étude d'une consommation électrique.

Remarques
Colinéarité. On a supposé inversible la matrice $X'X$ dans 3.4 ; ceci tient quand les colonnes de X ne sont pas colinéaires. La *colinéarité exacte* survient quand une combinaison linéaire de variables explicatives est égale à une autre variable explicative, éventuellement, la constante. On rencontrera cette situation dans la modélisation du niveau moyen de `nottem` par des fonctions trigonométriques dont la somme est constante (cf. section 9.2.2). Il suffit alors d'enlever la constante ou une des variables dont la somme est constante, pour éliminer la colinéarité.

1. Voir par `vignette("Anx3")`, des compléments.

On rencontre souvent de la *colinéarité approchée* : une combinaison de variables explicatives est approximativement constante. Ceci peut survenir quand le modèle n'est pas bien spécifié ; alors les estimateurs des coefficients des variables concernées peuvent être fortement corrélés et peu significatifs. On diagnostique le problème en examinant la matrice des corrélations associée à la matrice des covariances $s_u^2(X'X)^{-1}$ des estimateurs des paramètres. Des coefficients de corrélation proches de 1 suggèrent une telle colinéarité. Le remède consiste à éliminer *une à une* les variables non significatives. L'élimination d'une variable peut alors changer dramatiquement la significativité d'autres variables.

Orthogonalité. Quand la matrice $\mathbf{X'X}$ est diagonale, les variables explicatives sont dites orthogonales. On peut alors éliminer simultanément plusieurs variables explicatives. Les fonctions trigonométriques utilisées pour modéliser la moyenne de la température (chap. 9) ou celle de la collecte de lait (chap. 11) sont orthogonales. L'orthogonalité est l'opposé de la situation de colinéarité.

3.3 Comparaison de modèles et critères d'information

En régression linéaire comme en modélisation de séries temporelles, on est amené à choisir entre différents modèles, emboîtés ou non. Il est classique d'utiliser pour cela un critère d'information.

Le principe de ces critères est le suivant. Les modèles sont estimés par maximum de vraisemblance, mais la comparaison des valeurs des vraisemblances n'est pertinente que si les modèles ont le même nombre de paramètres. Par des arguments théoriques qui peuvent être très divers, les critères d'information fournissent une fonction de l'opposé de la log-vraisemblance augmentée (pénalisée) d'une fonction croissante du nombre de paramètres contenus dans le modèle. Cette dernière peut éventuellement croître avec le nombre d'observations. Etant donné plusieurs modèles et leurs estimations, et un critère étant choisi, on retient le modèle pour lequel le critère est minimum. Nous donnons quelques détails sur deux de ces critères.

AIC (Critère d'information d'Akaike). L'AIC (Akaike's Information Criterion) est :

$$AIC(k) = -2\ln(L) + 2k,$$

où L est la fonction de vraisemblance évaluée en les estimations par maximum de vraisemblance (MV) des paramètres, et k est le nombre de paramètres estimés. (On rencontre quelquefois une autre expression de l'AIC : $(-2\ln(L) + 2k)/T$ où T est le nombre d'observations.) Dans le cas de l'ajustement par maximum de vraisemblance de (3.3), l'AIC prend la forme :

$$AIC(k) = T\ln(\widehat{\sigma}^2) + 2k,$$

où $\widehat{\sigma}^2$ est l'estimation MV de σ_U^2.

SBC (Critère bayesien de Schwartz). Le SBC (Schwartz' Bayesian Criterion)

ou BIC (Bayesian Information Criterion) est :

$$SBC(k) = -2\ln(L) + 2k\ln(T).$$

Si les erreurs sont normalement distribuées, il prend la forme

$$SBC(k) = T\ln(\widehat{\sigma}^2) + k\ln(T).$$

On voit que ces critères (AIC et SBC) sont formés de deux termes :

- le terme $-2\ln(L)$ qui est d'autant plus faible que l'ajustement par maximum de vraisemblance est bon ;
- le terme $2k$ ou $2k\ln(T)$ qui pénalise cette faible valeur en l'augmentant par une fonction croissante du nombre de paramètres estimés.

3.4 Intervalle de confiance (IC)

Un intervalle de confiance est un intervalle dont les bornes sont aléatoires et qui contient une constante avec une probabilité qu'on se fixe [2]. Par exemple, si β_i est estimé par $\widehat{\beta}_i \sim \mathcal{N}(\beta_i, \sigma_i^2)$, un IC à $100\,(1-\alpha)\%$ pour β_i est

$$\widehat{\beta}_i \pm q(1-\alpha/2)\sigma_i,$$

où $q(1-\alpha/2)$ désigne le quantile d'ordre $1-\alpha/2$ d'une v.a. $\mathcal{N}(0,1)$. Si σ_i est inconnu, on peut approcher cet IC par $\widehat{\beta}_i \pm q(1-\alpha/2)s_i$ où s_i est l'estimation de σ_i.
Si maintenant on s'intéresse à une combinaison linéaire des paramètres $c'\boldsymbol{\beta}$, avec $\boldsymbol{\beta}$ estimé par $\widehat{\boldsymbol{\beta}} \sim \mathcal{N}(\boldsymbol{\beta}, \Sigma_\beta)$, un IC à $100\,(1-\alpha)\%$ pour $c'\boldsymbol{\beta}$ est

$$c'\widehat{\boldsymbol{\beta}} \pm q(1-\alpha/2)c'\Sigma_\beta c, \quad \text{approché par} \quad c'\widehat{\boldsymbol{\beta}} \pm q(1-\alpha/2)c'\widehat{\Sigma}_\beta c. \tag{3.12}$$

3.5 Prédiction

Etant donné X_F matrice $m \times (K+1)$ de valeurs des explicatives, pour m observations vérifiant (3.3) et indépendantes des premières observations, les y associés vérifient :

$$Y_F = X_F\boldsymbol{\beta} + U_F \qquad\qquad U_F \sim \mathcal{N}(0, \sigma_u^2 I_m). \tag{3.13}$$

Le terme « prédiction » recouvre deux situations.
(1) *Estimation ponctuelle* ou *estimation par intervalle* des composantes de $\mathsf{E}(Y_F)$.
Le terme $\mathsf{E}(Y_F) = X_F\boldsymbol{\beta}$ est un vecteur de combinaisons linéaires des paramètres. On *estime* sans biais ce vecteur certain par

$$\widehat{\mathsf{E}}(Y_F) = X_F\widehat{\boldsymbol{\beta}}, \tag{3.14}$$

2. On exprime habituellement cette probabilité en pourcentage.

de matrice des covariances : $\sigma_u^2 X_F(X'X)^{-1}X_F'$. Sous l'hypothèse de normalité, on peut calculer des IC pour chacune des composantes de $\mathsf{E}(Y_F)$, comme au paragraphe précédent (expression 3.12). Ainsi, nous prenons x_f une ligne de X_F et y_f la valeur correspondante de y : l'IC à $(1-\alpha)\%$ pour $\mathsf{E}(y_f)$ est

$$x_f'\widehat{\boldsymbol{\beta}} \pm q(1-\alpha/2)\sigma_u\sqrt{x_F(X'X)^{-1}x_F'}. \tag{3.15}$$

Une représentation simultanée de ces IC est appelée *bande de confiance*.
(2) *Prédiction ponctuelle* ou *prédiction par intervalle* des composantes de Y_F.
On *prédit* le vecteur aléatoire $Y_F = X_F\boldsymbol{\beta} + U_F$ par $\widehat{\mathsf{E}}(Y_F) = X_F\widehat{\boldsymbol{\beta}}$, c'est-à-dire par la quantité qui a servi à estimer son espérance (ou moyenne.) Ce prédicteur est sans biais : $\mathsf{E}(\widehat{\mathsf{E}}(Y_F) - Y_F) = 0$. L'erreur de prédiction est $X_F\widehat{\boldsymbol{\beta}} - Y_F = X_F(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) - U_F$; sa matrice des covariances est : $\sigma_u^2(X_F(X'X)^{-1}X_F' + I_m)$, elle est évidemment plus « grande » que la matrice des covariances de l'estimateur de la moyenne, en ce sens que $(X_F(X'X)^{-1}X_F' + I_m) - X_F(X'X)^{-1}X_F' = I_m$ est définie positive. On peut calculer des intervalles de prédiction (IP) pour les composantes de Y_F. La représentation simultanée de ces IP est appelée *bande de prédiction*. L'IP à $(1-\alpha)\%$ pour y_f est

$$x_f'\widehat{\boldsymbol{\beta}} \pm q(1-\alpha/2)\sigma_u\sqrt{1 + x_F(X'X)^{-1}x_F'}, \tag{3.16}$$

σ_u inconnu est remplacé par son estimation. Le tableau 3.1 récapitule les différentes situations, diag(C) désigne la diagonale de la matrice C écrite comme une matrice colonne, et $\mathbf{I}_m$ la matrice identité d'ordre m.

Tableau 3.1 – Formules pour la prédiction en régression linéaire.

Objectif	Estimation/Prédiction		Matrice des covariances de l'erreur
	ponctuelle	par intervalle	
Est. $\mathsf{E}(Y_F)$	$X_F\widehat{\boldsymbol{\beta}}$	$X_F\widehat{\boldsymbol{\beta}} \pm q(1-\alpha/2)\sqrt{\text{diag}(A)}$	$A = \sigma_u^2 X_F(X'X)^{-1}X_F'$
Préd. Y_F	$X_F\widehat{\boldsymbol{\beta}}$	$X_F\widehat{\boldsymbol{\beta}} \pm q(1-\alpha/2)\sqrt{\text{diag}(B)}$	$B = \sigma_u^2(\mathbf{I}_m + X_F(X'X)^{-1}X_F')$

Ces questions sont traitées par la fonction `predict()` qui s'applique à un modèle en sortie de `lm()`. Cette fonction a notamment l'option `interval=` qui peut prendre les valeurs `"none"`, `"confidence"` ou `"prediction"` (cf. section 3.6).

Remarques
- Conventionnellement, on *estime* une constante et on *prédit* une v.a., mais la pratique ne respecte pas toujours cette nomenclature. L'important est de savoir la nature des structures qu'on manipule.
- On devrait prédire le vecteur aléatoire Y_F par $\widehat{\mathsf{E}}(Y_F) + \widehat{U_F}$, mais l'erreur U_F est supposée indépendante des erreurs passées, donc sa prédiction connaissant le passé est son espérance : 0. La fonction `meanf()` de **forecast** illustre cette question et il faut examiner l'exemple de son aide en ligne. En séries temporelles, on ne peut généralement pas supposer que U_F est indépendante du passé.

– Une juxtaposition d'IC à $(1-\alpha)\%$ ne constitue pas, en toute rigueur, une bande de confiance à $(1-\alpha)\%$, car les intervalles ne sont généralement pas indépendants. On ne devrait donc pas en faire une lecture globale, mais nous n'approfondirons pas cette question.
– Une bande de confiance à $(1-\alpha)\%$ pour un ensemble de n points devrait approximativement contenir $n\,(1-\alpha)$ points. Si elle en contient un nombre bien plus élevé, c'est peut-être qu'il y a trop de paramètres dans le modèle, c'est-à-dire qu'il y a sur-ajustement. Si elle en contient un nombre bien plus faible, c'est sans doute que la modélisation est mal choisie.

Pour approfondir l'ensemble de ces questions, le lecteur pourra lire avec profit Cornillon & Matzner-Løber (2010, chap. 2 et 3).

3.6 Exemple : consommation d'électricité

On étudie la dépendance de la consommation d'électricité `kwh` sur les variables décrivant la température `cldd` et `htdd`[3]. Avant tout, observons que la consommation d'électricité dépend de la température, mais que la réciproque est fausse. Les variables explicatives sont prédéterminées. D'autre part, ce n'est pas la température qui est directement utilisée comme variable explicative, mais des transformations non linéaires de cette variable. Ceci pour au moins une raison : l'augmentation de consommation d'électricité due à un degré supplémentaire un mois d'été (climatisation), n'est pas nécessairement la même que l'augmentation de consommation due à un degré de moins un mois d'hiver.
Nous allons conduire une régression linéaire par MCO. Elle montre des résidus autocorrélés. Dans ce chapitre, nous ignorons délibérément cette autocorrélation et examinons les résultats de la régression linéaire comme si les présupposés de la section 3.1 étaient vérifiés. Même si les estimations ne sont pas optimales, ce qui rend inexactes les p-values indiquées, la démarche donne des indications sur la dépendance de la moyenne par rapport aux explicatives. Le présent exemple va nous permettre également de présenter les tests classiques sur une régression linéaire et la prévision dans ce cadre.
Chargeons les séries :

```
> require(caschrono)
> data(khct)
```

et examinons leurs chronogrammes (fig. 3.1). Le temps est en année et en douzième d'année, comme on l'a expliqué au chapitre 2. La série `kwh` est croissante (à cause de l'accroissement de la population, du niveau de vie...) alors que les séries associées à la température ne le sont évidemment pas.

```
> plot.ts(khct,xlab='temps',main="",cex.lab=.9,cex.axis=.8,
+  oma.multi=c(4.5,3,.2,0),mar.multi=c(0,4,0,.5),las=0)
```

3. L'aide en ligne de `khct` de **caschrono** explique la construction de ces séries.

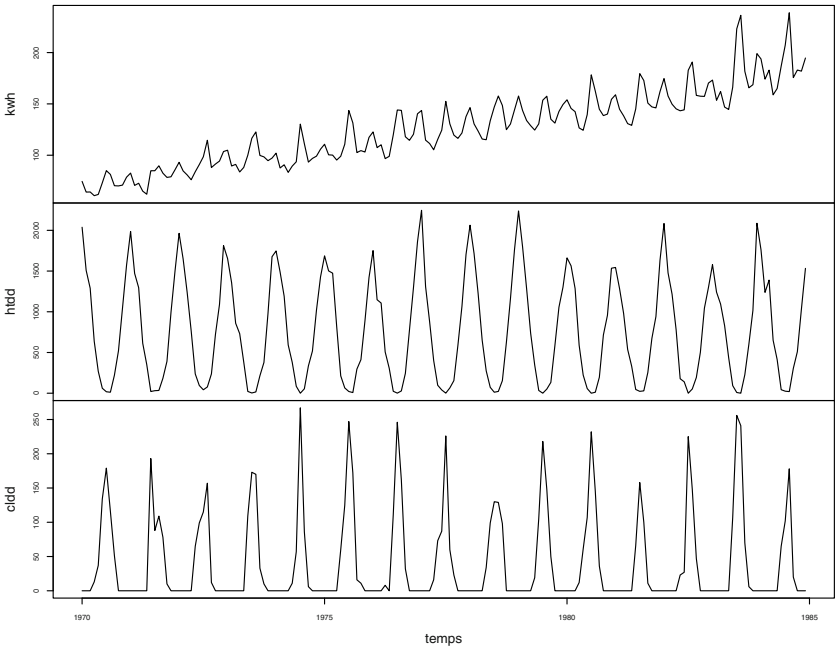

Fig. 3.1 – Chronogrammes de la consommation d'électricité et des variables de température.

Elle montre aussi une certaine hétéroscédasticité : la variabilité de la série augmente avec son niveau. On envisage deux transformations de la série (log(.) et $\sqrt{.}$) en vue de stabiliser la variance. On calcule les régressions MCO sur le temps des trois séries et on représente les chronogrammes de `kwh`, `log(kwh)` et $\sqrt{\texttt{kwh}}$ ainsi que les droites ajustées (fig. 3.2). On utilise la classe `zoo` car elle offre l'argument `panel` qui permet d'ajouter la droite de régression dans un graphe multiple de séries.

```
> ytr=cbind(khct[,"kwh"],log(khct[,"kwh"]),(khct[,"kwh"])^.5)
> colnames(ytr)=c("kwh","log(kwh)","kwh^.5")
> my.panel <- function(x, y,..., pf = parent.frame()) {
+  lines(x, y, ...)
+  abline(lm(y~x),lty=2)
+ }
> plot(zoo(ytr),ylab=c("kwh","log(kwh)",expression(sqrt(kwh))),main="",
+  xlab='temps',panel=my.panel,cex=1.2)
```

On voit que ces transformations diminuent l'hétéroscédasticité, stabilisent la variance. Mais on note que la transformation log(.) crée vers 1977 une courbure dans la série, alors que la transformation racine carrée courbe moins la série. C'est donc sur cette dernière série que nous travaillons.

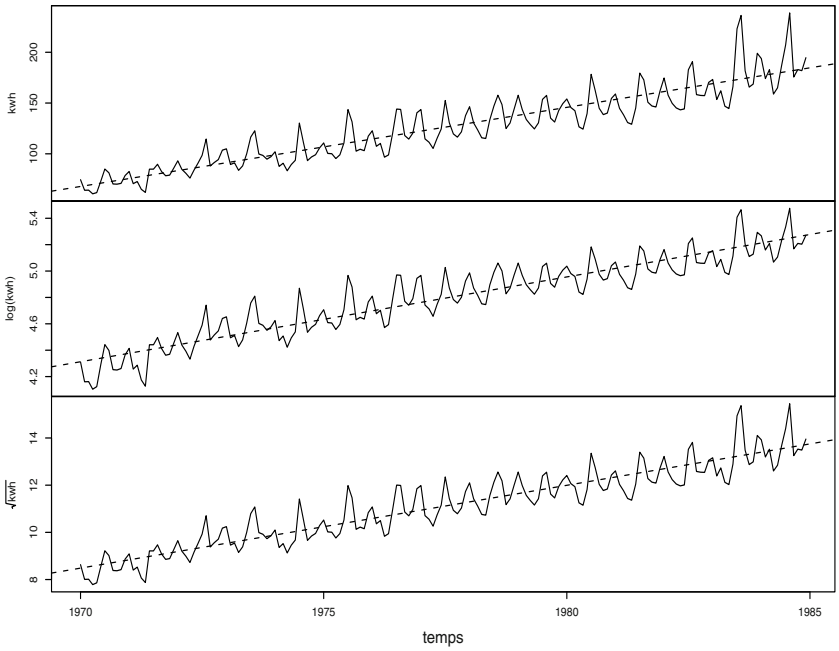

Fig. 3.2 – Chronogrammes de kwh, $\log(kwh)$ et $\sqrt{kwh}$.

Introduisons le temps comme variable explicative et considérons le modèle

$$\sqrt{\texttt{kwh}_t} = \beta_0 + \beta_1 \texttt{htdd}_t + \beta_2 \texttt{cldd}_t + \beta_3 \texttt{t1}_t + u_t,\ t = 1, \cdots, T \quad \text{(Modèle 1)} \quad (3.17)$$

où $T = 168$, est la longueur de la série diminuée de la dernière année, réservée pour comparer prévision et réalisation. On utilise `lm()` pour l'estimation de ce modèle.

```
> khct.df=as.data.frame(window(cbind(khct,time(khct)),end=c(1983,12)))
> colnames(khct.df)=c("kwh","htdd","cldd","t1")
> mod1=lm(sqrt(kwh)~htdd+cldd+t1,data=khct.df)
```

En vue de la prédiction, nous mettons en réserve la dernière année, 1984, par l'emploi de `window()`. On transforme également par racine carrée la variable dépendante et enfin on estime le modèle 1. L'objet `mod1` contient tous les résultats de l'estimation et, comme on l'a déjà indiqué, `str(mod1)` donne les noms et classes de chacun d'eux.
Le lecteur pourra consulter les notes de Maindonald (2010) ou de Verzani (2002), les ouvrages de Maindonald & Braun (2003), de Dalgaard (2008) pour une présentation détaillée de cette fonction. Cornillon & Matzner-Løber (2010) et Sheather (2009) présentent de façon très appliquée et moderne la régression par R.

Examen des résidus du modèle 1. Avant toute analyse des estimations, il faut examiner les résidus $\widehat{u}$ pour voir si les présupposés de la régression sont vérifiés.

Nous ne rappelons ici que les techniques directement utiles dans l'étude d'une série temporelle. On examinera principalement trois graphiques :
– graphique des résidus contre les valeurs ajustées ou contre le temps. Ils peuvent montrer des erreurs dans la spécification de la moyenne ;
– lag plot des résidus. Ils peuvent indiquer une éventuelle autocorrélation des erreurs u_t, donc la non vérification du présupposé **P3** (section 3.1) ;
– Q-Q plot de normalité.

Evaluons la modélisation à travers ces graphiques. Le chronogramme des résidus (fig. 3.3 haut) a une forme légèrement parabolique, de sommet situé vers 1977. C'est peut-être l'indice d'une mauvaise spécification du modèle que nous devons corriger. Nous abandonnons donc l'examen du modèle (3.17), au profit d'un modèle captant l'aspect quadratique. Pour cela nous introduisons la variable `t1.2=(t1-1977)^2` susceptible de prendre en compte ce phénomène et effectuons la régression pout t variant de $1, \cdots, T$:

$$\sqrt{\texttt{kwh}_t} = \beta_0 + \beta_1 \texttt{htdd}_t + \beta_2 \texttt{cldd}_t + \beta_3 \texttt{t1}_t + \beta_4 \texttt{t1.2}_t + u_t, \text{ (Modèle 2)} \quad (3.18)$$

Le code pour l'estimation du modèle 2 est :

```
> khct.df$t1.2=(khct.df$t1-1977)^2
> mod2=lm(sqrt(kwh)~htdd+cldd+t1+t1.2,data=khct.df)
```

Passons à l'examen des résidus de cet ajustement. Le chronogramme des résidus (fig. 3.3 bas) ne montre plus une forme parabolique, mais on voit que les résidus sont de même signe par paquet. Comme il y a une constante dans la régression, ils sont de moyenne (empirique) nulle. Ce regroupement en blocs de même signe indique qu'ils sont autocorrélés positivement.

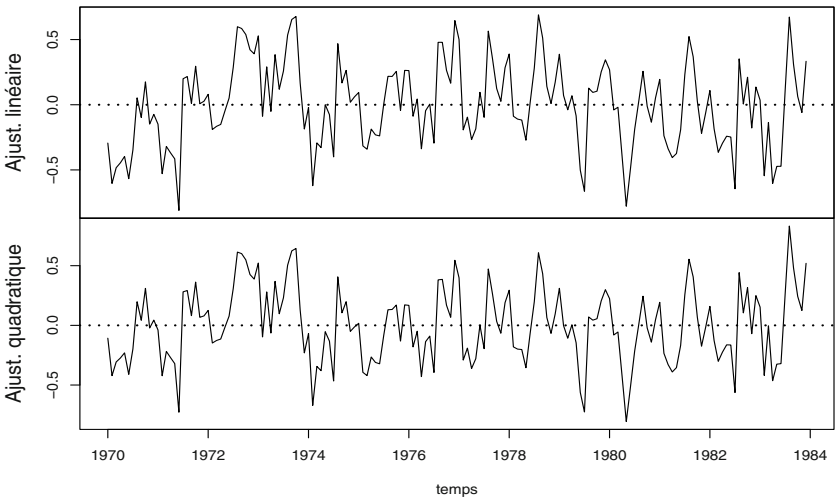

Fig. 3.3 – Résidus du modèle 1 (en haut) et du modèle 2 (en bas).

Pour examiner cette structure d'autocorrélation, il est naturel d'utiliser un lag plot. On l'obtient, jusqu'au retard 12, pour le modèle avec terme quadratique par :

```
> lag.plot(rev(residuals(mod2)),12,layout=c(4,3),diag.col="red",cex.main=.7)
```

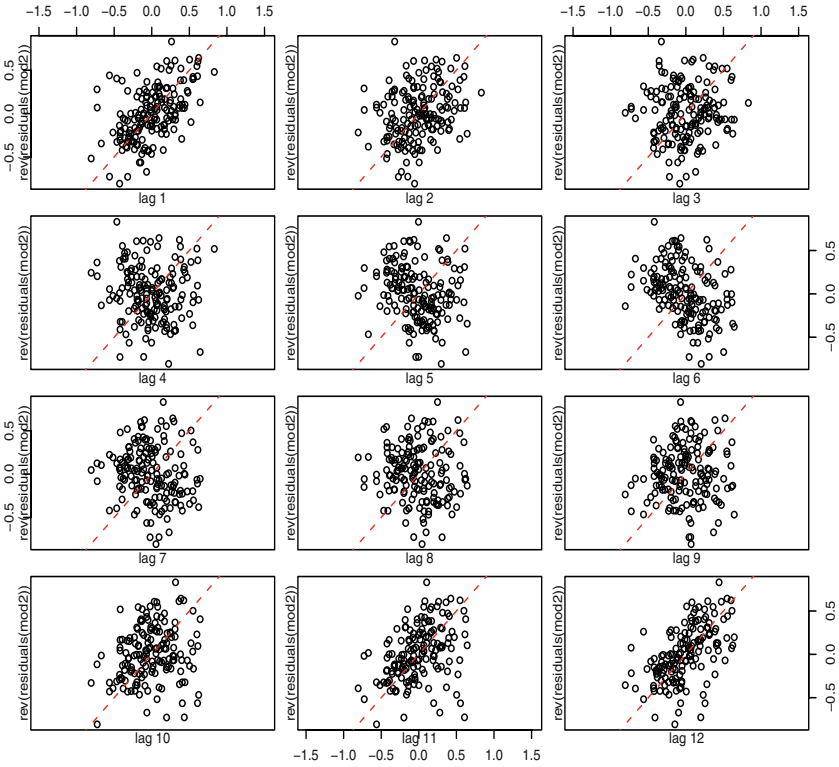

Fig. 3.4 – Lag plot des résidus du modèle 2.

On observe sur ce lag plot une forte corrélation aux retards 1 et 12 (fig. 3.4) ; le présupposé P3 ne tient donc pas. Nous tiendrons compte de cette corrélation au chapitre 10. Dans l'immédiat, nous poursuivons le commentaire de la régression comme si les quatre présupposés étaient vérifiés.

Examen de la normalité des résidus du modèle 2. L'aide de `plot.lm()` nous indique que six graphiques peuvent être dessinés en sortie d'une régression linéaire et que le Q-Q plot de normalité est le deuxième de ces graphiques. D'où le code qui donne le QQ-plot des résidus de la régression (fig. 3.5) et montre des points raisonnablement alignés.

```
> plot(mod2,which=2)
```

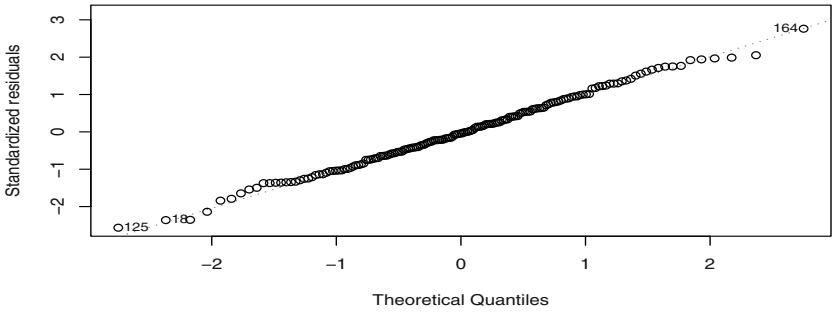

Fig. 3.5 – QQ-plot des résidus du modèle 2.

Le test de d'Agostino (cf. vignette `Anx3`) donne des p-values élevées (tableau 3.2).

```
> require(fBasics)
> a.dag=dagoTest(residuals(mod2))
> aa2=cbind(a.dag@test$statistic,a.dag@test$p.value)
> colnames(aa2)=c( "Stat.","p-value")
> rownames(aa2)=c("Omnibus D'Agos.","Skewness D'Agos.","Kurtosis D'Agos.")
```

Tableau 3.2 – Tests de normalité des résidus.

	Stat.	p-value
Omnibus D'Agos.	0.6927	0.7073
Skewness D'Agos.	0.3170	0.7512
Kurtosis D'Agos.	-0.7696	0.4416

Nous n'avons donc aucune raison de rejeter l'hypothèse de normalité. Nous prendrons en compte l'autocorrélation des résidus au chapitre 10. Dans l'immédiat, nous ignorons la non-optimalité des MCO qui découle de l'autocorrélation des erreurs, et poursuivons l'examen des résultats du modèle 2 comme si tous les présupposés étaient vérifiés. Pour une présentation des graphiques que propose `lm()`, le lecteur aura intérêt à consulter Cornillon & Matzner-Løber (2010), chapitre 4.

Examen de l'estimation du modèle 2. Nous avons examiné les résidus et n'y avons rien décelé de pathologique à l'autocorrélation près, qui sera prise en compte au chapitre 10. Etudions la sortie de la régression linéaire par `summary(mod2)`.

```
> summary(mod2)

Call:
lm(formula = sqrt(kwh) ~ htdd + cldd + t1 + t1.2, data = khct.df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.80630 -0.21680 -0.01287  0.20944  0.83219
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.731e+02  1.197e+01  -56.25  < 2e-16 ***
htdd         6.552e-04  5.029e-05   13.03  < 2e-16 ***
cldd         9.995e-03  4.752e-04   21.03  < 2e-16 ***
t1           3.456e-01  6.052e-03   57.10  < 2e-16 ***
t1.2        -5.943e-03  1.674e-03   -3.55 0.000503 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3168 on 163 degrees of freedom
Multiple R-squared: 0.9585,        Adjusted R-squared: 0.9575
F-statistic: 940.5 on 4 and 163 DF,  p-value: < 2.2e-16
```

On trouve d'abord le minimum, les quartiles et le maximum des résidus ; on peut voir ainsi s'ils sont symétriquement distribués, s'il n'y a pas un point atypique puis un tableau comportant une ligne par paramètre estimé et différentes colonnes :
- colonne `Estimate`, les estimations MCO des coefficients, $\widehat{\beta}_i$;
- colonne `Std. Error`, les écarts types d'estimation, $s(\beta_i)$;
- colonne `t value`, la statistique T : `t value = Estimate/Std. Error` qui est bien une t-statistique, voir la discussion après l'expression (3.11) ;
- colonne `Pr(>|t|)`, la p-value quand l'hypothèse alternative est $H_0 : \beta_i \neq 0$.

On trouve ensuite l'estimation de l'écart type des résidus, s_u, les coefficients de détermination et la statistique de Fisher pour tester la significativité globale de la régression.
Le modèle ajusté est :

$$\begin{aligned}\sqrt{\texttt{kwh}_t} \quad &= \quad -673.06 + 0.00066\ \texttt{htdd}_t + 0.01\ \texttt{cldd}_t \\ &\qquad +0.3456\ \texttt{t1}_t - 0.0059\ (\texttt{t1} - 1977)_t^2 + u_t \end{aligned} \tag{3.19}$$

avec $\widehat{\text{var}}(u_t) = 0.3168^2$, obtenue d'après `Residual standard error: 0.3168`.

Significativité de la régression

Significativité globale. La statistique de Fisher (3.10) prend la valeur $F = 940.5$, ligne `F-statistic`. Sous H_0 (la régression n'est pas significative), F est une observation d'une loi $\mathcal{F}(4, 163)$. La p-value (`< 2.2e-16`) est donnée sur la même ligne. Il n'y a pratiquement aucune chance qu'une v.a. suivant une loi $\mathcal{F}(4, 163)$ dépasse la valeur 940.5. La régression est très significative (on s'en doutait).

Significativité de chaque variable. Elle est donnée par la colonne `Pr(>|t|)`. Un examen rapide montre que toutes les variables sont très significatives.

Remarques
- Si l'on veut tester que le coefficient du temps est 0.3 contre l'alternative qu'il est plus grand que 0.3, on calcule la statistique `(0.346-0.3)/0.00605=7.6033` ; on rejette l'hypothèse pour les grandes valeurs de la statistique de test. La p-value est approximativement $P[Z > 7.6033 | Z \sim \mathcal{N}(0, 1)]$, elle est pratiquement nulle ;

on rejette donc l'hypothèse que le coefficient du temps est 0.3 au profit de « le coefficient du temps est supérieur à 0.3 ».
– La matrice des covariances des estimateurs des paramètres de la régression s'obtient par `vcov()`. Si l'on veut disposer de ces quantités pour des calculs futurs, il suffit de les stocker :

```
> resum2b=summary(mod2); vcov2b=vcov(mod2)
```

– En général, on ne doit conserver dans le modèle que les variables explicatives dont le coefficient est significativement différent de 0 à un niveau de confiance qu'on se fixe préalablement. Cependant, si la non-significativité d'une variable a un sens en soi, il arrive qu'on la conserve.
– Si les variables explicatives ne sont pas orthogonales, la suppression d'une variable modifie la significativité des autres variables. On ne peut donc pas dans ce cas supprimer simultanément plusieurs explicatives à l'aide du seul examen des t-statistiques ci-dessus.

Prévision de la consommation en 1984

On connaît les valeurs des explicatives `htdd`, `cldd`, `t1` et `t1.2` pour chaque mois de 1984 et nous voulons prédire la consommation de ces mêmes mois. La démarche a été rappelée à la section 3.5.
Nous fondant sur le modèle ajusté, nous estimons l'espérance de $\sqrt{\texttt{kwh}}$ par mois en 1984, `p0` dans le code ci-dessous, calculons des intervalles (à 80%) de confiance, `p1`, et de prédiction, `p2`, pour la variable dépendante $\sqrt{\texttt{kwh}}$ pour chaque mois prédit et dessinons les bandes de confiance correspondantes.

```
> khct.df.84=as.data.frame(window(cbind(khct,time(khct),
+ (time(khct)-1977)^2),start=c(1984,1)))
> colnames(khct.df.84)=c("kwh","htdd","cldd","t1","t1.2")
> p0=predict(mod2,khct.df.84,se.fit=TRUE)
> p1=predict(mod2,khct.df.84,interval="confidence",level=0.80,se.fit=TRUE)
> p2=predict(mod2,khct.df.84,interval="prediction",level=0.80,se.fit=TRUE)
> cbind(p1$fit[1:6,2:3], p2$fit[1:6,2:3])
```

```
       lwr      upr      lwr      upr
1 13.28732 13.50435 12.97397 13.81770
2 12.97339 13.17528 12.65435 13.49431
3 13.09014 13.29987 12.77407 13.61595
4 12.62541 12.84084 12.31147 13.15479
5 12.48422 12.71331 12.17531 13.02222
6 12.91328 13.14176 12.60415 13.45090
```

On commence par isoler l'année 1984 à l'aide de `window()`, on fabrique également la matrice des variables explicatives qui serviront à la prédiction de y. Par ailleurs, `?predict.lm` nous apprend que `predict()` sans précision d'un intervalle fournit la prédiction ponctuelle, c'est-à-dire l'expression (3.14). L'option `se.fit=TRUE` fournit également l'écart type des moyennes prédites, voir l'expression (3.15), mais pas les écarts types des erreurs de prédiction intervenant dans

(3.16). On peut les calculer directement ou les récupérer du calcul de la bande de prédiction à 80% par :

```
> (etyp.pmco=(p2$fit[,3]-p0$fit)/qnorm(.9))
```

```
        1         2         3         4         5         6
0.3291839 0.3277138 0.3284623 0.3290235 0.3304261 0.3303607
        7         8         9        10        11        12
0.3305726 0.3332624 0.3320962 0.3325902 0.3320606 0.3335524
```

opération qui nous permet d'isoler l'écart type dans (3.16).
Nous pouvons maintenant représenter simultanément les points observés en 1984 et les bandes de prédiction et de confiance. Sur la figure 3.6 on a superposé : en trait plein, la réalisation de $\sqrt{\texttt{kwh}}$ en 1984, en pointillé, la bande de confiance pour la moyenne, en tiret la bande de prédiction pour la série.

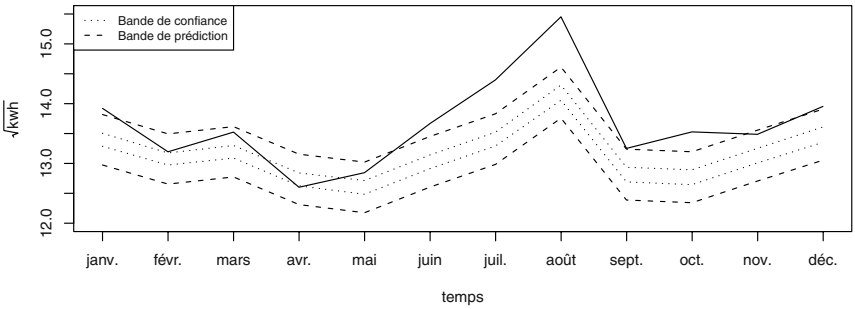

Fig. 3.6 – Consommation en 1984, bandes de prédiction et de confiance par MCO.

D'abord, on repère les minimum et maximum des séries à représenter en vue de la mise à l'échelle ; pour cela on utilise le vecteur de toutes les ordonnées `poub` et on en prend le minimum et le maximum. Nous identifions les mois par leurs abréviations d'après `months()` (SiteST). On définit le graphique, d'abord sans tracer l'axe horizontal, `xaxt="n"`, ensuite, à l'aide de `axis()`, on construit cet axe : on choisit l'emplacement des graduations et leurs étiquettes. On superpose ensuite les intervalles de confiance et de prédiction par `points()`, où l'on indique qu'on veut lier les points (`type=l`).

```
> kwh2rc=window(sqrt(khct["kwh"]),start=c(1984,1))
> temps2=time(kwh2rc); aa=seq(as.Date("2000/1/1"),by="month",length.out=12)
> id.mois=months(aa,abbreviate=TRUE)
> par(oma=rep(0,4),mar=c(4,4,3,2))
> plot(kwh2rc,ylim=c(12,15.5),xlab='temps',
+              ylab=expression(sqrt(kwh)),xaxt="n")
> axis(1,at=seq(1984,1984.917,length.out=12),labels=id.mois)
> points(p1$fit[,2]~temps2,type='l',lty="dotted") #intervalle de confiance
> points(p1$fit[,3]~temps2,type='l',lty="dotted")
> points(p2$fit[,2]~temps2,type='l',lty="dashed") #intervalle de prédiction
> points(p2$fit[,3]~temps2,type='l',lty="dashed")
```

```
> legend("topleft",c("Bande de confiance","Bande de prédiction"),
+         lwd=1,lty=c("dotted","dashed"),cex=.8)
```

On observe que 7 points sur les 12 sortent de l'intervalle de prédiction à 80%, soit une proportion de 58%, très supérieure à la proportion théorique de 20/100, même si le faible nombre d'observations, 12, ne permet pas de conclure. On peut espérer, en tenant compte de la dynamique de l'erreur observée sur les résidus des MCO, améliorer la prédiction. Le calcul de cette proportion, souvent plus pratique que le comptage sur un graphique, peut s'effectuer par le code :

```
> sum((kwh2rc<p2$fit[,2])|(kwh2rc>p2$fit[,3]))/length(kwh2rc)

[1] 0.5833333
```

Nous n'avons pas retransformé la série en `kwh`. A cause de la non-linéarité de la transformation $\sqrt{.}$, il est difficile de prédire correctement la distribution des prédictions de `kwh`. Nous pouvons cependant contourner la difficulté par simulation.

Prévision de la consommation par simulation. Le modèle ajusté (3.19) donne une estimation de la moyenne de $\sqrt{\texttt{kwh}}$ ainsi qu'une estimation de la loi de l'erreur. Et nous disposons de la prédiction ponctuelle de la moyenne de $\sqrt{\texttt{kwh}}$ pour chaque mois de 1984 par `p0 = predict(mod2, an84)`. Pour un mois donné de 1984, c'est-à-dire pour une ligne de `an84`, nous pouvons simuler des valeurs de l'erreur, additionner chaque valeur à la prédiction de la moyenne de $\sqrt{\texttt{kwh}}$ pour obtenir une prédiction de $\sqrt{\texttt{kwh}}$ et élever au carré pour obtenir une prédiction de `kwh`. Effectuant un grand nombre de simulations pour chaque mois, nous obtenons la distribution des prédictions. C'est ainsi que nous procéderons au chapitre 10 où, après avoir modélisé la dynamique du bruit, nous effectuons de telles simulations intégrant cette dynamique (section 10.5).
On ne peut se contenter des estimations des paramètres et de leurs significativités, il faut, nous le constatons, examiner attentivement les résidus. Leur comportement reflète celui des erreurs. Leur analyse est indispensable pour vérifier la qualité de la modélisation. On étudie ces résidus, d'abord par leur chronogramme, puis par tout autre moyen nécessaire pour saisir leur comportement : ACF (cf. 4.3), diagramme de dispersion des résidus contre chaque variable explicative...
Notre traitement de la série `kwh` n'est pas totalement satisfaisant. Certes, le R2 ajusté vaut 0.9575, ce qui est élevé, mais la qualité prédictive du modèle est faible, alors que l'horizon, 12 mois, n'est pas très lointain. Un remède réside sans doute dans la prise en compte de l'autocorrélation des erreurs manifestée par l'autocorrélation des résidus. Si l'on arrive à modéliser le mécanisme de l'erreur d'après la dynamique des résidus, on pourra faire une estimation par MCG du modèle. Les estimations des coefficients changeront légèrement, leur significativité également, mais surtout on pourra faire la prédiction de la série en additionnant la prédiction de sa moyenne et la prédiction de l'erreur. Pour arriver à faire cette modélisation, il nous faut d'abord réviser des notions sur les séries temporelles : stationnarité, modèle ARMA. Nous reprendrons la modélisation de cette série au chapitre 10.

Chapitre 4

Modèles de base en séries temporelles

4.1 Stationnarité

Une série temporelle $\{y_t\}$, ou processus stochastique, est dite *strictement stationnaire* si la distribution conjointe de $(y_{t_1}, \ldots, y_{t_k})$ est identique à celle de $(y_{t_1+t}, \ldots, y_{t_k+t})$, quels que soient k le nombre d'instants considérés, $(t_1, \ldots, t_k)$ les instants choisis et t, le décalage ; c'est-à-dire que, quels que soient le nombre de dates et les dates choisis, quand on décale ces dates d'une même quantité, la distribution ne change pas. En somme, la stationnarité stricte dit que la distribution conjointe de tout sous-vecteur de $\{y_t\}$, quels que soient sa longueur et les instants choisis, est invariante quand on translate ces instants d'une même quantité. Cette condition est difficile à vérifier et on utilise une version plus faible de stationnarité, la stationnarité faible ou du second ordre, souvent suffisante.

Définition 4.1
$\{y_t\}$ *est dite faiblement stationnaire si :*
– $\mathsf{E}(y_t) = \mu$, *constante indépendante de* t ;
– $\mathsf{cov}(y_t, y_{t-l})$ *ne dépend que de* l *entier et dans ce cas elle est notée :*

$$\gamma_l = \mathsf{cov}(y_t, y_{t-l}).$$

Ainsi, une série temporelle $\{y_t\}$ est *faiblement stationnaire* si sa moyenne ne dépend pas de t et si la covariance entre y_t et y_{t-l} ne dépend que de l et non de t.
On a distingué stationnarité stricte et stationnarité faible, et la plupart des modèles que nous allons examiner concernent des séries normalement distribuées ; pour elles, les deux notions coïncident. La suite du chapitre concerne les séries faiblement stationnaires. Des séries faiblement (et non fortement) stationnaires se

rencontrent dans les modèles à hétéroscédasticité conditionnelle ; l'étude de tels modèles constituerait un prolongement du chapitre 12.

Considérations pratiques pour apprécier la stationnarité d'une série. On dispose d'une trajectoire d'une série temporelle $\{y_t\}$ et on veut se faire une première idée de la stationnarité de cette série par l'observation du chronogramme de la trajectoire. Une condition nécessaire de stationnarité est que la moyenne et la variance de la série soient constantes. Elle implique donc que le graphe de la série en fonction du temps montre un niveau moyen à peu près constant et des fluctuations à peu près de même ampleur autour de la moyenne supposée, quelle que soit la date autour de laquelle on examine la série. Examinons quelques séries pour nous faire une opinion sur leur stationnarité éventuelle.

Exemples (Outils graphiques pour la stationnarité)

1. Une série stationnaire a une moyenne constante. Considérons le cours de l'action Danone (fig. 1.4). Imaginons un intervalle de 200 points environ et faisons glisser cet intervalle. Il est manifeste que pour cette série la moyenne dépend de t : elle n'est donc pas stationnaire. En résumé, si le niveau d'une série fluctue peu autour d'un niveau moyen sur un petit intervalle de temps mais que ce niveau moyen change avec la position de l'intervalle, on peut conclure que la série n'est pas stationnaire.
2. Une série stationnaire a une variance constante. Ce qui veut dire que l'ampleur de la fluctuation de la série reste la même sur un petit intervalle de temps, quel que soit cet intervalle. Le nombre de morts sur les routes en France (fig. 1.2) décroît avec le temps et montre une variabilité, donc une variance, qui diminue. Cette série n'est donc pas stationnaire.
3. La série des températures de l'air à Nottingham Castle présente plusieurs traits dont chacun montre qu'elle n'est pas stationnaire. Le chronogramme de la série est très régulier. Chaleur en été, froid en hiver... La moyenne sur 6 mois consécutifs est très différente selon que ces mois sont centrés sur l'été ou sur l'hiver. Le lag plot (fig. 1.9), qui montre une forme en anneau très marquée pour certains retards et des points alignés au retard 12, suggère une dépendance fonctionnelle de la température par rapport au passé et donc la non-stationnarité de la série. Enfin le month plot (fig. 1.11), où l'on observe que la température d'un mois, janvier par exemple, fluctue peu au cours des années autour d'un niveau stable, suggère que cette série est une fonction périodique entachée d'une erreur qui, elle, pourrait être stationnaire ; c'est un cas semblable à celui du lac Huron. Donc la série n'est vraisemblablement pas stationnaire ; régularité ne veut pas dire stationnarité. Cette série est étudiée au chapitre 9.

4.1.1 Fonction d'autocorrélation d'une série stationnaire

Soit $\{y_t\}$ une série à valeurs réelles, stationnaire. La covariance $\gamma_l = \mathsf{cov}(y_t, y_{t-l})$ est appelée autocovariance d'ordre (ou de décalage) l (lag-l autocovariance).

Définition 4.2 (Fonction d'autocovariance)
La fonction :

$$l \to \gamma_l,\ l = \ldots, -1, 0, 1, 2, \ldots$$

est la fonction d'autocovariance de $\{y_t\}$.

Cette fonction vérifie notamment :

Proposition 4.1
- $\gamma_0 = \mathsf{var}(y_t) \geq 0$;
- $|\gamma_l| \leq \gamma_0\ \forall l$;
- $\gamma_l = \gamma_{-l}\ \forall l$.

Cette fonction étant paire, on ne la représente que pour $l = 0, 1, 2, \ldots$. On a également :

Proposition 4.2
La fonction d'autocovariance d'une série $\{y_t\}$ *faiblement stationnaire est de type positif*[1].

Cette propriété exprime le fait que la variance d'une combinaison linéaire de n v.a. $y_{t_1}, \ldots, y_{t_n}$ est positive.

Fonction d'autocorrélation théorique.

Définition 4.3 (coefficient d'autocorrélation)
Le coefficient d'autocorrélation d'ordre l *est :*

$$\rho_l = \frac{\mathsf{cov}(y_t, y_{t-l})}{\sqrt{\mathsf{var}(y_t)\mathsf{var}(y_{t-l})}} = \frac{\mathsf{cov}(y_t, y_{t-l})}{\mathsf{var}(y_t)} = \frac{\gamma_l}{\gamma_0}. \tag{4.1}$$

La dernière égalité tient car $\mathsf{var}(y_{t-l}) = \mathsf{var}(y_t) = \gamma_0$. Enfin, en notant que par la stationnarité $\mathsf{E}(y_t) = \mu$, indépendant de t, on a en terme d'espérance mathématique :

$$\rho_l = \frac{\mathsf{E}[(y_t - \mu)(y_{t-l} - \mu)]}{\mathsf{E}[(y_t - \mu)^2]}. \tag{4.2}$$

Définition 4.4 (Fonction d'autocorrélation)
La fonction :

$$l \to \rho_l,\ l = 0, 1, 2, \ldots$$

est la fonction d'autocorrélation (théorique) de la série $\{y_t\}$.

1. Une fonction à valeurs réelles κ définie sur les entiers est de type positif si pour tout entier n et tout vecteur $a = (a_1, \ldots, a_n)$,

$$\sum_{i,j=1}^{n} a_i \kappa(i-j) a_j \geq 0$$

.

Nous utiliserons l'abréviation anglaise, ACF, qui est aussi celle des sorties de R, de préférence à FAC. On appelle son graphique *corrélogramme*. On voit que : $\rho_0 = 1, -1 \leq \rho_l \leq 1$.

Fonction d'autocorrélation empirique. Etant donné une série observée $y_t, t = 1, \ldots, T$, notons $\overline{y} = \sum_{t=1}^{T} y_t/T$. L'autocovariance empirique d'ordre l est

$$\widehat{\gamma}_l = \frac{\sum_{t=l+1}^{T}(y_t - \overline{y})(y_{t-l} - \overline{y})}{T},\ 0 \leq l \leq T-1. \tag{4.3}$$

Le coefficient d'autocorrélation empirique d'ordre l est

$$\widehat{\rho}_l = \frac{\sum_{t=l+1}^{T}(y_t - \overline{y})(y_{t-l} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2},\ 0 \leq l \leq T-1. \tag{4.4}$$

Observons que le dénominateur dans (4.3) est T alors que le nombre de termes au numérateur dépend du décalage. Il faut se garder de corriger l'estimation en adoptant un dénominateur dépendant du nombre de termes dans la somme. En effet, avec un tel choix, la fonction d'autocovariance empirique $l \to \widehat{\gamma}_l$ ne serait plus de type positif.
La fonction :

$$l \to \widehat{\rho}_l, l = 0, 1, 2, \ldots$$

est la *fonction d'autocorrélation empirique*.
On l'abrégera en ACF empirique ; son graphique est le corrélogramme empirique. Supposons maintenant que $y_t,\ t = 1, \ldots, T$ soit une trajectoire de $\{y_t\}$ série stationnaire infinie. Alors, sous des conditions générales, voir par exemple Brockwell & Davis (2002), $\widehat{\rho}_l$ est un estimateur convergent de ρ_l.

4.1.2 Bruit blanc

Définition 4.5 (Bruit blanc)
Un bruit blanc $\{z_t\}$ est une suite de v.a. non corrélées (mais pas nécessairement indépendantes) de moyenne nulle et de variance constante σ_z^2.

C'est donc une série faiblement stationnaire. On note $z_t \sim \mathrm{BB}(0, \sigma_z^2)$.

Définition 4.6 (Bruit blanc gaussien)
Un bruit blanc gaussien $\{z_t\}$ est une suite de v.a. i.i.d. $\mathcal{N}(0, \sigma_z^2)$, on note : $z_t \sim \mathrm{BBN}(0, \sigma_z^2)$.

Un bruit blanc gaussien est une série strictement stationnaire.
Examinons ce que deviennent les coefficients d'autocorrélations empiriques quand ils sont calculés sur une série dont tous les coefficients d'autocorrélations théoriques sont nuls.

Propriété 4.1
Si $y_t,\ t = 1, \ldots, T$ *est une observation d'une suite de v.a. i.i.d. de moment d'ordre 2 fini,* $\mathsf{E}(y_t^2) < \infty$*, alors les* $\widehat{\rho}_l$ *sont approximativement indépendants et normalement distribués de moyenne* 0 *et de variance* $1/T$*.*

En s'appuyant sur cette propriété, on peut tracer des intervalles autour de 0, qui doivent contenir les $\widehat{\rho}_l$ si la série est effectivement une suite de v.a. i.i.d. Nous illustrerons cette technique après la propriété 4.3 (p. 71) plus générale.
Il n'est pas facile de vérifier qu'une série est formée de v.a. i.i.d. On peut par contre tester la nullité de coefficients d'autocorrélation, ce que fait le test du portemanteau.

Test de blancheur : le test du portemanteau. Soit la série observée $y_t,\ t = 1, \ldots, T$, considérons la statistique

$$Q(h) = T \sum_{j=1}^{h} \widehat{\rho}_j^2, \tag{4.5}$$

où h est un décalage choisi par l'utilisateur et $\widehat{\rho}_j$ l'estimateur (4.4) du coefficient d'autocorrélation d'ordre j de la série y_t. $Q(h)$ est appelée *statistique de Box-Pierce*. Elle permet de tester :

$$H_0^h : \rho_1 = \rho_2 = \ldots = \rho_h = 0$$

contre

$$H_1^h : \quad \text{au moins un des } \ \rho_1, \rho_2, \ldots, \rho_h \text{ est non nul.}$$

$Q(h)$ est la distance du χ^2 du vecteur $(\rho_1, \rho_2, \ldots, \rho_h)$ au vecteur $(0, 0, \ldots, 0)$ et on rejette l'hypothèse H_0^h pour les grandes valeurs de $Q(h)$.
En effet, sous l'hypothèse que $\{y_t\}$ est une suite de v.a. i.i.d. et vu la propriété (4.1), $Q(h)$ n'est autre que

$$Q(h) = T \sum_{j=1}^{h} \widehat{\rho}_j^2 = \sum_{j=1}^{h} (\frac{\widehat{\rho}_j - 0}{1/\sqrt{T}})^2,$$

c'est-à-dire la somme des carrés de h variables approximativement $\mathcal{N}(0, 1)$. Or, sachant que le carré d'une variable $\mathcal{N}(0, 1)$ suit une loi χ_1^2 et que la somme de deux v.a. indépendantes et distribuées suivant des lois $\chi_{n_1}^2$ et $\chi_{n_2}^2$ suit une loi $\chi_{n_1+n_2}^2$, la loi de $Q(h)$ est bien approximativement χ_h^2, sous l'hypothèse nulle. Notons qu'on doit choisir h, le nombre de coefficients dont on teste la nullité. Dans l'exemple ci-dessous on choisit successivement $h = 3, 6, 9, 12$.

Remarques (Variantes du test de blancheur)
– Pour des petits échantillons on utilise la *statistique de Ljung-Box* :

$$Q^*(h) = T(T+2) \sum_{k=1}^{h} \frac{\widehat{\rho}_k^2}{T-k}. \tag{4.6}$$

Elle a une distribution de probabilité mieux approchée par un χ^2 que la statistique de Box-Pierce.
– Quand le test est appliqué non sur des v.a. indépendantes, mais sur les résidus d'un ajustement estimant m paramètres, la loi approchée sous l'hypothèse nulle est un χ^2 à $h - m$ degrés de liberté.

Exemple 4.1 La simulation d'une série, qui permet notamment de fabriquer des trajectoires à partir d'un modèle de série, est étudiée en détail au chapitre 7. Cependant, simulons 100 observations de y_t vérifiant :

$$y_t = -0.7y_{t-1} + z_t, \text{ série } \texttt{y1} \tag{4.7a}$$

$$y_t = -0.7y_{t-12} + z_t, \text{ série } \texttt{y2} \tag{4.7b}$$

où z_t est un bruit blanc gaussien de variance 4 et testons la blancheur de chaque série en calculant la statistique de Ljung-Box (4.6). Nous utilisons la fonction `Box.test.2()` de **caschrono**. Il faut préciser les retards auxquels on veut calculer la statistique. Mais d'abord, nous fixons la *graine* (cf. chap. 7), un entier, par la fonction `set.seed()` :

```
> require(caschrono)
> set.seed(123)
> y1=arima.sim(n=100,list(ar=-.7),sd=sqrt(4))
> y2=arima.sim(n=100,list(ar=c(rep(0,11),-.7)),sd=sqrt(4))
> ret=c(3,6,9,12)
> a1=Box.test.2(y1,nlag=ret,type="Ljung-Box",decim=2)
> a2=Box.test.2(y2,nlag=ret,type="Ljung-Box",decim=2)
> a12=cbind(a1,a2[,2])
> colnames(a12)=c("Retard","p-val. y1","p-val.y2")
> a12
     Retard p-val. y1 p-val. y2
[1,]      3         0      0.66
[2,]      6         0      0.49
[3,]      9         0      0.43
[4,]     12         0      0.00
```

L'emploi de `arima.sim()` de **stats** est expliqué en détail au chapitre 7. Ici il suffit de noter qu'on doit donner la longueur de la série à simuler, les coefficients d'autorégression et de moyenne mobile si l'une ou l'autre de ces composantes est présente, et l'écart type du bruit blanc. Après simulation, on dispose de `y1` suivant (4.7a) et `y2` suivant (4.7b) [2].
On voit qu'on rejette la blancheur de la série `y1` quel que soit le retard où l'on calcule la statistique du portemanteau, alors que pour `y2`, si on arrête à un ordre inférieur à 12, on croit que la série est un bruit blanc. Des éléments (numériques) d'explication seront donnés à la section 4.2, où l'on calcule la représentation de ces deux séries en fonction du bruit blanc passé et présent.

2. Il n'est pas possible de vérifier étroitement qu'on a bien simulé suivant ces modèles, mais on peut estimer les modèles sur les séries simulées et voir si l'estimation ressemble au modèle. Ces estimations sont abordées à la section 4.5.

Remarques

- Plus le décalage l est grand, moins il y a d'observations pour estimer ρ_l dans (4.4). On s'arrête habituellement à $l \simeq T/4$.
- Le test du portemanteau vérifie que les h premiers coefficients d'autocorrélation sont nuls, mais ne dit rien sur les coefficients d'ordre supérieur à h. Si le phénomène examiné est susceptible de montrer une saisonnalité de période s, cas de `y2` ci-dessus, on doit choisir $h > s$.
- De même qu'un portemanteau rassemble plusieurs vêtements, le test du portemanteau traite simultanément plusieurs coefficients d'autocorrélation.
- Il est recommandé de faire, parallèlement au test de la blancheur d'une série, une inspection visuelle de son ACF.
- Observons que l'on peut calculer (4.4) *pour toute série, stationnaire ou non*. Si la série n'est pas stationnaire, (4.4) a un usage purement empirique. On montre que pour une série stationnaire, le *corrélogramme empirique*, graphe de $l \to \widehat{\rho}(l)$ décroît exponentiellement vers 0, avec éventuellement des oscillations. Inversement, un graphe du corrélogramme empirique qui ne montre pas de décroissance rapide est l'indice d'une non-stationnarité. Ainsi l'examen du corrélogramme empirique d'une série permet de se faire une idée de sa stationnarité. Il complète celui du chronogramme.

Exercice 4.1 (Danone - test de blancheur)
Test de la blancheur du rendement de l'action Danone et de celle de son carré. On suivra les étapes suivantes :

1. Calculer le carré du rendement centré.
2. Tester la blancheur du rendement sur toute la série (on pourra tester la blancheur aux retards 3, 6, 9 et 12). On obtient un résultat inattendu. Lequel ?
3. Après examen du chronogramme du rendement (fig. 1.4, chap. 1) on décide de se limiter à l'étude de la série des 600 premières valeurs. Pourquoi ce choix ? Qu'a-t-on observé sur le chronogramme ?
4. Tester la blancheur du rendement et du rendement centré au carré sur la série de ces 600 valeurs. Conclusion ?
5. Au vu de ces résultats, le rendement peut-il être un bruit blanc gaussien ?

Test de Durbin-Watson.

Le test de Durbin-Watson est un test d'absence d'autocorrélation d'ordre 1 sur le résidu d'une régression linéaire. Il s'intéresse à la situation

$$y_t = \mathbf{x}_t'\beta + u_t,\ t = 1, \ldots, T \tag{4.8}$$

$$u_t = \rho u_{t-1} + z_t \tag{4.9}$$

où $\mathbf{x}_t$ est un vecteur de $p+1$ variables explicatives (dont la constante), $z_t \sim$ BB. Il teste $H_0 : \rho = 0$. La statistique de Durbin-Watson est

$$DW = \frac{\sum_{t=2}^{T}(\widehat{u}_t - \widehat{u}_{t-1})^2}{\sum_{t=1}^{T}\widehat{u}_t^2},$$

où $\widehat{u}_t$ est le résidu de l'ajustement par moindres carrés ordinaires de y_t sur x_t. En développant numérateur et dénominateur, on voit que

$$DW \simeq 2(1-\widehat{\rho}) \in (0,4),$$

où $\widehat{\rho} = \sum_{t=2}^{T} \widehat{u}_{t-1}\widehat{u}_t / \sum_{t=1}^{T} \widehat{u}_t^2$. Les valeurs de DW proches de 0 indiquent une autocorrélation proche de 1.
Pour le test de $H_0 : \rho = 0$ contre $H_1 : \rho > 0$, la région critique correspond à de faibles valeurs de DW (DW sensiblement inférieur à 2) et pour $H_1 : \rho < 0$, la région critique correspond à de fortes valeurs de DW.

Remarques
- Pratiquement une statistique DW $\ll 2$ peut être le signe d'une mauvaise spécification du modèle (par exemple, ajustement d'une tendance linéaire alors que la tendance réelle est quadratique).
- Le test de Durbin-Watson est effectué par la plupart des logiciels à la suite d'une régression linéaire, que les observations soient ou ne soient pas une série temporelle. Il n'a pas de sens quand les données ne sont pas indicées par le temps.
- Le modèle (4.8) est de même nature que celui retenu pour le lac Huron au chapitre précédent.
- Le test est programmé dans `dwtest()` de **lmtest** et `durbin.watson()` de **car** où la p-value est calculée par bootstrap.

Le bruit blanc est une série de référence. Modéliser une série revient souvent à trouver les opérations qui la décrivent comme une transformation d'un bruit blanc. Dans la section suivante, nous examinons des modèles de séries bâtis à partir d'un bruit blanc.

4.2 Série linéaire

Définition 4.7
Une série $\{y_t\}$ est dite linéaire si elle peut s'écrire :

$$y_t = \mu + \sum_{i=-\infty}^{+\infty} \psi_i z_{t-i}, \tag{4.10}$$

où $z_t \sim \mathrm{BB}(0,\sigma_z^2)$, $\psi_0 = 1$ et la suite $\{\psi_i\}$ est absolument sommable, c'est-à-dire $\sum_i |\psi_i| < \infty$.
Une série $\{y_t\}$ est dite linéaire et causale si elle est linéaire avec $\psi_i = 0$, $i < 0$:

$$y_t = \mu + \sum_{i=0}^{\infty} \psi_i z_{t-i}. \tag{4.11}$$

On admettra qu'une série linéaire est stationnaire. L'étude des séries non causales conduit à des résultats non intuitifs difficilement utilisables, aussi nous ne considérerons parmi les séries linéaires que des séries causales (4.11). L'écriture (4.11) de y_t comme somme de v.a. non corrélées permet d'obtenir facilement :

$$\mathsf{E}y_t = \mu, \quad \mathsf{var}(y_t) = \sigma_z^2(1+\sum_{i=1}^{\infty}\psi_i^2), \quad \gamma_l = \mathsf{cov}(y_t, y_{t-l}) = \sigma_z^2\sum_{j=0}^{\infty}\psi_j\psi_{j+l} \quad (4.12)$$

Modèles autorégressifs, moyennes mobiles

On exprime souvent l'évolution d'une série en fonction de son passé. Les modèles autorégressifs sont les modèles les plus explicites pour exprimer cette dépendance.

Introduction aux modèles autorégressifs. Considérons le modèle

$$y_t = c + \phi y_{t-1} + z_t, \qquad z_t \sim \mathrm{BB}(0, \sigma_z^2), \quad (4.13)$$

où c et ϕ sont des constantes, appelé modèle autorégressif d'ordre 1 et étudions-le. Par substitutions successives, on obtient

$$y_t = c(1+\phi+\ldots+\phi^{k-1}) + \phi^k y_{t-k} + \sum_{j=0}^{k-1}\phi^j z_{t-j}.$$

Si $|\phi| < 1$ on peut représenter y_t par

$$y_t = \frac{c}{1-\phi} + \sum_{j=0}^{\infty}\phi^j z_{t-j},$$

ainsi, dans ce cas, y_t est une série linéaire, donc stationnaire et *causale*. Observons que cette écriture s'obtient aussi directement à l'aide de l'opérateur retard. En effet, l'équation (4.13) s'écrit

$$(1-\phi\mathrm{B})y_t = c + z_t.$$

– Supposons que $|\phi| < 1$, nous avons évidemment que $\phi \neq 1$ et donc

$$y_t = \frac{c}{1-\phi\mathrm{B}} + \frac{1}{1-\phi\mathrm{B}}z_t.$$

Ensuite, comme $|\phi| < 1$ on peut effectuer le développement en série

$$\frac{1}{1-\phi\mathrm{B}} = 1 + \phi\mathrm{B} + \phi^2\mathrm{B}^2 + \ldots \quad (4.14)$$

et par ailleurs, $c/(1-\phi\mathrm{B}) = c/(1-\phi)$, donc

$$y_t = \frac{c}{1-\phi} + z_t + \phi z_{t-1} + \phi^2 z_{t-2} + \ldots \quad (4.15)$$

est stationnaire, de moyenne $\mathsf{E}(y_t) = \mu = c/(1-\phi)$. Un processus autorégressif d'ordre 1 stationnaire est noté AR(1).
- Si $\phi = 1$, l'autorégressif (4.13) n'est pas stationnaire. C'est une marche aléatoire, avec dérive si $c \neq 0$, considérée au chapitre 5.
- Si $|\phi| > 1$, l'autorégressif (4.13) est explosif.

Exemple 4.2 Les représentations MA(∞) des séries simulées (4.7) sont

$$\frac{1}{1+0.7\mathrm{B}} = 1 - 0.7\mathrm{B} + 0.49\mathrm{B} + (-0.7^3)\mathrm{B}^3 + \ldots$$

et

$$\frac{1}{1+0.7\mathrm{B}^{12}} = 1 - 0.7\mathrm{B}^{12} + 0.49\mathrm{B}^{24} + (-0.7^3)\mathrm{B}^{36} + \ldots$$

On peut également faire ce calcul dans R à l'aide de `ARMAtoMA()` (SiteST).

Considérons maintenant les processus autorégressifs d'ordre p. Un processus $\{y_t\}$ est dit autorégressif d'ordre p s'il vérifie :

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + z_t, \qquad z_t \sim \mathrm{BB}(0, \sigma_z^2),$$

avec $\phi_p \neq 0$.
Avec l'opérateur retard on peut écrire cette autorégression comme :

$$\Phi(\mathrm{B}) y_t \quad = \quad c + z_t$$

où

$$\Phi(\mathrm{B}) = 1 - \phi_1 \mathrm{B} - \phi_2 \mathrm{B}^2 - \ldots - \phi_p \mathrm{B}^p$$

est l'*opérateur d'autorégression.*

Définition 4.8 (Processus AR(p))
Un processus $\{y_t\}$ est un AR(p) s'il obéit à

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + z_t, \qquad z_t \sim \mathrm{BB}(0, \sigma_z^2)$$

avec $\phi_p \neq 0$ et il est stationnaire.

A quelle condition le processus autorégressif (4.16) est-il stationnaire ? Supposons que $p = 2$. Appelons s_1 et s_2 les racines, réelles ou complexes, de $1 - \phi_1 z - \phi_2 z^2 = 0$. On a donc $1 - \phi_1 z - \phi_2 z^2 = (1 - z/s_1)(1 - z/s_2)$ et on voit que le développement en série de $1 - \phi_1 z - \phi_2 z^2$ est possible si les racines de ce polynôme sont en module strictement supérieures à 1 ; dans ce cas y_t est stationnaire, de moyenne $\mu = c/(1 - \phi_1 - \phi_2)$, définie car 1 n'est pas racine du polynôme. Pour un ordre p quelconque, nous admettrons :

Proposition 4.3
Le processus autorégressif d'ordre p (4.16) admet une représentation MA(∞) si les racines de l'équation : $1-\phi_1 z-\phi_2 z^2-\ldots-\phi_p z^p=0$ *sont strictement supérieures à 1 en module, (4.16) est alors stationnaire ; c'est un AR(p).*

Dans ce cas

$$\mathsf{E}(y_t)=\mu=\frac{c}{1-\phi_1-\phi_2-\ldots-\phi_p}$$

et on peut encore écrire (4.16) comme

$$y_t=\mu+\frac{1}{1-\phi_1\mathrm{B}-\phi_2\mathrm{B}^2-\ldots-\phi_p\mathrm{B}^p}z_t, \qquad z_t\sim \mathrm{BB}(0,\sigma_z^2). \tag{4.16}$$

Cette formulation sépare clairement le niveau moyen μ de la série, de l'erreur qui obéit à une dynamique autorégressive stationnaire. Le niveau moyen peut lui-même être une fonction du temps, comme dans la modélisation du niveau du lac Huron (chap. 1) où l'erreur est AR(1). Dans ce cas, la série est la somme d'une tendance déterministe et d'une erreur stationnaire.

Introduction aux modèles moyennes mobiles.

Définition 4.9 (Processus MA(q))
$\{y_t\}$ *est un processus moyenne mobile d'ordre q (MA(q)) si :*

$$y_t \;=\; \mu+z_t+\theta_1 z_{t-1}+\theta_2 z_{t-2}+\ldots+\theta_q z_{t-q}, \qquad z_t\sim \mathrm{BB}(0,\sigma_z^2), \tag{4.17}$$

avec $\theta_q\neq 0$.

Introduisant l'opérateur moyenne mobile

$$\Theta(\mathrm{B})=1+\theta_1\mathrm{B}+\theta_2\mathrm{B}^2+\ldots+\theta_q\mathrm{B}^q,$$

on peut noter de façon équivalente :

$$y_t \;=\mu+\Theta(\mathrm{B})z_t.$$

Un MA(q) est toujours stationnaire quelles que soient les valeurs des θ ; il est de moyenne μ.
On aimerait pouvoir exprimer ce processus en fonction de son passé (observé) et pas seulement en fonction du bruit passé non observé. C'est la question de l'*inversibilité* du processus. Examinons le cas d'un MA(1) centré :

$$y_t=z_t+\theta z_{t-1}=(1+\theta\mathrm{B})z_t, \qquad z_t\sim \mathrm{BB}(0,\sigma_z^2). \tag{4.18}$$

On voit que si $|\theta|<1$, on peut développer $(1+\theta\mathrm{B})^{-1}$ en série : $(1+\theta\mathrm{B})^{-1}=1-\theta\mathrm{B}+\theta^2\mathrm{B}^2-\theta^3\mathrm{B}^3+\ldots$ et écrire y_t, MA(1), comme une autorégression infinie :

$$y_t=z_t+\theta y_{t-1}-\theta^2 y_{t-2}+\theta^3 y_{t-3}+\ldots$$

on dit qu'il est *inversible*. Observons que la condition d'inversibilité d'un MA(1) est parallèle à la condition de représentation causale d'un AR(1). Un MA(q) est dit inversible si on peut le représenter comme une autorégression infinie.

Propriété 4.2
Un MA(q) (4.17) est inversible si les racines de $1 + \theta_1 z + \theta_2 z^2 + \ldots + \theta_q z^q = 0$ *sont, en module, strictement supérieures à 1.*

Nous pouvons maintenant combiner les deux mécanismes, moyenne mobile et autorégression.

Définition 4.10 (Processus ARMA(p, q))
y_t *obéit à un modèle ARMA*(p, q) *s'il est stationnaire et vérifie :*

$$y_t = c + \phi_1 y_{t-1} + \ldots + \phi_p y_{t-p} + z_t + \theta_1 z_{t-1} + \ldots + \theta_q z_{t-q}, \quad z_t \sim \mathrm{BB}(0, \sigma_z^2) \quad (4.19)$$

avec c constante arbitraire, $\phi_p \neq 0$, $\theta_q \neq 0$, *et les polynômes* $1 - \phi_1 B - \ldots - \phi_p B^p$ *et* $1 + \theta_1 B + \ldots + \theta_q B^q$ *n'ont pas de racines communes.*

En utilisant l'opérateur retard, (4.19) s'écrit

$$(1 - \phi_1 \mathrm{B} - \phi_2 \mathrm{B}^2 - \ldots - \phi_p \mathrm{B}^p) y_t = c + (1 + \theta_1 \mathrm{B} + \theta_2 \mathrm{B}^2 + \ldots + \theta_q \mathrm{B}^q) z_t \quad (4.20)$$

y_t obéissant à (4.19) est stationnaire si, comme dans le cas des autorégressifs, les racines du polynôme d'autorégression $1 - \phi_1 z - \phi_2 z^2 - \ldots - \phi_p z^p = 0$ sont en module strictement supérieures à 1. Par un calcul identique à celui fait pour un AR(p), on obtient que $\mu = \mathsf{E}(y_t)$ vérifie

$$(1 - \phi_1 - \phi_2 - \ldots - \phi_p)\mu = c, \quad (4.21)$$

par la stationnarité, $1 - \phi_1 - \phi_2 - \ldots - \phi_p \neq 0$ et $\mu = c/(1 - \phi_1 - \phi_2 - \ldots - \phi_p)$. Ainsi (4.19) peut encore s'écrire :

$$y_t = \mu + \frac{1 + \theta_1 \mathrm{B} + \theta_2 \mathrm{B}^2 + \ldots + \theta_q \mathrm{B}^q}{1 - \phi_1 \mathrm{B} - \phi_2 \mathrm{B}^2 - \ldots - \phi_p \mathrm{B}^p} z_t \quad (4.22)$$

On peut alors écrire une représentation MA(∞) de la série :

$$y_t = \mu + \sum_{i=0}^{\infty} \psi_i z_{t-i}, \ \psi_0 = 1.$$

Par ailleurs, y_t, ARMA(p, q), est inversible si les racines de $\Theta(\mathrm{B})$ sont en module strictement supérieures à 1 et on peut écrire alors une représentation AR(∞) de la série :

$$y_t = c + \sum_{i=1}^{\infty} \pi_i y_{t-i} + z_t.$$

Remarques
– L'absence de racines communes dans (4.19) est une condition pour éviter la redondance des paramètres.

– Il arrive que certaines racines du polynôme autorégressif soient égales à 1. L'autorégressif est alors non stationnaire (voir le calcul de moyenne après 4.21) et on dit qu'il est *intégré* d'ordre d si 1 est d fois racine. La question est abordée au chapitre 5.
– La théorie de l'ajustement d'un modèle autorégressif à une série suppose qu'elle est stationnaire. Donc, dans la pratique, on ne devrait essayer d'ajuster un modèle autorégressif à une série que si elle est stationnaire, ce qu'en général on ignore au début de l'étude. Observons que $1-\phi_1-\phi_2-\ldots-\phi_p = 0$ indique que 1 est racine du polynôme d'autorégression. Il est donc pertinent, une fois ajusté un AR(p) à une série qu'on a supposée stationnaire, d'examiner si $\widehat{\phi_1}+\widehat{\phi_2}+\ldots+\widehat{\phi_p}$ n'est pas trop proche de 1, indice de possible non-stationnarité. D'autres garde-fous sont à notre disposition. Si l'on essaie par exemple d'ajuster un AR(p) à une série autorégressive non stationnaire ou proche de la non-stationnarité, l'algorithme d'optimisation donne souvent des messages signalant un mauvais fonctionnement.
– L'abréviation AR pour « autorégressif », renvoie normalement à une série stationnaire. Nous suivrons cette convention.
– Les poids ψ s'obtiennent formellement, voir Brockwell & Davis (1991). On trouve un développement théorique des processus ARMA dans de nombreux ouvrages comme Gourieroux & Monfort (1995, chap. 5), Brockwell & Davis (2002, chap. 2), Bourbonnais & Terraza (2008) ou Hamilton (1994, chap. 3).
– Dans certains ouvrages, logiciels et packages de R, les termes retardés de la partie MA sont retranchés :

$$y_t = \mu + (1 - \theta_1 \mathrm{B} - \theta_2 \mathrm{B}^2 - \ldots - \theta_q \mathrm{B}^q) z_t.$$

Il faut donc être vigilant dans la retranscription des résultats.

4.3 Fonctions d'autocorrélation

4.3.1 Fonction d'autocorrélation d'un AR

Partant de la représentation MA(∞) d'un AR(1), (4.14), on obtient :

$$\mathsf{var}(y_t) = \sigma_z^2(1 + \phi^2 + \phi^4 + \ldots) \quad = \quad \frac{\sigma_z^2}{1-\phi^2}. \tag{4.23}$$

La fonction d'autocorrélation de l'AR(1) est donc :

$$\rho_k = \phi^k, k = 0, 1, 2, \ldots \tag{4.24}$$

Cette fonction décroît exponentiellement vers 0, en oscillant si $\phi < 0$.

Exercice 4.2 (Fonction d'autocovariance d'un AR(p))
Vérifier que la fonction d'autocovariance d'un AR(p) obéit à :

$$\begin{aligned} \gamma_0 &= \phi_1\gamma_1 + \phi_2\gamma_2 + \ldots + \phi_p\gamma_p + \sigma_z^2, \\ \gamma_l &= \phi_1\gamma_{l-1} + \phi_2\gamma_2 + \ldots + \phi_p\gamma_{l-p},\ l \geq 1 \end{aligned} \tag{4.25}$$

et que la fonction d'autocorrélation d'un AR(p) obéit à :

$$\rho_l = \phi_1\rho_{l-1} + \phi_2\rho_{l-2} + \ldots + \phi_p\rho_{l-p},\ l \geq 1. \tag{4.26}$$

La fonction d'autocorrélation d'un AR(p) montre une décroissance exponentielle, avec ou sans oscillations, vers 0.
On appelle *équations de Yule-Walker*, la présentation matricielle des $p+1$ premières équations (4.25).

$$\boldsymbol{\Gamma}_p\boldsymbol{\phi} = \boldsymbol{\gamma}_p, \qquad \sigma_z^2 = \gamma_0 - \boldsymbol{\phi}'\boldsymbol{\gamma}_p, \tag{4.27}$$

où $\boldsymbol{\Gamma}_p = \{\gamma_{k-j}\}_{j,k=1}^p$ est une matrice $p \times p$, $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_p)'$ une matrice $p \times 1$ et $\boldsymbol{\gamma}_p = (\gamma_1, \ldots, \gamma_p)'$ une matrice $p \times 1$. Notons que si l'on connaît la matrice des autocovariances, ces équations forment un système linéaire dont la solution est $\boldsymbol{\phi}$ et σ_z^2.

4.3.2 Fonction d'autocorrélation d'un MA

Commençons par calculer les moments d'ordre 2 d'un MA(1). La variance de y_t définie par (4.18) est la variance d'une combinaison linéaire de variables non corrélées donc : $\mathsf{var}(y_t) = (1+\theta^2)\sigma_z^2$. De même, $\mathsf{cov}(y_t, y_{t-1}) = \mathsf{cov}(z_t+\theta z_{t-1}, z_{t-1}+\theta z_{t-2}) = \theta\sigma_z^2$. On voit que $\mathsf{cov}(y_t, y_{t-k}) = 0,\ k > 1$. En résumé, $\forall\theta$, le processus MA(1) défini par (4.18), stationnaire, de moyenne μ, a pour fonction d'autocorrélation :

$$\rho(k) = \begin{cases} 1 & \text{si } k = 0, \\ \frac{\theta}{1+\theta^2} & \text{si } k = 1, \\ 0 & \text{si } k > 1. \end{cases}$$

Etudiant la fonction réelle $x \to \frac{x}{1+x^2}$, on note que $|\frac{x}{1+x^2}| \leq 0.5$; on ne peut donc pas décrire des phénomènes à forte autocorrélation à l'aide d'un processus MA(1).
A partir de la définition (4.17), on obtient la fonction d'autocovariance d'un MA(q).

Exercice 4.3 (Fonction d'autocovariance d'un MA(q))
Vérifier que la ACF d'un MA(q) vérifie :

$$\gamma_h = \begin{cases} (1 + \theta_1^2 + \theta_2^2 + \ldots + \theta_q^2)\sigma_z^2 & \text{si } h = 0, \\ (\theta_h + \theta_{h+1}\theta_1 + \ldots + \theta_q\theta_{q-h})\sigma_z^2 & \text{si } 1 \leq h \leq q, \\ 0 & \text{si } h > q. \end{cases} \tag{4.28}$$

Ainsi, la fonction d'autocorrélation d'un processus MA(q) est nulle à partir de l'ordre $q+1$. Si on observe une trajectoire d'un MA(q), on peut donc s'attendre que l'ACF empirique de la série ne soit pas significativement différente de 0 au-delà de l'ordre q. Inversement, si une ACF empirique semble nulle à partir d'un certain ordre $q+1$, on peut penser qu'il s'agit de l'ACF d'une série MA(q). La formule de Bartlett, ci-dessous, donne de la rigueur à cette intuition.

Propriété 4.3 (Formule de Bartlett)
Pour une série linéaire dont l'ACF vérifie : $\rho_k = 0,\ k > m$, *on a pour* $k > m$ *:*

$$\widehat{\rho}_k \sim AN(0, \mathsf{var}(\widehat{\rho}_k)), \tag{4.29}$$

$$\mathsf{var}(\widehat{\rho}_k) \simeq \frac{1}{T}(1 + 2\rho_1^2 + \ldots + 2\rho_m^2). \tag{4.30}$$

Ce résultat étend la propriété (4.1). Il est précieux pour deviner (identifier) l'ordre de moyenne mobile convenable pour modéliser une série. En effet, en présence d'un corrélogramme empirique non significativement différent de 0 à partir d'un certain ordre $m+1$, on essaiera d'ajuster à la série correspondante un modèle dont l'ACF est nulle à partir de l'ordre $m+1$, un MA(m) par exemple (cf. section 4.6). Mais comment savoir que l'ACF empirique à partir de l'ordre $m+1$ est une estimation de 0 ? La formule de Bartlett permet de calculer des intervalles autour de 0 pour l'ACF d'un processus MA(m), à partir du décalage $m+1$: pour chaque retard $k > m$ on a en effet :

$$\widehat{\rho}_k \in (-1.96\sqrt{\frac{1}{T}(1 + 2\rho_1^2 + \ldots + 2\rho_m^2)}, +1.96\sqrt{\frac{1}{T}(1 + 2\rho_1^2 + \ldots + 2\rho_m^2)})$$

avec une probabilité d'environ 95%.
Supposons en particulier que le processus étudié est un bruit blanc, alors $\widehat{\rho}_k,\ k > 0$ doit appartenir à l'intervalle $-1.96/\sqrt{T}, +1.96/\sqrt{T}$ à 95% environ. En superposant le graphique de l'ACF $\widehat{\rho}_k$ et cet intervalle ou son approximation $-2/\sqrt{T}, +2/\sqrt{T}$, on peut voir si l'hypothèse de blancheur est raisonnable. On peut tracer ces intervalles pour une série supposée bruit blanc (cf. prop. 4.1). On représente habituellement ces intervalles sur les graphiques d'ACF empirique (voir par exemple figure 4.1). Dans la mise en pratique de la formule (4.30) on ne connaît pas les ρ_k, qu'on remplace par leurs estimations.

Exemple 4.3 `TacvfARMA()` de **FitARMA** ou `ARMAacf()` donnent la fonction d'autocovariance théorique d'un processus ARMA dont le bruit est de variance 1. Considérons le processus

$$y_t = -0.7y_{t-1} + 0.2y_{t-2} + z_t + 0.6z_{t-1},\ \ z_t \sim \mathrm{BB}(0, \sigma_z^2).$$

On obtient son ACF théorique jusqu'au retard 5 par `TacvfARMA()`

```
> require(FitARMA)
> phi0=c(-.7,.2); theta0=0.6
> g=TacvfARMA(phi=phi0,theta=-theta0,lag.max=5)
> g/g[1]
```

```
[1]  1.0000000 -0.3306452  0.4314516 -0.3681452  0.3439919
[6] -0.3144234
```

ou par `ARMAacf()`

```
> ARMAacf(ar=phi0,ma =theta0,lag.max=5)
```

```
         0          1          2          3          4
 1.0000000 -0.3306452  0.4314516 -0.3681452  0.3439919
         5
-0.3144234
```

Notons que **FitARMA** inverse le signe de la partie MA par rapport à la convention de R.

Dans l'exemple suivant, nous utilisons la fonction `arima()` pour estimer un modèle ARMA$(0, 2)$. Les principes de l'estimation de ces modèles sont brièvement rappelés à la section 4.5 où nous donnons également des détails sur `arima()`.

Exemple 4.4 Simulons une série de 200 observations suivant le modèle MA(2) :

$$y_t = (1 - 0.3\mathrm{B} + 0.6\mathrm{B}^2)z_t \qquad z_t \sim \mathrm{BBN}(0, 1.5).$$

```
> set.seed(219)
> y3=arima.sim(n=200,list(ma=c(-.3,.6)),sd=sqrt(1.5))
```

Maintenant ajustons un modèle MA(2) à la série simulée :

```
> mod0=arima(y3,order=c(0,0,2))
```

Les résultats sont dans la liste `mod0`. On obtient à l'écran les estimations par `summary(mod0)`. Comme on a ajusté le modèle qui a servi à la simulation, les résidus de l'ajustement, `residuals(mod0)`, devraient ressembler à un bruit blanc, c'est-à-dire avoir une ACF non significativement différente de zéro. Examinons l'ACF de ces résidus :

```
> acf(residuals(mod0),xlab="retard",main="")
```

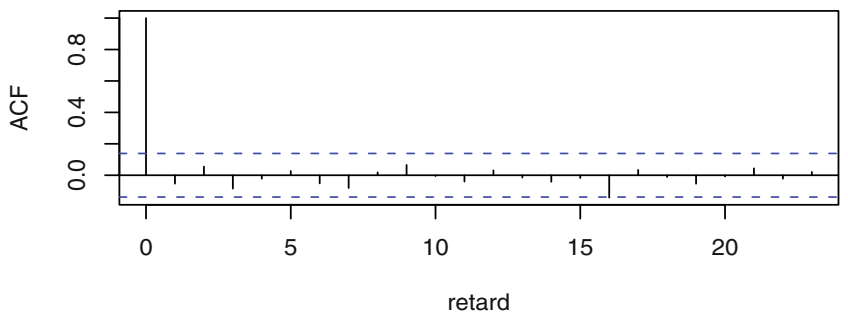

Fig. 4.1 – Fonction d'autocorrélation du résidu de l'ajustement d'un MA(2) à `y3`.

On note (fig. 4.1) que les autocorrélations sont bien incluses dans l'intervalle à 95% autour de zéro. Il peut arriver qu'avec une autre simulation, une autocorrélation sorte de l'intervalle. Dans ce cas on doit se demander si les valeurs hors de l'intervalle correspondent à des estimations de zéro accidentellement significatives ou à des estimations de valeurs effectivement non nulles.

4.4 Prévision

Nous présentons le principe de la prévision, sans entrer dans le détail des calculs. Le calcul de la prévision nous amène à introduire la fonction d'autocorrélation partielle, utile pour identifier le modèle ARMA d'une série (cf. section 4.6).

4.4.1 Principe

On dispose de $y_1, y_2, \ldots, y_T$, observations d'une série stationnaire de moyenne μ et de fonction d'autocovariance γ_l, $l = 0, 1, 2 \ldots$ et l'on veut la série en $T + h$, $h \geq 1$. Par convention, la prédiction de y_{T+h} connaissant le passé $y_T, y_{T-1}, \ldots, y_1$ de la série est la fonction de ces observations qui minimise l'erreur quadratique moyenne (EQM) de prévision $\mathsf{E}((y_{t+h} - g)^2)$ pour toutes les fonctions $g(.)$ du passé jusqu'en t. Ce minimum est atteint pour la fonction, espérance conditionnelle au passé, notée

$$\mathsf{E}(y_{t+h}|y_t, y_{t-1}, \ldots, y_1).$$

C'est le meilleur prédicteur de y_t à l'*horizon* h. Il est souvent difficile de calculer cette espérance conditionnelle, aussi se restreint-on à $g(.)$, fonction *linéaire* des observations passées. On appelle alors $g(.)$ *espérance conditionnelle linéaire* (EL) et la prédiction associée est le meilleur prédicteur linéaire, abrégé en BLP (*Best Linear Predictor*) de y_{T+h} ; nous le noterons $y_{T+h|T}$ ou $\mathsf{EL}(y_{T+h}|y_T, y_{T-1}, \ldots, y_1)$; cette question est présentée notamment par Gourieroux & Monfort (1995, chap. 3) et Hamilton (1994). Le BLP est de la forme : $a_0 + \sum_{k=1}^{T} a_k y_{T+1-k}$. Nous admettrons qu'il vérifie le système linéaire :

$$\begin{aligned} a_0 &= \mu(1 - \sum_{i=1}^{T} a_i), \\ \Gamma_T \boldsymbol{a}_T &= \gamma_T(h) \end{aligned} \tag{4.31}$$

où

$$\boldsymbol{a}_T = [a_1, \ldots, a_T]', \quad \Gamma_T = [\gamma_{i-j}]_{i,j=1}^{T}, \quad \gamma_T(h) = [\gamma_h, \gamma_{h+1}, \ldots, \gamma_{h+T-1}]'.$$

Propriété 4.4 (Meilleur prédicteur linéaire d'un processus stationnaire) *Etant donné $y_1, \ldots, y_T$, trajectoire d'une série stationnaire de moyenne μ, le BLP*

de y_{T+h} vérifie

$$\begin{aligned} y_{T+h|T} &= \mu + \sum_{i=1}^{T} a_i (y_{T+1-i} - \mu), && (4.32)\\ \mathsf{E}[y_{T+h} - y_{T+h|T}] &= 0, && (4.33)\\ \mathsf{E}[(y_{T+h} - y_{T+h|T}) y_t] &= 0,\ t = 1, \ldots, T, && (4.34)\\ \mathsf{E}[(y_{T+h} - y_{T+h|T})^2)] &= \gamma_0 - \boldsymbol{a}'_T \gamma_T(h). && (4.35) \end{aligned}$$

Remarques

- La deuxième équation dit que l'espérance mathématique du BLP est égale à celle de la quantité à prédire. Le BLP est dit *sans biais*.
- La troisième équation dit que l'erreur de prédiction est orthogonale à $y_t,\ t \leq T$.
- L'erreur quadratique de prévision à l'horizon h est donnée par (4.35). A l'horizon 1, cette erreur est aussi appelée *erreur quadratique moyenne*, EQM ou MSE (*Mean Square Error*).
- Si y_t est gaussien, le meilleur prédicteur se confond avec le meilleur prédicteur linéaire. Dans ce livre, à l'exception du chapitre 12 où les séries étudiées montrent de l'hétéroscédasticité conditionnelle et ne peuvent donc pas être considérées comme gaussiennes, espérance conditionnelle et espérance conditionnelle linéaire se confondent.
- Comparons (4.31) avec les équations de Yule-Walker (4.27) obtenues pour un AR(p). On voit que pour un AR(p), $(a_1, \ldots, a_p) \equiv (\phi_1, \ldots, \phi_p)$, autrement dit, la prévision à l'horizon 1, basée sur p instants passés, est l'équation d'autorégression, au bruit près. Nous reviendrons sur cette observation dans la prochaine section.

4.4.2 Fonction d'autocorrélation partielle

Considérons une série stationnaire $\{y_t\}$ centrée et ses régressions linéaires sur son passé résumé à une, deux, trois ... observations :

$$\begin{aligned} y_t &= \phi_{1,1} y_{t-1} + u_{1t} && (4.36)\\ y_t &= \phi_{1,2} y_{t-1} + \phi_{2,2} y_{t-2} + u_{2t}\\ y_t &= \phi_{1,3} y_{t-1} + \phi_{2,3} y_{t-2} + \phi_{3,3} y_{t-3} + u_{3t}\\ &\vdots \end{aligned}$$

Par exemple, $\phi_{1,2} y_{t-1} + \phi_{2,2} y_{t-2}$ désigne le BLP de y_t connaissant y_{t-1} et y_{t-2}, et s'obtient en résolvant (4.31) où $a_0 = 0$, c'est-à-dire :

$$\begin{bmatrix} \gamma_0 & \gamma_1 \\ \gamma_1 & \gamma_0 \end{bmatrix} \begin{bmatrix} \phi_{1,2} \\ \phi_{2,2} \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix}.$$

Considérons les $\phi_{k,k},\ k = 1, 2, \ldots$ Ils ont la même interprétation que les coefficients d'une régression linéaire classique : $\phi_{k,k}$ représente l'apport d'explication de y_{t-k} à y_t, *toutes choses égales par ailleurs*, c'est-à-dire étant donné qu'on régresse également sur $y_{t-1}, \ldots, y_{t-k+1}$. Ils forment la *fonction d'autocorrélation partielle* que nous abrégerons en PACF.
Supposons en particulier que y_t soit autorégressif, un AR(3) pour fixer les idées, alors il est clair que y_{t-4} n'apporte rien de plus que $y_{t-1},\ y_{t-2},\ y_{t-3}$ pour expliquer y_t et on montre en effet que pour un AR(3), $\phi_{k,k} = 0,\ k > 3$.
D'une façon générale on a :

Propriété 4.5

(a) *La PACF d'un AR(p) est nulle à partir de l'ordre* $p + 1$.

(b) *La PACF d'un processus qui a une composante moyenne mobile a une décroissance exponentielle.*

Ainsi la PACF d'un ARMA(p, q), $q > 0$ présente une décroissance exponentielle ou sinusoïdale amortie.

Calcul de la PACF. L'algorithme de Durbin-Levinson (voir Brockwell & Davis, 2002 ou Shumway & Stoffer, 2006) calcule itérativement la PACF à partir de l'ACF. Les estimations $\widehat{\phi}_{k,k},\ k = 1, 2, \ldots$ obtenues en appliquant l'algorithme de Durbin-Levinson à l'ACF empirique forment la *fonction d'autocorrélation partielle empirique*. L'algorithme de Durbin-Levinson est programmé dans la fonction `PacfDL()` de **FitAR**. D'un point de vue pratique, on pensera qu'une série suit un AR(p) si les $\widehat{\phi}_{k,k} \simeq 0,\ k > p$, précisément :

Propriété 4.6
Si y_t *est un AR(p), alors :*

(a) $\widehat{\phi}_{p,p}$ *converge vers* $\phi_{p,p}$ *quand* $T \to \infty$,

(b) $\widehat{\phi}_{l,l}, \forall l > p$ *converge vers* 0 *quand* $T \to \infty$,

(c) $\mathsf{var}(\widehat{\phi}_{l,l}) \ \simeq 1/T\ \forall l > p$.

Si la PACF empirique d'une série n'est plus significativement différente de zéro à partir d'un certain ordre k, on essaiera de lui ajuster un modèle AR$(k-1)$.

Exemple 4.5 Calculons à l'aide de l'algorithme de Durbin-Levinson la PACF d'un AR(2)

$$y_t = -0.7y_{t-1} + 0.2y_{t-2} + z_t \tag{4.37}$$

puis les coefficients $\boldsymbol{a}_T$ jusqu'à $T = 4$ du système (4.31). Vu la discussion de (4.36), nous nous attendons à trouver, pour $k > 2$, des coefficients a_k nuls. Nous calculons d'abord la fonction d'autocovariance théorique, `g`, à l'aide de `TacvfARMA()`. La fonction `PacfDL()` a comme argument la fonction d'autocorrélation, `g/g[1]`.

```
> g=TacvfARMA(phi=c(-.7,.2),lag.max=4)
> (a=PacfDL(g/g[1],LinearPredictor=TRUE))
```

```
$Pacf
[1] -8.750000e-01  2.000000e-01  7.401487e-16  1.644775e-16

$ARCoefficients
[1] -7.000000e-01  2.000000e-01  8.552829e-16  1.644775e-16

$ResidualVariance
[1] 0.225
```

```
> g[1]-t(as.matrix(a$ARCoefficients))%*%as.matrix(g[-1])

     [,1]
[1,]    1
```

`a$Pacf` est le vecteur des $\phi_{k,k}$, effectivement nuls à partir de $k = 3$. Le vecteur `a$ARCoefficients` est le vecteur $(a_1, \ldots, a_4)$, avec $(a_1, a_2) = (\phi_1, \phi_2)$ et $a_k = 0$ à partir de $k = 3$. La dernière ligne de code calcule la variance de l'erreur de prédiction à l'horizon 1, expression (4.35). Elle vaut 1, comme on s'y attendait, car le bruit utilisé pour calculer la fonction d'autocovariance par `TacvfARMA()` est supposé de variance 1.

Exercice 4.4
Simuler une trajectoire de 200 valeurs d'un processus autorégressif obéissant à (4.37) et calculer la PCF empirique jusqu'au retard 4. Comparer avec l'exemple précédent.

4.4.3 Prévision d'un modèle autorégressif

Nous considérons $\{y_t\}$ AR(p), avec z_t, bruit blanc gaussien. Après la discussion qui précède, nous admettrons les résultats suivants.

Prévision à l'horizon 1.
L'espérance linéaire de

$$y_{t+1} = \phi_0 + \phi_1 y_t + \phi_2 y_{t-1} + \ldots + \phi_p y_{t+1-p} + z_{t+1},$$

conditionnellement à son passé est :

$$y_{t+1|t} = \phi_0 + \phi_1 y_t + \phi_2 y_{t-1} + \ldots + \phi_p y_{t+1-p},$$

l'erreur de prédiction associée est donc

$$e_t(1) = y_{t+1} - y_{t+1|t} = z_{t+1}$$

et l'erreur quadratique moyenne de prévision est égale à la variance, σ_z^2. C'est la variance de y_{t+1} conditionnellement au passé de la série. Dans ce modèle, elle est indépendante de t. Le bruit blanc z_{t+1} peut être interprété comme la correction à la prédiction mécanique de y_{t+1} par les valeurs les plus récemment observées. On appelle d'ailleurs souvent *innovation* le bruit blanc z_t dans les séries représentables par un filtre linéaire et causal.

Prévision à l'horizon 2.
La prévision de

$$y_{t+2} = \phi_0 + \phi_1 y_{t+1} + \phi_2 y_t + \ldots + \phi_p y_{t+2-p} + z_{t+2}$$

connaissant le passé $y_t, y_{t-1}, \ldots$ est toujours l'espérance conditionnelle linéaire par rapport à ce passé. Comme l'espérance d'une somme de v.a. est la somme des espérances :

$$y_{t+2|t} = \phi_0 + \phi_1 y_{t+1|t} + \phi_2 y_t + \ldots + \phi_p y_{t+2-p}$$

et l'erreur de prédiction est :

$$e_t(2) = y_{t+2} - y_{t+2|t} = z_{t+2} + \phi_1 e_t(1) = z_{t+2} + \phi_1 z_{t+1},$$

d'espérance nulle et de variance $\sigma_z^2(1 + \phi_1^2)$. La variance de l'erreur de prévision augmente évidemment avec l'horizon de prévision.
La prévision d'un AR(p) à un horizon h quelconque est :

$$y_{t+h|t} = \phi_0 + \phi_1 y_{t+h-1|t} + \phi_2 y_{t+h-2|t} + \ldots + \phi_p y_{t+h-p|t} \tag{4.38}$$

où $y_{t+h-k|t} = y_{t+h-k}$ si $h - k \leq 0$.
On peut montrer que pour un AR(p), $y_{t+h|t} \to \mathsf{E}(y_t)$ quand $h \to \infty$. C'est la propriété dite de *retour à la moyenne* et la variance de l'erreur de prévision tend vers la variance de y_t. Dans la pratique, on remplace les ϕ par leurs estimations et on ne tient pas compte de la variabilité de ces dernières pour établir la prévision, qu'on note alors $\widehat{y}_{t+h|t}$. Nous n'aurons pas besoin de préciser davantage les notions d'espérance conditionnelle et d'espérance conditionnelle linéaire.

4.4.4 Prévision d'un MA(q)

On a observé y_t, MA(q), jusqu'en t et on veut le prédire en $t + h$, $h \geq 1$. Le passé est engendré de façon équivalente par $y_t, y_{t-1}, \ldots$ ou par $z_t, z_{t-1}, \ldots$.

Prévision à l'horizon 1.

$$y_{t+1} = \mu + z_{t+1} + \theta_1 z_t + \theta_2 z_{t-1} + \ldots + \theta_q z_{t+1-q},$$

donc

$$y_{t+1|t} = \mathsf{E}(y_{t+1}|y_t, y_{t-1}, \ldots) = \mu + \theta_1 z_t + \theta_2 z_{t-1} + \ldots + \theta_q z_{t-q}$$

et l'erreur de prédiction associée est :

$$e_t(1) = y_{t+1} - y_{t+1|t} = z_{t+1},$$

de variance σ_z^2. Pour un horizon $h > q$, on voit que la prévision est μ, le retour à la moyenne se fait en q étapes. Cette formulation de la prévision d'un MA n'est pas très pratique car elle fait intervenir l'erreur, non observée. Shumway & Stoffer (2006) et Brockwell & Davis (2002) présentent la prévision en détail et l'illustrent de nombreux exemples. Nous pratiquerons la prévision sur des modèles préalablement estimés dans les chapitres étudiant des cas réels.

4.5 Estimation

Nous nous limitons à quelques principes qui permettront de comprendre le vocabulaire des méthodes proposées par R et d'apercevoir la complexité de l'estimation d'un modèle ayant une composante MA. Pour la clarté des notations, il est nécessaire dans ce paragraphe de distinguer une v.a. notée en capitales et sa réalisation notée en minuscules.

Fonction de vraisemblance d'un processus gaussien AR(1). On dispose de la série $y_t, t = 1, 2, \ldots, T$, observation de $\{Y_t\}$ AR(1) :

$$Y_t = c + \phi Y_{t-1} + Z_t, \qquad Z_t \sim \text{BBN}(0, \sigma^2).$$

On sait qu'alors Y_t suit une loi normale avec : $\mathsf{E}(Y_t) = \mu = \frac{c}{1-\phi}$, $\mathsf{var}(Y_t) = \frac{\sigma^2}{1-\phi^2}$. La fonction de densité de probabilité (f.d.p.) de Y_1 est :

$$f_{Y_1}(y_1; (c, \phi, \sigma^2)) = \frac{1}{\sqrt{2\pi\sigma^2/(1-\phi^2)}} \exp[-\frac{(y_1 - c/(1-\phi))^2}{2\sigma^2/(1-\phi^2)}]$$

Sachant que $Y_1 = y_1$, Y_2 est normalement distribué de moyenne $c+\phi y_1$, de variance σ^2 d'où la f.d.p. de Y_2 conditionnelle au passé :

$$f_{Y_2|Y_1=y_1}(y_2; (c, \phi, \sigma^2)) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp[-\frac{(y_2 - c - \phi y_1))^2}{2\sigma^2}]$$

on en déduit la f.d.p. conjointe du couple (Y_1, Y_2) :

$$f_{Y_1,Y_2}(y_1, y_2; (c, \phi, \sigma^2)) = f_{Y_1}(y_1; (c, \phi, \sigma^2)) f_{Y_2|Y_1=y_1}(y_2; (c, \phi, \sigma^2)).$$

On observe par ailleurs que Y_t ne dépend explicitement que de y_{t-1} :

$$\begin{aligned} f_{Y_t|Y_{t-1}=y_{t-1},\ldots,Y_1=y_1}(y_t; (c, \phi, \sigma^2)) &= f_{Y_t|Y_{t-1}=y_{t-1}}(y_t; (c, \phi, \sigma^2)) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp[-\frac{(y_t - c - \phi y_{t-1}))^2}{2\sigma^2}]. \end{aligned}$$

La f.d.p. conjointe des observations est donc :

$$f_{Y_1,\ldots,Y_T}(y_1, \ldots, y_T; (c, \phi, \sigma^2)) = f_{Y_1}(y_1; \theta) \prod_{t=2}^{T} f_{Y_t|Y_{t-1}=y_{t-1}}(y_t; (c, \phi, \sigma^2)).$$

D'où on obtient la (fonction) log vraisemblance

$$\begin{aligned} \mathcal{L}(c, \phi, \sigma^2) &= -\frac{1}{2}\ln(2\pi\frac{\sigma^2}{1-\phi^2}) - \frac{1}{2}\frac{(y_1 - c/(1-\phi))^2}{\sigma^2/(1-\phi^2)} - \frac{T-1}{2}\ln(2\pi\sigma^2) \\ &\quad -\frac{1}{2}\sum_{t=2}^{T}\frac{(y_t - c - \phi y_{t-1})^2}{\sigma^2}. \end{aligned} \tag{4.39}$$

- ▶ Si on néglige le log déterminant dans cette expression, sa maximisation donne l'*estimateur des moindres carrés inconditionnels* de θ.
- ▶ Si on travaille conditionnellement à la première valeur y_1, la log vraisemblance se simplifie en la *log vraisemblance conditionnelle* :

$$\mathcal{L}_c(\theta) \quad = \quad -\frac{T-1}{2}\ln(2\pi\sigma^2) - \sum_{t=2}^{T}\frac{(y_t - c - \phi y_{t-1})^2}{2\sigma^2}. \qquad (4.40)$$

  Sa maximisation donne l'*estimateur du maximum de vraisemblance conditionnelle*.
- ▶ Si dans (4.40) on néglige le log déterminant, on obtient l'*estimateur des moindres carrés conditionnels* de c et ϕ, ce qui correspond à `method='CSS'` dans `arima()`.

Valeurs initiales et méthode des moments. En remplaçant les γ par leurs estimations dans les équations de Yule-Walker et résolvant alors la version empirique de (4.27), on obtient des estimations des ϕ par la méthode des moments. Ces estimations servent de valeurs initiales pour l'estimation par maximum de vraisemblance.

Fonction de vraisemblance d'un processus gaussien MA(1). On dispose de la série $y_t, t = 1, 2, \ldots, T$, observation de $\{Y_t\}$ MA(1) :

$$Y_t = \mu + Z_t + \theta Z_{t-1}, \qquad Z_t \sim \text{BBN}(0, \sigma^2).$$

Si on connaît z_{t-1}, alors la loi de Y_t sachant que $Z_{t-1} = z_{t-1}$ est $\text{N}(\mu + \theta z_{t-1}, \sigma^2)$. Supposons que $Z_0 = 0$, alors étant donné l'observation de Y_1 on peut déduire la valeur de Z_1 : $z_1 = y_1 - \mu$. Ensuite $Y_2 = \mu + Z_2 + \theta z_1$ permet d'obtenir $z_2 = y_2 - \mu - \theta z_1$. On obtient ainsi la loi conditionnelle de Y_2 sachant que $Z_0 = 0, Y_1 = y_1$. Sa f.d.p. est :

$$f_{Y_2|Y_1=y_1,Z_0=0}(y_2; \theta, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(y_2 - \mu - \theta z_1)^2}{2\sigma^2}).$$

Ensuite connaissant z_1 et y_2 on peut calculer $z_2 = y_2 - \mu - \theta z_1$ etc. Ainsi ayant fixé la valeur de Z_0 (ici à la moyenne) et disposant des observations $y_1, \ldots, y_T$ on peut calculer *pour chaque valeur de* (θ, μ, σ^2) : $z_1 = y_1 - \mu,\ z_2 = y_2 - \mu - \theta z_1, \ldots, z_t = y_t - \mu - \theta z_{t-1}$ et la distribution conditionnelle de $Y_t|Y_{t-1} = y_{t-1}, \ldots, Y_1 = y_1, Z_0 = 0$. Sa f.d.p. est :

$$f_{Y_t|Y_{t-1}=y_{t-1},\ldots Y_1=y_1,Z_0=0}(y_t; \theta, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp[-\frac{1}{2\sigma^2}(y_t - \mu - \theta z_{t-1})^2].$$

La f.d.p. conjointe de $Y_1, \ldots, Y_T|Z_0 = 0$ est donc :

$$f_{Y_1|Z_0=0}(y_1; \theta, \mu, \sigma^2)\prod_{t=2}^{T} f_{Y_t=y_t|Y_{t-1}=y_{t-1},\ldots Y_1=y_1,Z_0=0}(y_t; \theta).$$

La log vraisemblance est :

$$\mathcal{L}(\theta, \mu, \sigma^2) = -\frac{T}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{t=1}^{T}(y_t - \mu - \theta z_{t-1})^2.$$

Le calcul a reposé sur le choix de la valeur de Z_0. On peut décider, une fois les z_t obtenus, de retourner le temps pour calculer z_0 et faire éventuellement plusieurs allers-retours, ce qui change l'estimation des paramètres. C'est pourquoi il est très difficile d'obtenir exactement les mêmes résultats pour le même modèle sur la même série avec deux logiciels.

Note. L'estimation des ARMA peut se faire par la fonction `arima()` de **stats**. Cette fonction autorise l'estimation de modèles ARMAX (cf. section 4.5.3). Différents clones de `arima()`, utilisant notamment les mêmes algorithmes d'estimation, la complètent. Par exemple : `Arima()` de **forecast** gère mieux les constantes dans les séries intégrées (section 4.2), `arimax()` de **TSA** permet d'estimer des modèles d'intervention (section 7.4). Shumway & Stoffer (2006, section 3.6) présentent en détail les techniques d'estimation dans les ARMA. Par défaut, `Arima()` estime par maximum de vraisemblance. La fonction `ar()` propose différents algorithmes pour estimer les modèles AR.
McLeod & Zhang (2008) présentent des méthodes précises et rapides pour l'étude des ARMA, implémentées dans **FitARMA** et **ltsa**, mais qui ne permettent pas d'estimer les modèles ARMAX.

4.5.1 Exemples

Examinons maintenant la mise en pratique de l'estimation des ARMA dans R. Les fonctions de R pour l'estimation des ARMA, basées sur `arima()`, ont au moins comme arguments la série sur laquelle on estime le modèle et le modèle à estimer. Celui-ci est décrit au minimum par un vecteur donnant dans l'ordre : p, d, q, c'est-à-dire l'ordre d'autorégression, l'ordre de différenciation [3] et l'ordre de moyenne mobile. Nous aurons l'occasion d'utiliser d'autres arguments de ces fonctions, mais il est d'ores et déjà instructif de consulter l'aide en ligne de `Arima()`.

Exemple 4.1 (Estimation sur `y1`) L'AR(1) suivant lequel on a simulé `y1`, expression (4.7a), est de moyenne nulle ; il vaut mieux le préciser à la fonction `Arima()` par l'option `include.mean = FALSE`. On écrit :

```
> (my1=Arima(y1,order=c(1,0,0),include.mean=FALSE))

Series: y1
ARIMA(1,0,0) with zero mean
...
Coefficients:
```

3. Des séries non stationnaires peuvent donner par différenciation à l'ordre $d = 1, 2, \ldots$, une série stationnaire. C'est le sujet du chapitre suivant. Pour l'instant, $d = 0$.

```
          ar1
      -0.6324
s.e.   0.0776

sigma^2 estimated as 3.050:  log likelihood = -197.91
AIC = 399.82   AICc = 399.95   BIC = 405.04
```

La fonction `summary()` appliquée à `my1` donne quelques résultats complémentaires. Les valeurs ajustées, les résidus, la matrice des covariances des estimateurs sont dans `my1` ; `str(my1)` nous renseigne sur leur emplacement et la façon de les récupérer. On peut calculer des t-statistiques approchées en effectuant le quotient, paramètre estimé/estimation de l'écart type de l'estimateur (voir la discussion qui suit l'expression (3.11), chap. 3). La fonction `t_stat()` de **caschrono** effectue ces quotients et donne les p-values approchées. Mais ces p-values n'ont de sens que si le résidu peut être considéré comme un bruit blanc. Il faut donc, avant de commenter les p-values des paramètres, tester la blancheur du résidu. Ce résidu résultant d'un ajustement à un paramètre, on indique la perte de ddl à la fonction `Box.test.2()`, par l'option `fitdf=1` (voir la discussion après l'expression (4.6)).

```
> ret=c(3,6,9,12)
> aa=Box.test.2(residuals(my1),nlag=ret,type="Ljung-Box",decim=4,fitdf=1)
> colnames(aa)= c("Retard","p-val.")
> t(aa)

         [,1]   [,2]   [,3]    [,4]
Retard 3.0000 6.0000 9.0000 12.0000
p-val. 0.8164 0.9647 0.9889  0.9812
```

Les p-values sont élevées pour les 4 retards qu'on a choisis. A la section 4.1.2, on avait testé et, évidemment rejeté, la blancheur de la série `y1`. Maintenant, une fois ajusté un modèle correct à cette série, il est normal d'accepter la blancheur du résidu de cet ajustement. On peut à présent tester la significativité du coefficient. On peut effectuer ce test en calculant la statistique directement. L'estimateur est dans `m1$coef` et sa variance dans `my1$var.coef` ; la statistique du test de significativité est `my1$coef/my1$var.coef^.5 =` -8.1549. La p-value est approximativement la probabilité qu'une v.a. $\mathcal{N}(0,1)$ dépasse ce quotient en valeur absolue. Comme elle est très faible, on refuse l'hypothèse que le coefficient d'autorégression soit non significatif. On peut aussi utiliser `t_stat()` qui fait exactement ce travail :

```
> t_stat(my1)

             ar1
t.stat -8.154936
p.val   0.000000
```

La p-value est directement fournie. `summary()` d'un objet en sortie de `Arima()` fournit de nombreux résultats dont ces t-statistiques et les p-values, mais, comme nous l'avons vu, leur examen n'a de sens qu'une fois acceptée la blancheur du résidu (cf. section 6.1.1).

Exemple 4.2 (Estimation sur y2) Comment estimer l'AR(12) suivant lequel on a simulé y2, expression (4.7b) ? Il s'agit d'un autorégressif d'ordre 12 dont on sait que les 11 premiers coefficients sont nuls. Si on écrit

```
> (my2a=Arima(y2,order=c(12,0,0),include.mean=FALSE))

Series: y2
ARIMA(12,0,0) with zero mean
...
Coefficients:
          ar1      ar2      ar3     ar4      ar5      ar6
      -0.0489  -0.0368  -0.0661  0.0875  -0.0295  -0.0721
s.e.   0.0748   0.0743   0.0756  0.0761   0.0755   0.0843
         ar7      ar8     ar9     ar10     ar11     ar12
      0.0291  -0.0190  0.0741  -0.0041  -0.1153  -0.6661
s.e.  0.0756   0.0771  0.0768   0.0774   0.0789   0.0753

sigma^2 estimated as 3.964:  log likelihood = -214.82
AIC = 455.63   AICc = 459.86   BIC = 489.5
```

On estime certes un AR(12) mais avec des estimations de 11 coefficients qu'on sait nuls (qu'on voit d'ailleurs non significatifs) et dont la présence va diminuer la qualité de l'estimation de ϕ_{12}. Examinons l'aide en ligne de `Arima()`. Une solution consiste à utiliser l'option `fixed` qui permet de fixer la valeur de certains paramètres. Habituellement, il y a un paramètre de plus que les p qui décrivent l'autorégressif : l'intercept. Mais nous savons que la moyenne est nulle, c'est pourquoi nous avons choisi l'option `include.mean=FALSE`. En résumé, l'option est donc `fixed=c(rep(0,11),NA)` (on affecte `NA` aux positions des paramètres à estimer et la valeur 0 aux positions des paramètres contraints à 0). L'estimation s'obtient finalement par :

```
> (my2a=Arima(y2,order=c(12,0,0),include.mean=FALSE,fixed=c(rep(0,11),NA)))

Series: y2
ARIMA(12,0,0) with zero mean
...
Coefficients:
      ar1  ar2  ar3  ar4  ar5  ar6  ar7  ar8  ar9  ar10
        0    0    0    0    0    0    0    0    0     0
s.e.    0    0    0    0    0    0    0    0    0     0
      ar11     ar12
         0  -0.6918
s.e.     0   0.0712

sigma^2 estimated as 4.2:  log likelihood = -217.56
AIC = 439.11   AICc = 443.34   BIC = 472.98
```

On peut observer que l'écart type de l'estimateur du paramètre du retard 12 a sensiblement diminué par rapport à la version précédente. Anticipant sur le paragraphe suivant ou révisant ce qui a été dit autour du modèle (1.3), on voit que (4.7b) définit un mécanisme saisonnier, situation qu'on décrit en donnant les

valeurs qui conviennent dans la liste suivante `seasonal = list(order = c(0, 0, 0), period = NA)`. La période de la saisonnalité est 12 et l'ordre d'autorégression dans la saisonnalité est 1, d'où la commande

```
> (my2a=Arima(y2,include.mean=FALSE,seasonal=list(order=c(1,0,0),period=12)))

Series: y2
ARIMA(0,0,0)(1,0,0)[12] with zero mean
...
Coefficients:
         sar1
      -0.6918
s.e.   0.0713

sigma^2 estimated as 4.2:  log likelihood = -217.56
AIC = 439.11   AICc = 439.23   BIC = 444.32
```

4.5.2 Modèle ARMA saisonnier (modèle SARMA)

Imaginons une série mensuelle stationnaire y_t montrant une saisonnalité, par exemple la série y_t du chapitre 1, expression (1.3), qui obéit à :

$$y_t - 50 = 0.9(y_{t-12} - 50) + z_t - 0.7z_{t-1},$$

où les z_t sont i.i.d. $\mathcal{N}(0, 17.5)$.
Supposons qu'on modélise la dépendance d'un mois sur un ou deux mois précédents (sans s'occuper de l'effet saisonnier) et qu'on adopte un ARMA(p, q) :

$$\Phi(\mathrm{B})y_t = c + \Theta(\mathrm{B})b_t.$$

Il est fort probable, si la série présente une saisonnalité, que le résidu $\widehat{b_t}$ ne sera pas blanc, mais aura une structure de corrélation saisonnière. On peut envisager deux traitements de cette « non-blancheur » : ou bien on ajoute des termes de retard dans les polynômes Φ et Θ, ou bien on modélise b_t par un ARMA dont l'unité de temps est la période de la saisonnalité, 12 par exemple. Poursuivons dans cette direction et supposons la modélisation suivante :

$$b_t = \frac{\Theta_s(\mathrm{B}^s)}{\Phi_s(\mathrm{B}^s)} z_t, \tag{4.41}$$

où s désigne la période (ici, $s = 12$). Ce qui donne :

$$\Phi_s(\mathrm{B}^s)\Phi(\mathrm{B})y_t = c_1 + \Theta(\mathrm{B})\Theta_s(\mathrm{B}^s)z_t, \tag{4.42}$$

avec $z_t \sim \mathrm{BB}(0, \sigma^2)$ et $c_1 = c\,\Phi_s(1)$, où $\Phi(\mathrm{B})$, $\Theta(\mathrm{B})$, $\Phi_s(\mathrm{B}^s)$, $\Theta_s(\mathrm{B}^s)$ sont respectivement des polynômes de degrés p, q en B et P, Q en B^s. On dit que y_t est un SARMA$(p, q)(P, Q)_s$ s'il vérifie (4.42) et est stationnaire.
Stationnarité et inversibilité tiennent sous les conditions suivantes :

– y_t est stationnaire si les racines des polynômes $\Phi(\mathrm{B})$ et $\Phi_s(\mathrm{B}^s)$ sont, en module, strictement supérieures à 1 ;
– y_t est inversible si les racines des polynômes $\Theta(\mathrm{B})$ et $\Theta_s(\mathrm{B}^s)$ sont, en module, strictement supérieures à 1.

Observons que (4.42) est en fait un ARMA$(p + sP, q + sQ)$, mais avec des trous et une paramétrisation parcimonieuse.

Exemple 4.6 Supposons y_t stationnaire, obéissant à

$$y_t = \frac{(1 + \theta_1 \mathrm{B})(1 + \Theta_1 \mathrm{B}^{12})}{(1 - \phi_1 \mathrm{B} - \phi_2 \mathrm{B}^2)(1 - \Phi_1 \mathrm{B}^{12})} z_t,\ z_t \sim \mathrm{BB}(0, \sigma_z^2).$$

y_t est un SARMA$(2, 1)(1, 1)_{12}$. Développons les différents polynômes, nous obtenons :

$$y_t = \frac{1 + \theta_1 \mathrm{B} + \Theta_1 \mathrm{B}^{12} + \theta_1 \Theta_1 \mathrm{B}^{13}}{1 - \phi_1 \mathrm{B} - \phi_2 \mathrm{B}^2 - \Phi_1 \mathrm{B}^{12} + \phi_1 \Phi_{12} \mathrm{B}^{13} + \phi_2 \Phi_{12} \mathrm{B}^{14}} z_t,$$

ainsi, y_t est également un ARMA$(14, 13)$ dont 9 coefficients AR et 10 coefficients MA sont nuls et dont les coefficients aux retards 13 et 14 sont des fonctions non linéaires d'autres coefficients.

Exemple 4.7 Considérons le SARMA$(1, 2)(1, 0)_4$:

$$y_t = 4 + \frac{1 + 0.6\mathrm{B}^2}{(1 + 0.8\mathrm{B})(1 - 0.7\mathrm{B}^4)} z_t, \qquad z_t \sim \mathrm{BBN}(0, 1.5) \tag{4.43}$$

et examinons ses ACF et PACF ainsi que leurs versions empiriques basées sur une série de 200 observations de ce processus. y_t est une série stationnaire saisonnière de saisonnalité 4. Pour le simuler nous commençons par effectuer le produit des polynômes d'autorégression, ce qui donne un terme autorégressif d'ordre 5.

```
> set.seed(7392)
> (autopol=polynomial(c(1,0.8))*polynomial(c(1,0,0,0,-0.7)))

1 + 0.8*x - 0.7*x^4 - 0.56*x^5

> yd=arima.sim(n=200,list(ar=-autopol[-1],ma=c(0,0.6)),sd=sqrt(1.5))
> yd=yd+4
> acf.th=ARMAacf(ar=-autopol[-1],ma=c(0,0.6),lag.max=20,pacf=FALSE)
> pacf.th=ARMAacf(ar=-autopol[-1],ma=c(0,0.6),lag.max=20,pacf=TRUE)
```

Il est utile de noter la façon dont R représente les polynômes : il n'imprime pas les termes nuls, mais par contre un polynôme de degré k est bien stocké comme un vecteur de $k + 1$ termes, du degré 0 au degré k.
La figure 4.2 représente les ACF et PACF. On observe une nette composante autorégressive d'ordre 5 à laquelle se superpose un terme MA qui donne des valeurs proches des limites de la bande autour de 0 aux retards 6, 7 et 9.

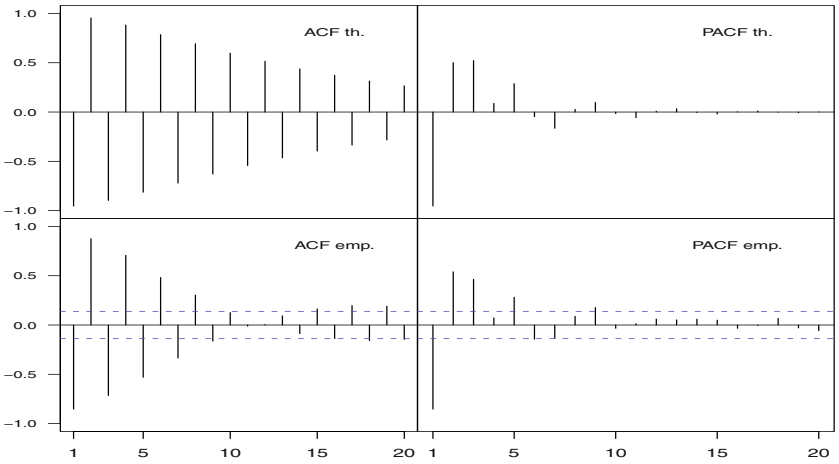

Fig. 4.2 – Fonctions d'autocorrélation d'un SARMA$(1,2)(1,0)_4$.

Il est difficile de détecter la saisonnalité de cette série sans connaissances a priori, aussi traçons-nous un lag plot de `yd` (fig. 4.3)

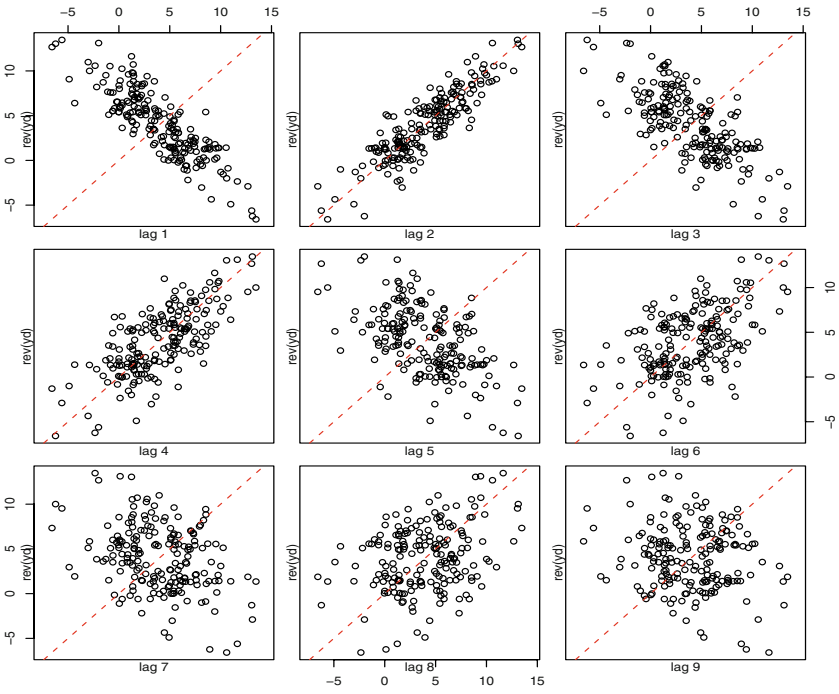

Fig. 4.3 – Lag plot du SARMA (4.43).

obtenu par :

```
> lag.plot(rev(yd),9,layout=c(3,3),do.lines=FALSE,diag.col="red")
```

Il montre un diagramme de dispersion plus allongé au retard 4 qu'au retard 5. Il suggère la saisonnalité 4, plus clairement que ne le fait l'ACF, mais, si on n'essaie pas plusieurs modélisations suggérées par le lag plot et les ACF et PACF, il est difficile de détecter à coup sûr la saisonnalité de cette série. Nous passons à l'estimation d'un modèle SARMA$(1,2)(1,0)_4$ sur `yd` par :

```
> (msarma=Arima(yd,order=c(1,0,2),seasonal=list(order=c(1,0,0),period=4)))

Series: yd
ARIMA(1,0,2)(1,0,0)[4] with non-zero mean
...
Coefficients:
          ar1      ma1     ma2    sar1  intercept
      -0.6864  -0.0220  0.6108  0.7021     4.1355
s.e.   0.0650   0.0743  0.0703  0.0643     0.2814

sigma^2 estimated as 1.717:  log likelihood = -339.84
AIC = 691.69   AICc = 692.12   BIC = 711.48
```

Or, dans le modèle de `yd`, (4.43), le paramètre de retard 1 dans la partie MA est nul. On peut donc améliorer l'estimation en tenant compte de ce fait. Repérant la position des différents paramètres sur la sortie, on contraint à 0 le terme MA1 :

```
> (msarma=Arima(yd,order=c(1,0,2),
+  seasonal=list(order=c(1,0,0),period=4),fixed=c(NA,0,NA,NA,NA)))

Series: yd
ARIMA(1,0,2)(1,0,0)[4] with non-zero mean
...
Coefficients:
          ar1  ma1     ma2    sar1  intercept
      -0.6972    0  0.6029  0.6971     4.1331
s.e.   0.0531    0  0.0662  0.0628     0.2777

sigma^2 estimated as 1.718:  log likelihood = -339.89
AIC = 689.77   AICc = 690.21   BIC = 709.56
```

4.5.3 Modèle ARMAX

Un ARMAX (*Auto Regressive Moving Average with eXogeneous inputs*), appelé aussi REGARMA, est un modèle de régression linéaire avec une erreur ARMA. Le modèle du niveau du lac Huron (1.1 et 1.2) est un ARX ; le X ou le REG indiquent que la moyenne dépend de variables explicatives exogènes. Le modèle de régression linéaire (3.1, chap. 3) devient, si l'erreur suit un modèle ARMA :

$$y_t = \beta_0 + \beta_1 x_{1,t} + \ldots + \beta_k x_{k,t} + u_t,\ t = 1, 2, \ldots, T, \tag{4.44}$$

$$u_t = \frac{1 + \theta_1 \mathrm{B} + \ldots + \theta_q \mathrm{B}^q}{1 - \phi_1 \mathrm{B} - \ldots - \phi_p \mathrm{B}^p} z_t, \tag{4.45}$$

où z_t est un bruit blanc de variance σ_z^2. Il est clair qu'on est en présence d'un modèle linéaire de $\mathsf{E}(y_t)$, mais avec des erreurs corrélées. Son estimation est un problème de moindres carrés généralisés, évoqué au chapitre 3, expression (3.7). Nous appellerons u_t, *erreur structurelle* et z_t, *innovation*. Le problème est d'identifier le modèle de l'erreur u_t. On procède en deux temps : (1) on commence par faire une régression MCO de y_t sur les variables explicatives, puis on identifie le modèle de u_t sur le résidu $\widehat{u}_t$, ce qui donne une expression de Ω dans (3.7) dépendant de paramètres ; (2) une fois un modèle ARMA satisfaisant obtenu pour le résidu, on estime simultanément la moyenne et la structure de covariance de l'erreur structurelle, à savoir les paramètres de Ω.
On teste ensuite la blancheur de l'innovation $\widehat{z}_t$; éventuellement on corrige ou on simplifie le modèle d'après l'examen des t-statistiques de tous les paramètres. On conclut toujours par un test de blancheur sur $\widehat{z}_t$. Les estimations des β, notées $\widehat{\beta}$ ci-dessous, sont souvent assez proches des estimations obtenues par MCO, mais leurs significativités peuvent sensiblement changer. La prévision de y_{t+1} à partir de $y_1, \ldots, y_t$ et des variables explicatives $(x_{1,1}, \ldots, x_{k,1}), \ldots, (x_{1,t+1}, \ldots, x_{k,t+1})$ (éventuellement on fait une prédiction des explicatives en $t+1$) est donnée par :

$$\widehat{y}_t(1) = \widehat{\beta}_0 + \widehat{\beta}_1 x_{1,t+1} + \ldots + \widehat{\beta}_k x_{k,t+1} + \widehat{u}_{t+1}.$$

C'est la somme de la prédiction de la moyenne et de la prédiction de l'erreur.
Nous rencontrerons d'autres modèles de ce type. Par exemple, au chapitre 11, nous modéliserons la log-collecte de lait par un ARMAX (voir l'équation 11.2, p. 227).

ARMAX et MCG

Un certain nombre de fonctions de R permettent d'ajuster des modèles linéaires par MCG quand l'erreur a une dynamique : `gls()` de **nlme**, `lm()` de **dynlm** notamment. Mais elles offrent moins de modèles pour cette dynamique que `Arima()`, que nous utiliserons. De plus, en séries temporelles on veut non seulement estimer correctement la dépendance de la série par rapport à la série explicative, mais également prédire correctement la série à un certain horizon. Or cet aspect de prédiction est généralement ignoré par les transpositions dans les logiciels des moindres carrés généralisés non orientées vers les séries temporelles.

Exemple 4.8 Pour comprendre cette variété de traitements numériques, considérons le modèle du niveau du lac Huron (1.1, 1.2) qu'on rappelle ici :

$$y_t = \beta_0 + \beta_1 x_t + u_t, \qquad u_t = \phi u_{t-1} + z_t \;\; t = 1, \ldots, T. \tag{4.46}$$

Présentons l'estimation de ce modèle telle que la réalisent les fonctions `Arima()`, `gls()` et `dynlm()`. Pour chaque fonction, nous testons la blancheur des résidus fournis par `residuals()`.

- Dans la fonction `Arima()`, `order=` décrit la dynamique de l'erreur et les variables explicatives sont indiquées par `xreg=`.

```
> temps=time(LakeHuron)
> mod1.lac=Arima(LakeHuron,order=c(1,0,0),xreg=temps,method='ML')
> (cf.arima=mod1.lac$coef)

        ar1   intercept        temps
 0.78347144 618.29557860  -0.02038543

> ychap.arima=fitted(mod1.lac)
> resi.arima=residuals(mod1.lac)
> nret=c(3,6,9,12)
> bl1=Box.test.2(resi.arima,nlag=nret,type="Ljung-Box",decim=4,fitdf=2)
```

► La fonction `gls()` estime des modèles linéaires par MCG et n'est pas tournée vers les séries temporelles ; on doit introduire une variable qui donne la chronologie (variable `tu`). Après quoi, on peut décrire la régression et le type de corrélation retenu :

```
> require(nlme)
> tu = 1:length(temps)
> mod2.lac=gls(LakeHuron~temps,correlation=corAR1(form=~tu),method="ML")
> (cf.gls = mod2.lac$coef)

 (Intercept)        temps
618.29378881  -0.02038447

> ychap.gls = fitted(mod2.lac)
> resi.gls = residuals(mod2.lac)
> bl2=Box.test.2(resi.gls,nlag=nret,type="Ljung-Box",decim=4,fitdf=2)
```

Nous pouvons constater que `Arima()` et `gls()` donnent les mêmes estimations de la régression sur le temps, mais `gls()` ne fournit pas le paramètre d'autorégression.

► Examinons maintenant le fonctionnement de `dynlm()`. L'expression (4.46) s'écrit également :

$$y_t = \beta_0 + \beta_1 x_t + \frac{1}{1-\phi \mathrm{B}} z_t.$$

Multipliant des deux côtés par $1 - \phi\mathrm{B}$, nous obtenons :

$$y_t - \phi y_{t-1} = (1-\phi)\beta_0 + \beta_1(x_t - \phi x_{t-1}) + z_t,$$

or dans cet exemple, $x_t = 1875 + (t-1)$ et les calculs se simplifient. On obtient :

$$y_t = c_0 + c_1\, t + c_2\, y_{t-1} + z_t \tag{4.47}$$

où $c_0 = \beta_0(1-\phi) + \beta_1\,(1875(1-\phi) + 2\phi - 1)$, $c_1 = \beta_1(1-\phi)$ et $c_2 = \phi$.
Ce calcul est fastidieux et, heureusement, rarement nécessaire ; par contre on doit en retenir que s'il n'y a pas de composante MA dans l'erreur, un modèle ARMAX est équivalent à un modèle où les régresseurs contiennent des valeurs retardées des variables explicatives et de la variable dépendante. `dynlm()` peut estimer un tel modèle, mais pas un modèle où l'erreur contient une composante MA. Estimons maintenant le modèle du lac Huron par `dynlm()` :

```
> require(dynlm)
> mod3.lac=dynlm(LakeHuron~1+temps+L(LakeHuron,1))
> cf.dynlm=mod3.lac$coef
> ychap.dynlm=fitted(mod3.lac)
> resi.dynlm=residuals(mod3.lac)
> bl3=Box.test.2(resi.dynlm,nlag=nret,type="Ljung-Box",decim=4,fitdf=2)
```

Enfin nous reparamétrons l'estimation par `Arima()` pour la comparer à celle fournie par `dynlm()`.

```
> cfa=cf.arima
> c2=cfa[1]; c1=cfa[3]*(1-cfa[1])
> c0=cfa[2]*(1-cfa[1])+cfa[3]*(1875-1-cfa[1]*1875+2*cfa[1])
> coefx=round(cbind(as.matrix(cf.dynlm),c(c0,c1,c2)),digits=5)
> colnames(coefx)=c('dynlm','Arima')
> rownames(coefx)=c('Intercept','temps','phi')
```

On peut constater (tableau 4.1) que les estimations de (4.47), qu'elles soient directement fournies par `dynlm()`, ou obtenues par transformation de celles de (4.46) fournies par `Arima()`, sont proches mais ne coïncident pas, notamment parce que les méthodes diffèrent. Le choix de `'CSS'` comme méthode dans `Arima()` à la place de `'ML'` n'améliore pas la situation.

Tableau 4.1 – Estimation du modèle avec variable dépendante retardée, (1) directe par `dynlm()` et (2) via `Arima()`.

	dynlm	Arima
Intercept	127.6481	125.5908
temps	-0.0038	-0.0044
phi	0.7922	0.7835

► Nous voyons dans le tableau 4.2 des cinq premiers résidus que `Arima()` et `dynlm()` donnent des résidus très proches et qui sont bien des innovations (cf. tableau 4.3), alors que `gls()` donne des résidus, écarts à la moyenne estimée.

Tableau 4.2 – Résidus, suivant les fonctions utilisées.

	t	resi.arima	resi.gls	resi.dynlm
1	1.0000	0.1908	0.3071	
2	2.0000	1.5669	1.8075	1.5893
3	3.0000	-0.4782	0.9379	-0.4693
4	4.0000	0.0535	0.7882	0.0696
5	5.0000	-0.8189	-0.2014	-0.8020

Tableau 4.3 – Test de blancheur des résidus, suivant les fonctions utilisées.

	retard	Arima	gls	dynlm
1	3.0000	0.0562	0.0000	0.0664
2	6.0000	0.2344	0.0000	0.2633
3	9.0000	0.2547	0.0000	0.2805
4	12.0000	0.3063	0.0000	0.3252

Dans ce travail, nous utilisons principalement `Arima()` ; l'aspect MCG est évidemment présent, intégré à la méthode d'estimation, mais nous n'avons pas besoin de l'expliciter davantage.

4.6 Construction d'un ARMA ou d'un SARMA

On dispose d'une trajectoire $y_1, \ldots, y_T$ d'une série y_t, éventuellement obtenue après transformation d'une série initiale par passage en log... ; jugeant que cette série est stationnaire, on veut lui ajuster un modèle ARMA(p, q) ou, si elle présente une saisonnalité, un SARMA$(p, q)(P, Q)_s$.
La première étape consiste à choisir les ordres p, q et éventuellement, P, Q. C'est ce qu'on appelle l'*étape d'identification*. Ensuite, *étape d'estimation*, il faut estimer le modèle pour confirmer ces choix et finir la modélisation. Le choix de ces ordres est rarement unique. Pour chaque choix, on doit estimer le modèle correspondant et en tester la qualité. Enfin, si le modèle retenu est satisfaisant, on peut l'utiliser pour prédire la série, *étape de prédiction*, sinon, on change les ordres et on recommence l'estimation. La première qualité du modèle est la blancheur du résidu obtenu ; si ce n'est pas le cas, c'est que le modèle n'a pas capté toute la dynamique du phénomène et il faut choisir d'autres ordres.

4.6.1 Identification d'un ARMA

On examine d'abord le chronogramme et l'ACF de la série. Supposons que :

1. le chronogramme ne révèle pas de tendance, d'aspect saisonnier ou d'hétéroscédasticité marqués ;
2. l'ACF décroît exponentiellement vers 0.

Alors on peut penser que la série est stationnaire et qu'il n'y a pas à choisir (P, Q). Donc il s'agit d'identifier un ARMA, c'est-à-dire de trouver des paramètres p et q raisonnables, estimer le modèle correspondant et le valider en examinant la blancheur du résidu.
Nous sommes maintenant dans la situation où, directement ou après transformation, on dispose d'une série stationnaire pour laquelle on veut identifier un modèle ARMA. On commence par examiner l'ACF et la PACF empiriques de la série. On a

vu que la PACF d'un AR(p) est nulle à partir de l'ordre $p+1$ et l'ACF d'un MA(q) est nulle à partir de l'ordre $q+1$. D'autre part, l'ACF a une décroissance exponentielle pour un AR(p), la PACF a une décroissance exponentielle pour un MA(q) ; enfin ces deux fonctions ont une décroissance exponentielle pour un ARMA(p, q) dont les deux ordres sont non nuls. Ces comportements sont récapitulés dans le tableau 4.4 où : Nul(q) signifie : nul à partir du décalage $q+1$, Exp : décroissance exponentielle, et 0 : fonction nulle à tous les décalages.

Tableau 4.4 – Comportements des fonctions d'autocorrélation suivant le modèle.

Fonction	MA(q)	AR(p)	ARMA(p,q)	bruit blanc
ACF	Nul(q)	Exp	Exp	0
PACF	Exp	Nul(p)	Exp	0

Il est donc assez facile de reconnaître un processus purement AR sur la PACF ou un processus purement MA sur l'ACF d'une série. Une fois l'ordre choisi, on estime le modèle, on teste la blancheur du résidu et, si elle est acceptée, on essaie de simplifier le modèle en examinant la significativité des estimateurs.
Si l'on rejette la blancheur du résidu, il faut reprendre l'identification.

Remarques (Considérations pratiques pour la modélisation ARMA)

1. *Aller du simple au compliqué.* On commence par repérer sur les fonctions d'autocorrélation de la série l'aspect dominant : MA ou bien AR purs. On choisit l'ordre correspondant (q ou p) et on modélise ainsi la série. Il faut examiner ensuite l'ACF et la PACF du résidu de l'ajustement à la lumière du tableau 4.4. Cet examen suggère habituellement les corrections d'ordres à apporter. Modifier les ordres p et q suivant ces suggestions et recommencer l'estimation et l'examen des résidus de cette estimation. C'est une démarche assez proche de celle qui nous a permis d'introduire les SARMA (section 4.5.2).
   Inconvénient de cette approche : on peut passer à côté d'un modèle intéressant. Il faut donc parfois aller également du compliqué au simple.
2. *Simplifier.* On commence par un modèle comportant (relativement) beaucoup de paramètres. Si on accepte la blancheur de son résidu, on essaie de le simplifier suivant les mêmes étapes qu'en régression linéaire, par examen des t-statistiques des coefficients.
   Inconvénient de cette approche : quand le modèle contient de nombreux retards, que ce soit sur la variable à modéliser ou sur les erreurs, il y a risque de colinéarité et les estimations des paramètres restent imprécises. On peut alors conclure que des paramètres sont non significatifs alors qu'ils le deviennent en enlevant certains retards. Il est donc important d'examiner la matrice des corrélations des estimateurs des paramètres (des coefficients supérieurs à 0.9 indiquent souvent une mauvaise spécification). `Arima()` donne en sortie la matrice des covariances des estimateurs et `cor.arma()`, de **caschrono**, en déduit cette matrice de corrélation.

3. Les logiciels qui ajustent un modèle ARMA d'ordres p et q donnés à une série supposée stationnaire, fournissent une représentation inversible : les racines de $\Phi(B) = 0$ et $\Theta(B) = 0$ sont en module strictement supérieures à 1. Quand on essaie d'ajuster un modèle ARMA à une série non stationnaire, on obtient souvent un message d'avertissement ou d'erreur (*à lire attentivement*) car, dans un tel cas, les procédures numériques d'optimisation employées pour l'estimation ne convergent pas ou convergent mal.

4. Les significativités données par les méthodes d'estimation des ARMA sont approximatives, au contraire de ce qui se passe en régression linéaire avec des erreurs normalement distribuées.

5. *Parcimonie.* On ne doit pas accumuler des paramètres. C'est un principe général en statistique. Dans une modélisation ARMA, il ne faut donc pas chercher à ajouter des paramètres pour capter la moindre trace d'autocorrélation dans les résidus d'un ajustement antérieur. La factorisation dans les modèles SARMA va dans le sens de cette parcimonie.

6. *Surajustement.* Si on laisse des paramètres non significatifs dans une modélisation, on dit qu'il y a surajustement (*overfitting*). Conséquence en général : des intervalles de prédiction trop larges.

7. *Choix entre plusieurs modèles.* On utilise, comme en régression (section 3.3) un critère d'information. Supposons qu'on ait ajusté un ARMA(p, q) de bruit blanc $\mathcal{N}(0, \sigma_z^2)$. L'AIC vaut alors :

$$AIC = T \ln(\widehat{\sigma}_z^2) + 2(p + q)$$

où $\widehat{\sigma}_z^2 = \frac{1}{T} \sum \widehat{z}_t^2$ est l'estimation MV de σ_Z^2. Le SBC vaut :

$$SBC = T \ln(\widehat{\sigma}_z^2) + (p + q) \ln(T).$$

Une fois choisie une forme de critère, on retient le modèle pour lequel le critère prend la valeur minimum.

8. Si on doit utiliser le modèle estimé pour prédire la série, il est recommandé de n'utiliser qu'une partie de la série pour estimer le modèle, de façon à pouvoir comparer ensuite, pour un même intervalle de temps, réalisations et prévisions.

9. *Modélisation automatique.* Il existe de nombreuses procédures d'identification automatique de modèles, la méthode MINIC présentée ci-dessous en est un exemple. Elles peuvent être longues et passer à côté d'un modèle manifeste. Aussi, tant qu'on n'est pas familier d'une telle procédure, il est préférable d'identifier d'abord en s'aidant de l'examen du graphe de l'ACF et de la PACF. On peut ensuite comparer ce qu'on a obtenu à ce qu'obtient la procédure automatique. Dans R, les fonctions `auto.arima()` de **forecast**, `armasubsets()` de **TSA** et `BICqLL` de **FitAR** notamment effectuent des choix de meilleurs modèles parmi les ARIMA suivant différents critères (cf. chap. 10).

10. Toutes les séries ne se prêtent pas à une modélisation ARMA et une modélisation ARMA ne permet pas certaines opérations telles que la mise en évidence d'une composante saisonnière. Ce n'est donc pas la panacée.
11. *Prévision sans estimation.* Si l'on doit prédire beaucoup de séries en peu de temps, on utilise des méthodes de prévision qui choisissent automatiquement un modèle dans une certaine classe, sur un critère de qualité de la prévision à un certain horizon. Le chapitre 6, consacré au lissage exponentiel, aborde cette question. L'étude du trafic à l'aéroport de Toulouse-Blagnac y a recours (cf. section 8.5).

Les cas traités dans les chapitres suivants utilisent souvent les techniques et stratégies énumérées ci-dessus.

4.6.2 La méthode MINIC

MINIC est un acronyme pour *Minimum Information Criterion*. Cette méthode aide à découvrir les ordres p et q pour une série stationnaire susceptible de recevoir une modélisation ARMA.
On dispose d'une série $y_t, t = 1, \ldots, n$, centrée, observation d'une série de moyenne nulle, *stationnaire* et *inversible* suivant un modèle ARMA, d'ordres p et q inconnus :

$$y_t = \frac{1 + \theta_1 \mathrm{B} + \ldots + \theta_q \mathrm{B}^q}{1 - \phi_1 \mathrm{B} - \ldots - \phi_p \mathrm{B}^p} z_t \tag{4.48}$$

où $z_t \sim \mathrm{BBN}(0, \sigma_z^2)$.
On a noté qu'un modèle ARMA inversible peut être représenté comme un AR(∞) ; en choisissant un ordre d'autorégression suffisamment grand, on peut approcher de façon satisfaisante le modèle ARMA de la série par un modèle AR d'ordre fini. Le résidu de l'ajustement de la série à un tel modèle est alors proche du bruit blanc sous-jacent à la série, même si les paramètres trop nombreux sont mal estimés. Par ailleurs, l'ajustement d'un modèle autorégressif à une série peut se faire rapidement par différentes méthodes, MCO et Yule-Walker notamment. On obtient ainsi un substitut de résidu, proche du résidu qu'on aurait obtenu en régressant avec le modèle correct. Une fois un tel résidu disponible, il est facile d'ajuster un modèle ARMA en considérant les résidus retardés comme des covariables.
Par exemple, supposons qu'on ait obtenu des résidus $\widetilde{z}_t$ après une régression AR d'ordre élevé. On veut ajuster un ARMA$(2, 3)$ à la série. On peut effectuer l'ajustement *linéaire* donc rapide :

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \theta_1 \widetilde{z}_{t-1} + \theta_2 \widetilde{z}_{t-2} + \theta_3 \widetilde{z}_{t-3} + z_t, \ t = 4, \ldots, T,$$

pour obtenir des estimations des ϕ et θ.
On voit que les ajustements d'ARMA de différents ordres p et q sont ainsi facilement effectués ; ils doivent ensuite être comparés. Comme ces modèles ne sont pas

emboîtés, on doit choisir d'après un critère d'information. On retient le modèle correspondant au minimum du critère, par exemple le SBC.
On retrouve l'approximation d'un ARMA par un AR assez long au chapitre 5 dans le test augmenté de Dickey-Fuller. La fonction `armaselect()` de **caschrono** effectue cette sélection. L'utilisateur donne la série, il peut choisir des valeurs maximum pour p et q (15 par défaut) et le nombre de modèles de plus faible SBC affichés (10 par défaut). La fonction renvoie les ordres p et q de ces meilleurs modèles.

Exemples
Utilisons cette fonction pour identifier le modèle de la série `yc`, simulée suivant

$$y_t = -10 + \frac{1 - 0.3\mathrm{B} + 0.6\mathrm{B}^2}{1 + 0.8\mathrm{B}} z_t, \qquad z_t \sim \mathrm{BBN}(0, 1.5).$$

```
> armaselect(yc,nbmod=5)

     p q      sbc
[1,] 4 0 107.0094
[2,] 4 1 110.7486
[3,] 5 0 111.8614
[4,] 1 2 113.1092
[5,] 5 1 115.5889
```

Le critère retient un ARMA$(1, 2)$, modèle effectivement simulé.

Identifions maintenant le modèle de `yd`, avec saisonnalité, obéissant à (4.43). (Nous savons que $p = 5$ et $q = 2$, mais voyons ce que nous suggère cette méthode.)

```
> armaselect(yd,nbmod=5)

     p q      sbc
[1,] 5 2 139.1428
[2,] 5 1 139.9511
[3,] 6 0 141.4269
[4,] 9 0 142.4936
[5,] 7 0 142.6165
```

Le modèle correct arrive en tête. Evidemment, les factorisations saisonnières ne sont pas détectées par cette approche.

4.7 Exercices

Dans un exercice faisant appel à la simulation, le résultat dépend évidemment de la graine retenue. Il est fructueux de faire l'exercice en changeant plusieurs fois de graine et d'évaluer la stabilité des résultats qu'on obtient.

Exercice 4.5
Ajuster un MA(1) à `y1` (simulé suivant 4.7a), c'est-à-dire un modèle incorrect. Effectuer un test montrant que ce modèle ne convient pas.

Exercice 4.6 (Modèle MA(2))
On considère le modèle MA(2) suivant :

$$y_t = (1 - 0.3\mathsf{B} + 0.6\mathsf{B}^2)z_t \qquad z_t \sim \mathrm{BBN}(0, 1.5). \tag{4.49}$$

1. Calculer les ACF et PACF théoriques de cette série par `ARMAacf()`.
2. Simuler une trajectoire de 200 observations de (4.49) par `arima.sim()`.
3. Par `acf()` appliquée à la trajectoire simulée, calculer des versions empiriques de l'ACF et de la PACF.
4. Comparer graphiquement les versions théorique et empirique de chaque fonction.

Exercice 4.7 (Modèle AR(1))
On considère le modèle AR(1) :

$$y_t = -10 + \frac{1}{1 + 0.8\mathsf{B}} z_t \qquad z_t \sim \mathrm{BBN}(0, 1.5). \tag{4.50}$$

Répondre pour ce modèle aux questions de l'exercice précédent.

Exercice 4.8
Simuler une trajectoire de 200 observations du modèle ARMA$(1, 2)$ combinant les composantes autorégressive et moyenne mobile des modèles précédents (4.50 et 4.49) et comparer les ACF et PACF théoriques et empiriques.

Exercice 4.9
La fonction `armasubsets()` de **TSA** aide à l'identification des modèles ARMA suivant le même principe que `armaselect()` mais reconnaît les trous (les plages de coefficients nuls) dans les ordres de régression. Utiliser `armasubsets()` pour identifier le modèle de `yd` simulé suivant (4.43).

Chapitre 5

Séries temporelles non stationnaires

Nous présentons à travers quelques exemples différents aspects de la non-stationnarité. Une série dont l'évolution autour d'une fonction déterministe du temps est stationnaire est dite *stationnaire à une tendance près* (*trend stationary*). L'estimation de cette tendance et l'identification de l'évolution autour d'elle sont deux questions qu'on peut dissocier en première approche. Mais d'autres séries ne se prêtent pas à une telle modélisation ; on n'en obtient une série stationnaire qu'après différenciation : elles sont dites *stationnaires en différence* (*difference stationary*). Les modèles ARIMA et SARIMA sont des modèles de séries stationnaires en différence qui comportent un trend stochastique (et non déterministe) ou une saisonnalité stochastique (et non déterministe).

Nous examinerons au cours de ce chapitre des exemples simples de séries ayant un trend stochastique comme la marche aléatoire. Avec l'aide de ces exemples nous envisagerons les tests de racine unité. Ils s'intéressent à la question : étant donné une série, est-elle mieux modélisée par un ARIMA, c'est-à-dire un ARMA avec trend stochastique, hypothèse nulle du test, que par un ARMA, hypothèse alternative ? Ensuite nous présentons un test où les rôles sont inversés. L'hypothèse nulle y est - la série possède une tendance déterministe - et l'alternative est - la série possède une tendance stochastique. L'examen des résultats de ces tests est délicat : il faut toujours s'appuyer sur le chronogramme de la série pour les lire correctement. Nous concluons ce chapitre sur des exemples de significativité illusoire ou de régression fallacieuse : quand on régresse une série non stationnaire sur une autre série non stationnaire, il arrive que les résultats paraissent très significatifs alors qu'ils n'ont pas de sens.

5.1 Séries intégrées - Modèles ARIMA et SARIMA

Les processus ARIMA et leur version saisonnière SARIMA sont des processus non stationnaires qui reviennent, après différenciation simple ou saisonnière, à des processus ARMA ou SARMA.
Une série est dite *intégrée* d'ordre d, noté I(d) s'il faut la différencier d fois pour obtenir une série stationnaire. Ainsi, un processus ARIMA(p, d, q) est un processus dont la différence d'ordre d est un ARMA(p, q). Une série I(0) est stationnaire. Un ARMA(p, q) est un ARIMA$(p, 0, q)$.
Par exemple, la série y_t vérifiant :

$$(1 - \mathrm{B})y_t = c + \frac{1 + \theta_1 \mathrm{B} + \theta_2 \mathrm{B}^2}{1 - \phi_1 \mathrm{B} - \phi_3 \mathrm{B}^3} z_t$$

est un ARIMA$(3, 1, 2)$, la constante c est la dérive (*drift*). (Dans ce chapitre, z_t désigne un bruit blanc.)
La série y_t obéissant à :

$$(1 - \mathrm{B})^2 y_t = c + (1 + \theta_1 \mathrm{B}) z_t$$

est un ARIMA$(0, 2, 1)$.
Un ARIMA(p, d, q), $d > 0$, est autorégressif mais pas stationnaire, car 1 est d fois racine de son polynôme d'autorégression. Ainsi l'ARIMA$(3, 1, 2)$ ci-dessus peut s'écrire

$$(1 - (1 + \phi_1)\mathrm{B} + \phi_1 \mathrm{B}^2 - \phi_3 \mathrm{B}^3 + \phi_3 \mathrm{B}^4) y_t = b + (1 + \theta_1 \mathrm{B} + \theta_2 \mathrm{B}^2) z_t$$

avec $b = c(1 - \phi_1 - \phi_3)$; il y a bien une partie autorégressive, mais le polynôme d'autorégression a 1 comme racine et la série est non stationnaire. Alors on n'écrit pas que c'est un AR$(4, 2)$ mais un ARIMA$(3, 1, 2)$. La série différenciée $(1-\mathrm{B})y_t$ est la série des accroissements de y_t, alors que la série différenciée deux fois $(1 - \mathrm{B})^2 y_t$ est la série des *accroissements des accroissements*.
Considérons maintenant la *version saisonnière de l'intégration*. Un processus y_t est un SARIMA$(p, d, q)(P, D, Q)_s$ s'il obéit à :

$$\begin{aligned} & (1 - \mathrm{B})^d (1 - \mathrm{B}^s)^D y_t \\ = \; & c + \frac{(1 + \theta_1 \mathrm{B} + \cdots + \theta_q \mathrm{B}^q)(1 + \Theta_1 \mathrm{B}^s + \cdots + \Theta_Q \mathrm{B}^{sQ})}{(1 - \phi_1 \mathrm{B} - \cdots \phi_p \mathrm{B}^p)(1 - \Phi_1 \mathrm{B}^s - \cdots - \Phi_P \mathrm{B}^{sP})} z_t \end{aligned} \qquad (5.1)$$

ou

$$(1 - \mathrm{B})^d (1 - \mathrm{B}^s)^D y_t = c + \frac{\Theta(\mathrm{B})\Theta_s(\mathrm{B}^s)}{\Phi(\mathrm{B})\Phi_s(\mathrm{B}^s)} z_t$$

avec

$$\Theta_s(\mathrm{B}^s) = 1 + \Theta_1 \mathrm{B}^s + \cdots + \Theta_Q \mathrm{B}^{sQ}$$

et

$$\Phi_s(\mathrm{B}^s) = 1 - \Phi_1 \mathrm{B}^s - \cdots - \Phi_P \mathrm{B}^{sP}.$$

Par exemple un SARIMA$(0,1,1)(1,1,0)_{12}$ obéit à :

$$(1-\mathrm{B})(1-\mathrm{B}^{12})y_t = c + \frac{1+\theta_1\mathrm{B}}{1-\Phi_1\mathrm{B}^{12}}z_t \tag{5.2}$$

où $|\Phi_1| < 1$. Il faut donc différencier la série, simplement une fois et saisonnièrement une fois avec une saisonnalité de 12 pour obtenir une série stationnaire. Autre exemple, y_t ci-dessous est un SARIMA$(1,0,1)(2,1,0)_4$:

$$(1-\mathrm{B}^4)y_t = c + \frac{1+\theta_1\mathrm{B}}{(1-\phi_1\mathrm{B})(1-\Phi_1\mathrm{B}^4-\Phi_2\mathrm{B}^8)}z_t. \tag{5.3}$$

La série y_t est intégrée saisonnièrement d'ordre 1. La constante c est, en quelque sorte, une dérive saisonnière. Pour que la série différenciée saisonnièrement une fois soit bien stationnaire, il faut que les racines des polynômes $1-\phi_1 z = 0$ et $1-\Phi_1 z^4 - \Phi_2 z^8 = 0$ soient strictement supérieures à 1 en module.

Estimation. Si l'on veut ajuster par maximum de vraisemblance ce modèle à une série `y1` chargée dans R, on utilisera le code :

```
(mod1=Arima(y1,order=c(1,0,1),seasonal=list(order=c(2,1,0),period=4),
            include.drift=TRUE))
```

Mais R comprend le modèle SARIMA$(p,0,q)(P,1,0)_4$ avec dérive comme un modèle de Y_t de la forme :

$$Y_t = a + b\,t + e_t, \tag{5.4}$$

où après différenciation à l'ordre 4, $(1-\mathrm{B}^4)e_t$ est SARMA$(p,q)(P,0)_4$ centré. Cette différenciation donne :

$$(1-\mathrm{B}^4)Y_t = 4\,b + (1-\mathrm{B}^4)e_t,$$

$4\,b$ est la moyenne de la série différenciée à l'ordre 4 et R estime b et non $4\,b$, et l'appelle *drift*. Nous vérifierons cette assertion sur un modèle du trafic passager (section 8.6).

Exemples de séries intégrées

Sans autre précision, les différenciations évoquées dans ces exemples sont simples et non saisonnières.

Exemple 5.1 (Marche aléatoire) Examinons le modèle

$$y_t = y_{t-1} + z_t.$$

y_t est une marche aléatoire. En exprimant y_{t-1} en fonction de y_{t-2},... et d'une valeur initiale y_0 on obtient :

$$y_t = y_0 + z_1 + z_2 + \cdots + z_t.$$

Une série obéissant à une marche aléatoire prend, à deux dates consécutives, des valeurs proches et la variation par rapport à la date précédente est indépendante du passé. La figure 5.4 présente un bruit blanc et la marche aléatoire qui en est obtenue par intégration.

Exemple 5.2 (Marche aléatoire avec dérive) Considérons maintenant :

$$y_t = \delta + y_{t-1} + z_t. \tag{5.5}$$

On dit que y_t est une *marche aléatoire avec dérive* δ. En exprimant y_{t-1} en fonction de y_{t-2},... on obtient :

$$y_t = \delta\, t + y_0 + z_1 + z_2 + \cdots + z_t. \tag{5.6}$$

Le graphique de y_t en fonction du temps est donc celui d'une droite à laquelle est superposée une marche aléatoire.
On obtient facilement les moments d'ordre 1 et 2 d'une marche aléatoire :

$$\mathsf{E}(y_t) = y_0, \qquad \mathsf{var}(y_t) = t\,\sigma_z^2, \qquad \mathsf{cov}(y_t, y_{t+k}) = t\,\sigma_z^2,\ (k > 0)$$

Pour une marche aléatoire avec dérive : $\mathsf{E}(y_t) = y_0 + \delta\, t$ et les autres propriétés de la marche aléatoire sans dérive restent valables.
La fonction `rwf()` de **forecast** effectue la prévision d'une telle marche aléatoire. Le lissage exponentiel simple est un modèle de marche aléatoire bruitée (cf. section 6.1.1).

Exemple 5.3 (ARIMA(1, 1, 1)) Considérons :

$$y_t = y_{t-1} + u_t \text{ avec } \ u_t = \frac{1 + 0.8\mathrm{B}}{1 - 0.4\mathrm{B}} z_t, \tag{5.7}$$

u_t est un ARMA(1, 1) (on a remplacé le bruit blanc de la marche aléatoire par un autre bruit stationnaire). Utilisant l'opérateur retard, on note que y_t obéit à un mécanisme autorégressif-moyenne mobile :

$$y_t = \frac{1 + 0.8\mathrm{B}}{(1 - \mathrm{B})(1 - 0.4\mathrm{B})} z_t = \frac{1 + 0.8\mathrm{B}}{1 - 1.4\mathrm{B} + 0.4\mathrm{B}^2} z_t.$$

Mais y_t n'est pas stationnaire car son polynôme d'autorégression admet 1 comme racine ; mais $\Delta y_t = y_t - y_{t-1} = u_t$ est, lui, stationnaire ; y_t est stationnaire à une différenciation près, y_t est un ARIMA(1, 1, 1). On peut dire qu'une série ARIMA(1, 1, 1) prend, à deux dates consécutives, des valeurs proches et la variation par rapport à la date précédente dépend du passé par un mécanisme stationnaire. Une marche aléatoire est un ARIMA(0, 1, 0). L'ajout d'une dérive dans (5.7) introduit une constante additive au modèle de la série différenciée.

Exemple 5.4 (Série stationnaire à une tendance déterministe près) Examinons le modèle :

$$y_t = a + b\,t + u_t \text{ avec } \; b \neq 0 \text{ et } \; u_t \text{ comme ci-dessus} \tag{5.8}$$

on a $\mathsf{E}(y_t) = a + b\,t$ qui dépend de t donc y_t n'est pas stationnaire et

$$\Delta y_t = b + u_t - u_{t-1} = b + \frac{1 - 0.2\mathrm{B} - 0.8\mathrm{B}^2}{1 - 0.4\mathrm{B}} z_t$$

est stationnaire. En fait, y_t, somme d'une tendance déterministe et d'une erreur stationnaire, est stationnaire à une tendance déterministe près, (*trend stationary*). Notons que Δy_t obéit à un modèle non inversible, situation que les logiciels gèrent mal et qu'il vaut mieux éviter.

Exemple 5.5 (Différenciation saisonnière) Examinons un modèle convenant à un phénomène présentant une saisonnalité, par exemple des données économiques trimestrielles. Considérons d'abord une marche aléatoire qu'on peut qualifier de saisonnière :

$$y_t = c + y_{t-4} + z_t.$$

A un certain trimestre, la série prend une valeur proche de celle du même trimestre l'année précédente, augmentée de c, avec une correction trimestrielle indépendante du passé. Maintenant, faisons dépendre cette correction du passé :

$$y_t = c + y_{t-4} + u_t \text{ avec } \; u_t = \frac{1}{1 - 0.9\mathrm{B}} z_t. \tag{5.9}$$

Ici la correction est autocorrélée avec le trimestre précédent. La différenciation à l'ordre 4 de (5.9) donne une série stationnaire :

$$\Delta_4 y_t = c + (1 - \mathrm{B}^4) y_t = \frac{1}{1 - 0.9\mathrm{B}} z_t$$

y_t, stationnaire en différence (saisonnière), est un SARIMA$(1,0,0)(0,1,0)_4$ et donc $\Delta_4 y_t$ est un AR(1) ; (5.9) présente une saisonnalité stochastique.

Exemple 5.6 (Saisonnalité déterministe) Un modèle saisonnier à saisonnalité déterministe pourrait être :

$$y_t = a + b\cos(2\pi t/4) + c\sin(2\pi t/4) + u_t \text{ avec } \; u_t = \frac{1}{1 - 0.9\mathrm{B}} z_t \tag{5.10}$$

la saisonnalité déterministe y est exprimée par une fonction trigonométrique de période 4 et la série est dite stationnaire à une saisonnalité déterministe près.

Remarques

1. Nous venons d'examiner des modèles qui montrent deux formes de tendance ou de saisonnalité : (1) stochastique pour (5.7) et (5.9), qui s'élimine par différenciation, et (2) déterministe pour (5.8) et (5.10) qui peut se traiter de différentes

façons. Pour (5.8), on peut ajuster une droite et définir une erreur ARMA(1,1). On peut alternativement différencier y_t. Pour capter la tendance dans (5.10), on peut ajuster des fonctions périodiques de période égale à la saisonnalité. Alternativement, on peut différencier saisonnièrement la série. Nous utiliserons ces techniques pour la modélisation de la température à Nottingham Castle (chap. 9) ou de la collecte de lait (chap. 11).

2. Quand une tendance linéaire n'apparaît pas clairement sur un graphique ou que l'équation de la tendance semble changer au cours du temps, il est préférable de différencier. Il se peut qu'observant une série sur une période de temps assez courte, on puisse lui ajuster un modèle à tendance déterministe, mais si l'on sait que la série est fondamentalement non stationnaire et pas seulement stationnaire à une tendance déterministe près, il est préférable de lui ajuster un modèle avec tendance stochastique.

3. Si une série différenciée, série des accroissements, présente encore une tendance, on différencie la série initiale une seconde fois : on travaille ainsi sur la série des accroissements des accroissements.

4. Si l'on veut différencier une série présentant une saisonnalité et une tendance, il est préférable de commencer par différencier saisonnièrement la série et d'examiner la série obtenue avant de décider de différencier également à l'ordre 1. En effet, considérons par exemple le cas d'une série trimestrielle. La différenciation saisonnière se factorise en :

$$1 - \mathrm{B}^4 = (1 - \mathrm{B})(1 + \mathrm{B} + \mathrm{B}^2 + \mathrm{B}^3)$$

et l'on voit qu'elle contient une différenciation simple : souvent la seule différenciation saisonnière suffit.

5. Si $y_t, t = 1, \cdots, T$ est une trajectoire d'une série avec dérive, $y_t = c + y_{t-1} = +u_t$, où u_t est un ARMA, alors la série retournée $w_t = y_{T-t+1}$ admet $-c$ comme dérive et le fonctionnement de la fonction `lag()` induit en erreur sur le signe de la dérive qu'on peut déduire d'un lag plot. C'est pourquoi nous appliquons la fonction `lag.plot()` à la série retournée et non à la série à étudier.

6. Gourieroux & Monfort (1995, chap. 6) ou Box *et al.* (1999, chap. 5) présentent la prévision des séries intégrées par la méthode de Box et Jenkins.

Exercice 5.1 (Différenciation saisonnière)
Vérifier empiriquement l'effet d'une différenciation saisonnière sur (5.10). On pourra définir les séries $\cos(2\pi t/4)$ et $\sin(2\pi t/4)$, $t = 1, \cdots, 48$ et calculer leurs différences saisonnières.
Pour des compléments théoriques on peut consulter Gourieroux & Monfort (1995, chap. 3) qui présentent les propriétés algébriques des filtres de moyenne mobile, appelés aussi *filtres de moyenne glissante* (*running mean*). Ladiray & Quenneville (2001) expliquent en détail l'usage de ces filtres en macro-économétrie.

Exercice 5.2 (Estimation d'un SARIMA avec dérive)
On a indiqué à la section 5.1 comment R estime les modèles intégrés. S'il y avait différenciation aux ordres 1 et 12, il faudrait ainsi introduire le régresseur t^2 (la constante et le régresseur t sont éliminés dans les différenciations) :

$$Y_t = c\, t^2 + e_t.$$

La différenciation aux ordres 1 et 12 donne

$$(1-\mathsf{B})(1-\mathsf{B}^{12})Y_t = 24\, c + (1-\mathsf{B})(1-\mathsf{B}^{12})e_t$$

et R fournit donc une estimation de c et non de $24\, c$.
Vérifier cette assertion en simulant un SARIMA$(1,1,0)(0,1,1)_{12}$ puis en l'estimant.

Exercice 5.3 (Lag plot d'une série avec dérive)
Simuler des séries de 200 points suivant (5.9), avec $\sigma_z^2 = 1$ et les valeurs initiales égales à 0. D'abord avec $c = -.2$ puis $c = .2$. Dessiner les lag plots correspondants jusqu'au retard 4 et observer la dérive sur ces graphes. Commenter.

5.2 Construction d'un modèle SARIMA

On dispose d'une trajectoire $y_1, \cdots, y_T$ d'une série y_t, éventuellement obtenue après transformation d'une série initiale par passage en log,... ; la série n'est pas stationnaire et on veut lui ajuster un modèle ARIMA(p,d,q) ou, si elle présente une saisonnalité, un SARIMA$(p,d,q)(P,D,Q)_s$. Une fois choisis d et D, on est ramené à l'identification d'un ARMA ou d'un SARMA sur la série différenciée. Nous avons discuté intuitivement le choix de d au chapitre précédent et nous verrons dans la section suivante un test de l'hypothèse $d = 1$ contre $d = 0$ qui donne une forme rigoureuse à ce choix. Si un doute persiste, on pousse la modélisation de la série avec et sans différenciation, et on compare les qualités des modèles : critère d'information ou valeur prédictive selon l'objectif.

Séries montrant une saisonnalité. On pense que D est non nul si l'ACF de la série, examinée seulement aux décalages $s, 2s, 3s \ldots$ ne décroît pas exponentiellement. La situation est parallèle au cas $d = 1$ mais avec un pas de temps de s. La série de température à Nottingham Castle et la série y simulée suivant un SARMA$(0,0)(1,0)_{12}$ (chap. 1) présentent chacune une forte autocorrélation empirique aux retards multiples de 12, comme on l'a vu sur leurs lag plots (fig.1.9 et 1.10), mais alors que l'ACF empiririque du SARMA décroît rapidement de 12 en 12, celle de la température reste élevée (fig. 5.1), graphique obtenu grâce aux commandes suivantes :

```
> data(nottem)
> require(caschrono)
> plot2acf(nottem,y,main=c("ACF nottem","ACF SAR"))
```

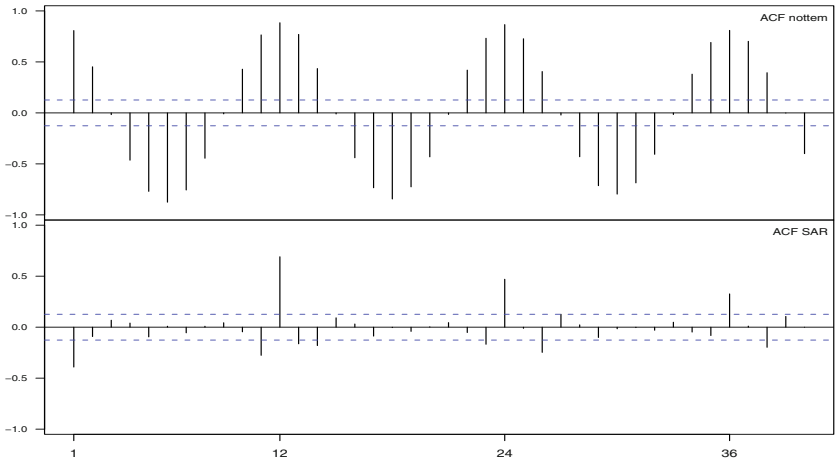

Fig. 5.1 – ACF empirique d'une série non stationnaire (haut) et d'une série stationnaire (bas), avec saisonnalité.

La fonction `auto.arima()` de **forecast** effectue le test de Canova-Hansen. Il teste si une saisonnalité est de nature déterministe ou stochastique. Nous nous limitons ici à une discussion de la saisonnalité.

5.3 Non-stationnarité stochastique ou déterministe

Dans la présentation des modèles autorégressifs nous avons bien distingué le cas où le polynôme d'autorégression a une racine égale à 1 (cas non stationnaire) du cas où toutes les racines sont en module strictement supérieures à 1. Ici nous examinons des tests concernant deux formes de non-stationnarité : la non-stationnarité due à une racine unité dans l'autorégression, et la non-stationnarité due à une tendance déterministe. Il peut être difficile de distinguer ces situations, surtout sur des trajectoires courtes.
Nous allons illustrer les tests de racine unité ou de stationnarité d'après leur programmation dans **urca**, Pfaff (2006). On trouve également ces tests dans d'autres packages comme : **FinTS**, **CADFtest**, **timeSeries** et **tseries**.

5.3.1 Test de non-stationnarité : introduction et pratique

Les tests de racine unité considèrent l'hypothèse nulle : « 1 est racine du polynôme d'autorégression ». Leur utilisation est délicate, aussi est-il indispensable de les conduire parallèlement à un examen du chronogramme. Ici nous présentons la situation et découvrons la marche à suivre sur des exemples et à travers le test ADF (*Augmented Dickey-Fuller*).

Principe du test de Dickey-Fuller. Il concerne les séries autorégressives d'ordre 1.

Cas I : la série ne montre pas de tendance. Elle ne peut donc être ni une marche aléatoire avec dérive (5.6), ni une série avec un trend déterministe comme (5.8). La régression considérée est

$$y_t = \beta_1 + \phi y_{t-1} + z_t,$$

la constante β_1 est là pour capter une moyenne non nulle dans le cas où $\phi \neq 1$, mais si $\phi = 1$, β_1 doit être nulle puisque la série ne montre pas de dérive. On teste $H_0 : \phi = 1$, c'est-à-dire la série est I(1) sans dérive contre $H_1 : |\phi| < 1$ et la série est I(0) avec une moyenne éventuellement non nulle. Cette approche convient aux séries financières sans tendance, comme les taux d'intérêt, les taux de change.
La statistique de test est $T(\widehat{\phi} - 1)$ où $\widehat{\phi}$ est l'estimateur MCO de ϕ. Rappelons que sous l'hypothèse nulle, $\phi = 1$, cette statistique **ne suit pas** une loi standard. On rejette l'hypothèse nulle pour les faibles valeurs de la statistique. On peut également utiliser la statistique de Student habituelle $(\widehat{\phi} - 1)/s_{\widehat{\phi}}$. Bien entendu, elle ne suit pas une loi de Student sous l'hypothèse nulle. Les valeurs critiques sont indiquées dans les sorties de R.

Cas II : la série montre une tendance. Celle-ci peut apparaître si la série est une marche aléatoire avec dérive (5.6) ou si elle est stationnaire à un trend déterministe près comme (5.8) (nous n'envisageons pas le cas, peu courant, où les deux aspects sont présents). La régression considérée est

$$y_t = \beta_1 + \beta_2\, t + \phi y_{t-1} + z_t, \tag{5.11}$$

β_1 et β_2 sont destinés à capter l'éventuelle tendance déterministe si $|\phi| < 1$. Posons $\pi = \phi - 1$. On teste $H_0 : (\phi = 1,\ \beta_2 = 0)$, ou $(\pi = 0,\ \beta_2 = 0)$ c'est-à-dire la série est I(1) avec dérive contre $H_1 : |\phi| < 1$, et la série est I(0) avec une tendance linéaire. Cette approche convient aux séries ayant une tendance, comme le cours d'un titre, le niveau d'un agrégat macroéconomique.

La statistique de test est la statistique de Fisher F, pour tester H_0, calculée comme dans la méthode MCO, mais, sous l'hypothèse nulle, cette statistique ne suit pas une loi de Fisher. Les valeurs critiques sont tabulées dans différents packages, **urca** notamment. On rejette l'hypothèse nulle pour les faibles valeurs de la statistique.

Principe du test de Dickey-Fuller augmenté. (*ADF test*) Il est peu probable qu'un modèle autorégressif d'ordre 1 suffise à décrire la dynamique. Considérons donc un autorégressif d'ordre p, pas nécessairement stationnaire, avec tendance :

$$y_t = \beta_1 + \beta_2\, t + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + z_t. \tag{5.12}$$

Cette série n'est pas stationnaire, à la tendance déterministe près, si le polynôme $1 - \phi_1 \mathrm{B} - \cdots - \phi_p \mathrm{B}^p$ admet la racine 1. Il serait utile de voir apparaître le coefficient

$\rho = \phi_1 + \cdots + \phi_p$ dans (5.12). C'est ce que permet l'écriture ECM (*Error Correction Model*) de (5.12) :

$$y_t = \beta_1 + \beta_2\, t + \rho y_{t-1} + \zeta_1 \triangle y_{t-1} + \cdots + \zeta_{p-1} \triangle y_{t-p+1} + z_t, \tag{5.13}$$

ou, retranchant des deux côtés y_{t-1}, et posant $\pi = \phi_1 + \cdots + \phi_p - 1 = \rho - 1$:

$$\triangle y_t = \beta_1 + \beta_2\, t + \pi y_{t-1} + \zeta_1 \triangle y_{t-1} + \cdots + \zeta_{p-1} \triangle y_{t-p+1} + z_t \tag{5.14}$$

où les ζ s'expriment en fonction des ϕ dans (5.13). On obtient la représentation (5.13) en écrivant $y_{t-p} = y_{t-p+1} - \triangle y_{t-p+1}$, $y_{t-p-1} = y_{t-p+2} - \triangle y_{t-p}, \cdots$. Examinons (5.14), le côté gauche est la variation de la série de $t-1$ à t ; elle est exprimée du côté droit comme une combinaison linéaire d'un trend, de la valeur de la série à la date précédente, des variations antérieures et de l'erreur en t ; la terminologie « modèle de correction d'erreur » est parlante. Si le polynôme d'autorégression de la série a une racine unité, alors le côté gauche de (5.14) qui est la série différenciée est stationnaire, donc le côté droit doit l'être aussi. Si le polynôme d'autorégression n'a pas de racine unité, alors les deux côtés sont stationnaires et la série peut avoir un trend linéaire.

Avant de conduire un test de non-stationnarité, il faut examiner le chronogramme de la série pour voir quelles hypothèses sont envisageables et, comme précédemment, deux cas sont possibles.

Cas I : la série ne montre pas de tendance. On ne doit donc pas mettre de terme en β_2 dans le modèle de y_t et il faut estimer

$$\triangle y_t = \beta_1 + \pi y_{t-1} + \zeta_1 \triangle y_{t-1} + \cdots + \zeta_{p-1} \triangle y_{t-p+1} + z_t \tag{5.15}$$

l'hypothèse à tester est $\mathrm{H}_0 : (\beta_1, \pi) = (0, 0)$, car s'il n'y a pas de tendance, il ne peut y avoir de dérive en cas de non-stationnarité. L'alternative logique est $\mathrm{H}_1 : \pi < 0$ et β_1 quelconque. Le modèle sans dérive ni tendance est :

$$\triangle y_t = \pi y_{t-1} + \zeta_1 \triangle y_{t-1} + \cdots + \zeta_{p-1} \triangle y_{t-p+1} + z_t. \tag{5.16}$$

Cas II : la série montre une tendance. Elle peut correspondre à une série non stationnaire avec dérive, hypothèse nulle, $\mathrm{H}_0 : (\beta_2, \pi) = (0, 0)$, ou à une série stationnaire avec trend déterministe, hypothèse alternative, $\mathrm{H}_1 : \pi < 0$. Il faut estimer (5.14).

Mise en pratique des tests ADF. Il existe plusieurs tests de racine unité. Nous présentons la fonction `ur.df()` de **urca** qui effectue les tests ADF. Dans les sorties de `ur.df()`, les statistiques `tau`... ci-dessous désignent les t-statistiques, de distribution non classique sous l'hypothèse nulle, pour tester $\pi = 0$ dans un des modèles rencontrés plus haut :

- `tau1` : pour le modèle (5.16), `type='none'` ;
- `tau2` : pour le modèle (5.15), `type='drift'` ;
- `tau3` : pour le modèle (5.14), `type='trend'`.

On a indiqué la valeur à donner à l'argument `type` de `ur.df()` pour estimer le modèle correspondant.

Dans les trois modèles, *on rejette l'hypothèse nulle et donc on conclut à la stationnarité pour des valeurs très négatives de la statistique.*

Les valeurs critiques, tirées de Fuller (1996), sont fournies en sortie. La fonction fournit également des statistiques `phi`..., ce sont des statistiques de Fisher, de distribution non classique sous l'hypothèse nulle :

- `phi1` : pour tester $(\beta_1, \pi) = (0, 0)$ dans le modèle (5.15) ;
- `phi2` : pour tester $(\beta_1, \beta_2, \pi) = (0, 0, 0)$ dans le modèle (5.14) ;
- `phi3` : pour tester $(\beta_2, \pi) = (0, 0)$ dans le modèle (5.14).

Dans les trois cas *on rejette l'hypothèse nulle pour de grandes valeurs de la statistique.*

Les valeurs critiques sont fournies dans les sorties. Ces tests sont moins puissants que ceux basés sur `tau` : la probabilité de rejeter la non-stationnarité alors qu'elle est présente est plus forte avec un test sur `phi` qu'avec un test sur `tau`, aussi nous ne considérerons pas leurs résultats.

Exemple 5.1 Etudions la consommation réelle trimestrielle en log au Royaume-Uni, du 4e trimestre 1966 au 2e trimestre 1991 (fig. 5.2), composante `lc` de `Raotbl3` de **urca**.

```
> require(urca)
> data(Raotbl3)
> attach(Raotbl3,warn.conflicts=FALSE)
> plot(lc,type="l",xaxt ="n",xlab="temps",cex=.8)
> axis(1,at=seq(from=2,to=98,by=12),
+      labels=as.character(seq(from=1967,to=1991,by=3)))
```

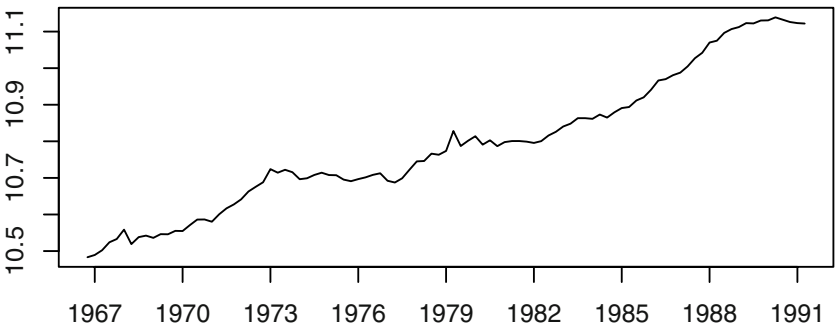

Fig. 5.2 – Log dépense réelle de consommation trimestrielle au Royaume-Uni.

La série (fig. 5.2) a bien une tendance, donc on ajuste un modèle avec tendance (5.14). Pour cela on utilise l'option `type='trend'` dans `ur.df()` mais p est inconnu. Nous commençons par choisir une valeur élevée et la diminuons tant que le coefficient du plus grand retard n'est pas significatif. Commençons par le retard 6, qui correspond à la valeur de $p-1$ dans la régression (5.13) :

```
> lc.df0=ur.df(y=lc,lags=6,type='trend')
```

La commande `str(lc.df0)` nous montre que `lc.df0` est un objet de classe S4 et que les t-statistiques figurent dans `lc.df0@testreg$coefficients`. Le premier retard très significatif est le 3, nous menons donc le test complet avec ce retard :

```
> lc.df1=ur.df(y=lc,lags=3,type='trend')
> summary(lc.df1)

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0447139 -0.0065246  0.0001288  0.0062253  0.0453532

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.7976591  0.3547775   2.248   0.0270 *
z.lag.1     -0.0758706  0.0338880  -2.239   0.0277 *
tt           0.0004915  0.0002159   2.277   0.0252 *
z.diff.lag1 -0.1063957  0.1006744  -1.057   0.2934
z.diff.lag2  0.2011373  0.1012373   1.987   0.0500 .
z.diff.lag3  0.2998586  0.1020548   2.938   0.0042 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01307 on 89 degrees of freedom
Multiple R-squared: 0.1472,        Adjusted R-squared: 0.09924
F-statistic: 3.071 on 5 and 89 DF,  p-value: 0.01325

Value of test-statistic is: -2.2389 3.7382 2.5972

Critical values for test statistics:
      1pct  5pct 10pct
tau3 -4.04 -3.45 -3.15
phi2  6.50  4.88  4.16
phi3  8.73  6.49  5.47
```

Observons la sortie `lc.df1`. La ligne `Value of test-statistic is:` contient trois statistiques `tau3`, `phi2`, `phi3` dont les valeurs critiques sont données dans les lignes qui suivent `Critical values for test statistics:`
Vu le chronogramme, nous n'avons qu'à tester $\pi = 0$, nous nous intéressons donc à `tau3`. Elle prend la valeur -2.2389 qui correspond à une p-value supérieure à 10%,

puisque pour ce niveau la région critique serait $(-\infty, -3.15)$. Nous gardons donc l'hypothèse de racine unité. Nous pensons que la série n'a pas, en plus du trend stochastique, un trend déterministe, donc a priori, $\beta_2 = 0$. Voyons, avec beaucoup de prudence, si par `phi3` nous pouvons effectivement conclure à $\beta_2 = 0$. `phi3` vaut 2.5972, très inférieur au seuil 5.47 qui correspond à un niveau de 10%, donc on accepte l'hypothèse que la série a une racine unité et pas de tendance. (Il peut arriver que les résultats des tests contredisent l'observation des graphiques et, comme on l'a dit et pratiqué, il faut prêter attention avant tout aux chronogrammes.)

Exemple 5.2 (Taux d'intérêt) Considérons la série `i1` de `UKpppuip`, série de taux d'intérêt des bons du Trésor au Royaume-Uni. *Three-month treasury bill rate in the UK.*

```
> data(UKpppuip)
> i1=ts(UKpppuip$i1,start=c(1971,1),frequency=4)
> attach(UKpppuip,warn.conflicts=FALSE)
> plot(i1,type='l',xlab='temps',ylab='taux')
```

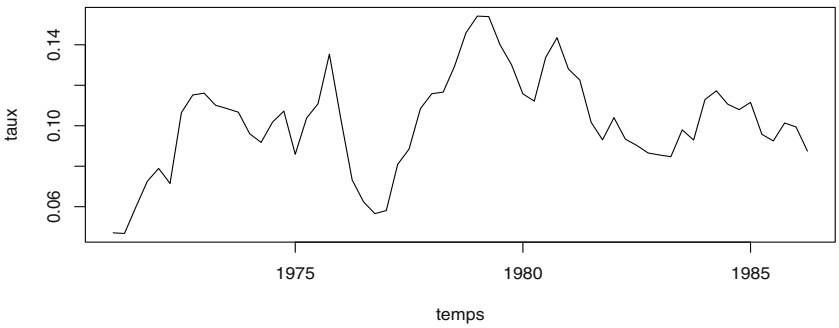

Fig. 5.3 – Taux d'intérêt des bons du Trésor au Royaume-Uni.

La série ne montre pas de tendance ; on peut lui ajuster le modèle (5.15) en précisant l'option `type='drift'`. Si on conclut que $\pi \neq 0$, alors le niveau moyen μ de la série vérifie : $\beta_1 + \pi\mu = 0$. Si on conclut par contre que $\pi = 0$, on devrait conclure également que $\beta_1 = 0$.

```
> i1.df0=ur.df(y=i1,lags=6,type='drift')
> summary(i1.df0)

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift

Call:
lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)
```

```
Residuals:
       Min         1Q     Median         3Q        Max
-0.0324769 -0.0051375 -0.0007515  0.0070319  0.0259021

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.02716    0.01129   2.406   0.0201 *
z.lag.1     -0.26174    0.10656  -2.456   0.0178 *
z.diff.lag1  0.32359    0.13588   2.381   0.0213 *
z.diff.lag2  0.08100    0.14052   0.576   0.5671
z.diff.lag3  0.01478    0.13898   0.106   0.9158
z.diff.lag4  0.04006    0.13374   0.300   0.7658
z.diff.lag5 -0.02015    0.13283  -0.152   0.8800
z.diff.lag6  0.10804    0.13209   0.818   0.4175
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01223 on 47 degrees of freedom
Multiple R-squared: 0.1928,        Adjusted R-squared: 0.07257
F-statistic: 1.604 on 7 and 47 DF,  p-value: 0.1578


Value of test-statistic is: -2.4563 3.0288

Critical values for test statistics:
      1pct  5pct 10pct
tau2 -3.51 -2.89 -2.58
phi1  6.70  4.71  3.86
```

Les retards 2 à 6 ne sont pas significatifs. Une régression jusqu'au retard 2 montre qu'il ne l'est pas non plus. Il suffit donc de régresser jusqu'au retard 1.

```
> i1.df1=ur.df(y=i1,lags=1,type='drift')
> summary(i1.df1)

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift

Call:
lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)

Residuals:
      Min        1Q    Median        3Q       Max
-0.031451 -0.006056 -0.000449  0.007220  0.029350
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.024109   0.007118   3.387  0.00129 **
z.lag.1     -0.229819   0.067644  -3.397  0.00125 **
z.diff.lag   0.268719   0.121901   2.204  0.03155 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01207 on 57 degrees of freedom
Multiple R-squared: 0.1975,        Adjusted R-squared: 0.1693
F-statistic: 7.013 on 2 and 57 DF,  p-value: 0.001893

Value of test-statistic is: -3.3975 5.8255

Critical values for test statistics:
      1pct  5pct 10pct
tau2 -3.51 -2.89 -2.58
phi1  6.70  4.71  3.86
```

Les statistiques fournies sont `tau2` et `phi1` ; `tau2` prend la valeur -3.3975 qui correspond à une p-value entre 1% et 5%, il n'est donc pas évident de conclure si cette série est ou non stationnaire. Examinons alors `phi1` : elle prend la valeur 5.8255, qui correspond encore à une p-value entre 1% et 5%. Il est encore difficile de conclure. Pour clarifier la situation, on peut tenter une modélisation par un ARMA et examiner la somme des coefficients d'autorégression. Il est aussi possible de tester la stationnarité de la série par un test dont l'hypothèse nulle est la stationnarité : c'est le cas du test de KPSS présenté au paragraphe suivant.

Remarques

- Utiliser une autorégression assez longue pour capter l'autocorrélation de la série est une démarche déjà rencontrée dans la méthode MINIC.
- L'ordre p de l'autorégression n'étant pas connu, on commence avec un ordre élevé, qu'on diminue tant que le coefficient d'ordre le plus élevé n'est pas significatif.
- Il se peut qu'une série soit intégrée d'ordre 2, on le voit si, après différenciation à l'ordre 1, on détecte encore une racine unité sur la série différenciée.
- Le test ADF, comme la plupart des tests, est *conservatif*, c'est-à-dire qu'il a tendance à garder l'hypothèse nulle, donc à conclure faussement qu'une série est non stationnaire. C'est pourquoi il est intéressant de disposer de tests où la non-stationnarité est attachée à l'alternative. C'est ce que fait le test de KPSS que nous examinons au prochain paragraphe.

5.3.2 Test de stationnarité à une tendance déterministe près

Nous envisageons ici le test KPSS de Kwiatkowski *et al.* (1992) utilisable sous R grâce à la fonction `ur.kpss()`. Dans ce test, l'hypothèse nulle est - la série est

stationnaire, soit à une tendance près, soit à une moyenne non nulle près - contre l'alternative - la série est non stationnaire en un certain sens. Précisément, le test suppose que la série est la somme d'une marche aléatoire, d'un trend déterministe et d'une erreur stationnaire :

$$y_t = R_t + \beta_1 + \beta_2 t + U_t,$$

où R_t est la marche aléatoire : $R_t = R_{t-1} + z_t$, $z_t, \simeq BNN(0, \sigma_z^2)$, $\beta_1 + \beta_2 t$ une tendance déterministe et d'une erreur stationnaire U_t. Notons qu'il n'est pas évident que y_t obéisse à un modèle ARIMA. Pour tester que la série y_t est stationnaire à une tendance près, l'hypothèse nulle est $\sigma_z^2 = 0$, c'est-à-dire qu'il n'y a pas de composante « marche aléatoire ». La statistique du test KPSS est celle du test du multiplicateur de Lagrange pour tester $\sigma_z^2 = 0$ contre l'alternative $\sigma_z^2 > 0$:

$$\texttt{KPSS} = (T^{-2} \sum_{t=1}^{T} \widehat{S}_t^2)/\widehat{\lambda}^2$$

où $\widehat{S}_t = \sum_{j=1}^{t} \widehat{u}_j$, $\widehat{u}_t$ est le résidu de la régression de y_t sur la composante déterministe supposée et $\widehat{\lambda}^2$ un estimateur convergent de la variance de long terme de u_t basé sur $\widehat{u}_t$. Sous l'hypothèse nulle, la loi de `KPSS` converge vers une loi non standard qui ne dépend pas des valeurs des β, mais seulement de la forme de la tendance qui peut être un niveau ($\beta_2 = 0$) ou une tendance linéaire, $\beta_1 + \beta_2 t$. *On rejette l'hypothèse nulle de stationnarité pour de grandes valeurs de la statistique de test.*

Exemple 5.3 (Séries simulées) Simulons `x` un bruit blanc, donc stationnaire, et formons `y`, la série intégrée de `x` (fig. 5.4)

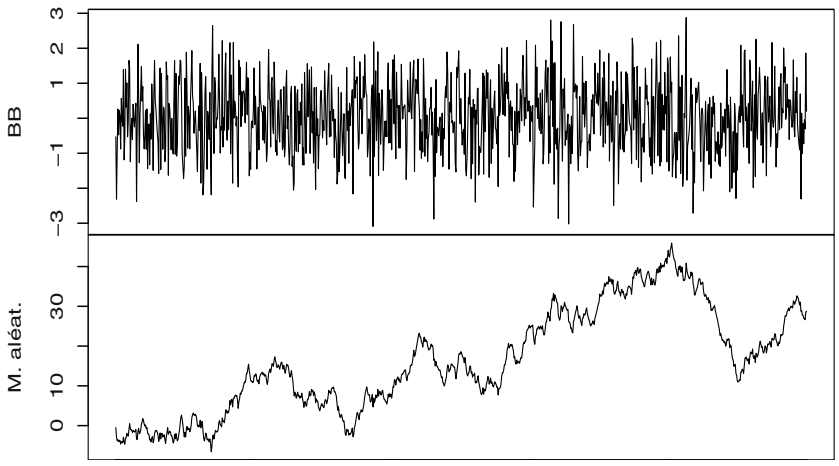

Fig. 5.4 – Bruit blanc et sa série intégrée (marche aléatoire).

grâce aux commandes :

```
> set.seed(231)
> x=rnorm(1000)
> y=cumsum(x)
> xy=ts(cbind(x,y))
> colnames(xy)=c('BB','M. aléat.')
> plot.ts(xy,xlab="temps",main='',
+   oma.multi=c(0,0,.2,0),mar.multi=c(0,4,0,.5),cex=.8)
```

Nous testons d'abord que `x` est stationnaire de moyenne constante. Pour cette hypothèse nulle, on utilise l'option `type = "mu"` dans `ur.kpss()`.

```
> summary(ur.kpss(x,type ="mu"))

#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 7 lags.

Value of test-statistic is: 0.0562

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

La statistique prend la valeur 0.0562, qui correspond à une p-value très supérieure à 10%. On ne rejette donc pas l'hypothèse de stationnarité. Considérons maintenant l'hypothèse nulle que `x` est stationnaire à une tendance linéaire près. On teste cette hypothèse grâce à l'option `type = "tau"`.

```
> summary(ur.kpss(x,type="tau"))

#######################
# KPSS Unit Root Test #
#######################

Test is of type: tau with 7 lags.

Value of test-statistic is: 0.0502

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.119 0.146  0.176 0.216
```

On ne rejette pas l'hypothèse de stationnarité de `x` à une tendance près.
Examinons maintenant la série intégrée `y` et testons sa stationnarité d'abord à un niveau moyen constant près, puis à une tendance linéaire près.

```
> summary(ur.kpss(y,type="mu"))
```

```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 7 lags.

Value of test-statistic is: 9.313

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

```
> summary(ur.kpss(y,type="tau"))
```

```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: tau with 7 lags.

Value of test-statistic is: 0.7157

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.119 0.146  0.176 0.216
```

Dans les deux cas, le niveau de signification empirique correspond à des p-values extrêmement faibles. On rejette chaque fois l'hypothèse de stationnarité. L'exemple est caricaturalement simple.

Exemple 5.4 (Taux d'intérêt - suite) Considérons maintenant la série `i1` pour laquelle on n'a pas pu conclure (stationnarité/non-stationnarité) au paragraphe précédent. L'examen du chronogramme de la série ne permet pas de décider a priori si elle peut être stationnaire à un niveau moyen près, à une tendance linéaire près, donc on considère les deux cas.

```
> summary(ur.kpss(i1,type="mu"))
```

```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 3 lags.

Value of test-statistic is: 0.2737

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

Le niveau de signification empirique est très élevé : on conclut que la série est stationnaire. Mais considérons l'hypothèse nulle : la série est stationnaire à une tendance déterministe près.

```
> summary( ur.kpss(i1,type="tau"))

#######################
# KPSS Unit Root Test #
#######################

Test is of type: tau with 3 lags.

Value of test-statistic is: 0.1439

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.119 0.146  0.176 0.216
```

La p-value est un peu plus faible que 5% : on conclut à la non-stationnarité de la série. Essayons une modélisation ARIMA. On essaie d'abord un ARIMA(1,1,1) avec dérive.

```
> (m1=Arima(i1,order=c(1,1,0),include.drift=TRUE))

Series: i1
ARIMA(1,1,0) with drift
...
Coefficients:
         ar1  drift
      0.1852  6e-04
s.e.  0.1256  2e-03

sigma^2 estimated as 0.0001636:  log likelihood = 179.33
AIC = -352.66   AICc = -352.23   BIC = -346.32
```

La dérive n'est pas significative, résultat cohérent avec le chronogramme de la série. On essaie également un modèle stationnaire AR(2).

```
> (m2=Arima(i1,order=c(2,0,0),include.mean=TRUE))

...
Coefficients:
         ar1      ar2  intercept
      1.1116  -0.2880     0.0984
s.e.  0.1211   0.1262     0.0085

sigma^2 estimated as 0.0001478:  log likelihood = 184.67
AIC = -361.35   AICc = -360.65   BIC = -352.84

> ret=c(3,6,9,12)
> t(Box.test.2(residuals(m2),nlag=ret,type="Ljung-Box",fitdf=3))
```

```
            [,1]      [,2]     [,3]       [,4]
Retard  3.0000000 6.0000000 9.000000 12.0000000
p-value 0.9528142 0.9966868 0.967664  0.8455333
```

```
> t_stat(m2)
```

```
           ar1       ar2 intercept
t.stat 9.17698 -2.281389  11.51864
p.val  0.00000  0.022525   0.00000
```

L'ajustement est satisfaisant, les critères d'information sont uniformément plus faibles que pour l'ajustement par un ARIMA(1,1,0) et la somme des coefficients d'autorégression vaut 0.922, sensiblement inférieur à 1.

5.4 Significativité illusoire en régression

Il arrive que la régression d'une série temporelle sur une autre soit très significative, alors que : (1) l'examen des résidus montre que les présupposés nécessaires à une régression pertinente ne sont pas réunis ou que (2) la nature même des données interdit d'établir un lien logique entre les variables. Une telle significativité n'a donc pas de sens dans cette situation. On peut parler alors de *significativité illusoire*, mais en fait, on parle, de façon approximative, de régression illusoire ou fallacieuse (*spurious regression*). Le problème se rencontre dans toutes les disciplines utilisant des méthodes de régression.
Les régressions rencontrées ici concernent : (1) la régression d'une série temporelle sur des séries non aléatoires, par exemple la régression de la température à Nottingham Castle sur des fonctions périodiques du temps ou (2) la régression d'une série temporelle sur des séries prédéterminées (consommation d'électricité régressée sur des transformations de série de température). Les difficultés surviennent quand on régresse une série temporelle x_{1t} sur une série de même nature x_{2t}, toutes deux non stationnaires. Ici, nous donnons seulement un aperçu du problème à travers un exemple. Nous considérons deux indices boursiers qui ont une évolution parallèle : le Nikkei et le Nasdaq. Effectuons une régression linéaire de l'un sur l'autre ; elle semble très significative mais, après examen du résidu, nous concluons que la relation n'est pas stable.

Exemple 5.5 (Nikkei et Nasdaq) Observons deux indices boursiers : le Nikkei et le Nasdaq. Leurs valeurs sont dans `indbourse.RData`, en positions 1 et 4.

```
> data(indbourse)
> nikkei=indbourse[,1]
> nasdaq=indbourse[,4]
```

On dessine le diagramme de dispersion et les chronogrammes superposés de ce couple. Pour ce dernier graphique, on ramène d'abord les deux séries à un même ordre de grandeur. Comme les deux premières valeurs de `nikkei` manquent (il n'y a pas eu de cotations à Tokyo les 2 et 3 janvier 2006, alors qu'il y en avait à New York), on met à l'échelle sur la valeur de la troisième date.

```
> fac0=as.numeric(nikkei[3]/nasdaq[3])
> nas1=fac0*nasdaq
```

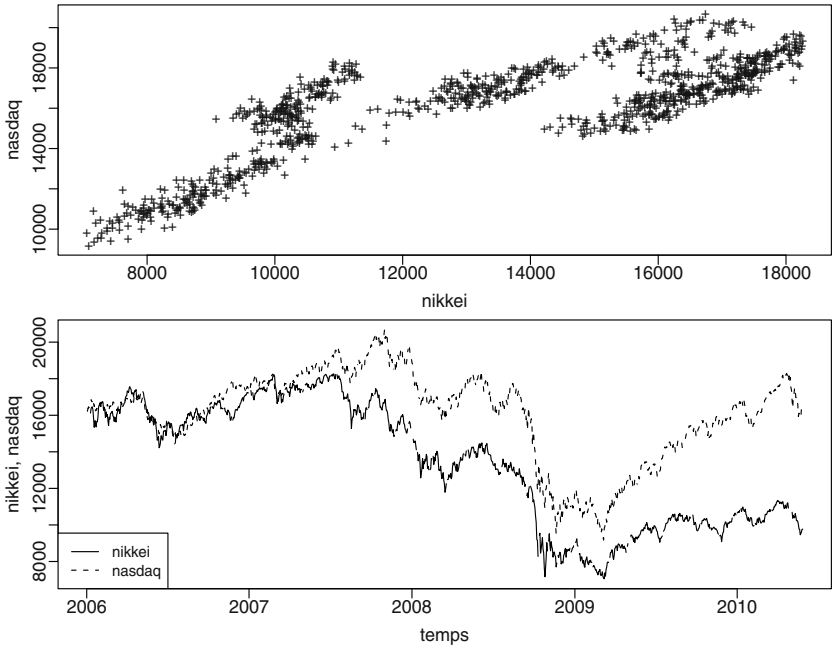

Fig. 5.5 – Diagramme de dispersion et chronogrammes des indices Nikkei et Nasdaq.

Nous voulons étiqueter le chronogramme en la première cotation de chaque année. Dans ce but, nous devons repérer les numéros des premières observations de chaque année. Nous utilisons les dates données sous forme de chaînes de caractères dans le nom des lignes de la série, `dimnames(nikkei@.Data)[[1]]`. Nous extrayons l'année de ces chaînes, `valan`, puis les noms des années différentes par `unique()`. Pour chaque année, nous repérons par `which()` la position de la première date de cotation dans l'année. A l'observation ayant cette position, nous donnons comme étiquette l'année correspondante. Dans l'appel de `plot.ts()`, nous demandons que l'axe des abscisses ne soit pas étiqueté dès l'appel de la fonction. L'étiquetage est fait ensuite par `axis()`.

```
> valan=substr(dimnames(nikkei@.Data)[[1]],1,4)
> lesdates=dimnames(nikkei@.Data)[[1]]
> anunique=unique(valan)
> man1=rep(NA,length(anunique))
> for(i in 1:length(anunique)){
+ sousdate=lesdates[valan==anunique[i]]
+ man1[i =which(lesdates==sousdate[1])}
```

```
> plot(nikkei,nas1,xlab="nikkei",ylab="nasdaq",pch="+")
> plot.ts(cbind(nikkei,nas1),plot.type='single',
+  ylab='nikkei, nasdaq',xlab='temps',xaxt='n',lty=1:2)
> axis(1,at=man1,labels=substr(lesdates[man1],1,4))
```

Ces deux séries (fig. 5.5) sont manifestement non stationnaires : elles sont I(1) comme on peut le vérifier facilement. On observe sur les chronogrammes superposés des évolutions, parfois parallèles, parfois divergentes, des deux séries. La question se pose : ces écarts sont-ils stationnaires ou bien les deux séries ont-elles des évolutions complètement indépendantes ? Sans entrer dans des détails théoriques, nous admettrons que si le résidu de la régression de `nikkei` sur `nasdaq` est stationnaire, alors on peut considérer que les séries ont des évolutions parallèles.

Effectuons maintenant cette régression grâce aux commandes :

```
> mod2=lm(nikkei@.Data~nasdaq@.Data)
> aa=summary(mod2)
> aa


Call:
lm(formula = nikkei@.Data ~ nasdaq@.Data)

Residuals:
    Min      1Q  Median      3Q     Max
-4862.2 -1306.5  -172.4  1893.7  3656.4

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3844.5986   419.8328  -9.157   <2e-16 ***
nasdaq@.Data     7.7550     0.1868  41.518   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2030 on 1043 degrees of freedom
  (103 observations deleted due to missingness)
Multiple R-squared: 0.623,        Adjusted R-squared: 0.6227
F-statistic:  1724 on 1 and 1043 DF,  p-value: < 2.2e-16
```

Nous obtenons R2 = 0.623, valeur très élevée et la régression semble très significative. Pour aller plus loin, dessinons le chronogramme du résidu et son ACF (fig. 5.6) grâce aux commandes :

```
> plot.ts(residuals(mod2),xlab='temps',ylab='résidu MCO')
> abline(h=0)
> acf(residuals(mod2),main="",xlab='retard')
```

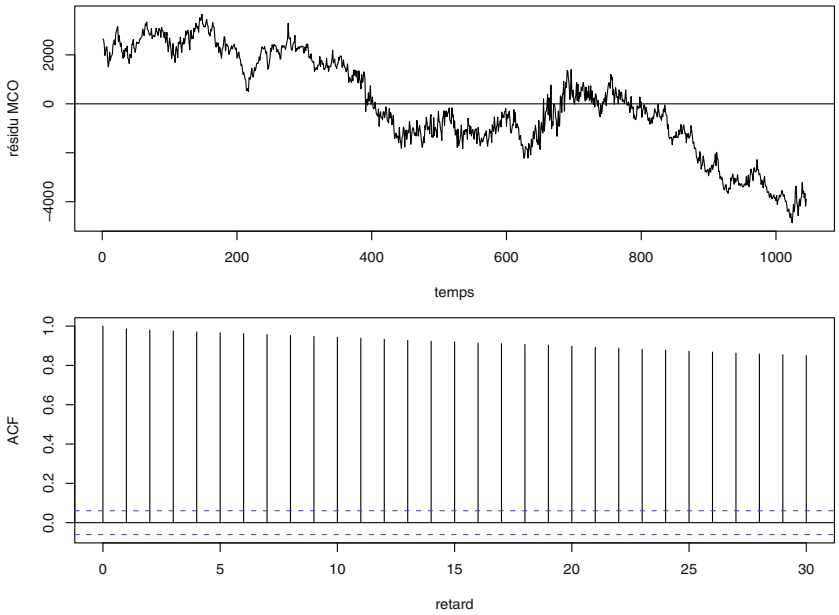

Fig. 5.6 – Résidu de la régression de `nikkei` sur `nasdaq` - Chronogramme et ACF.

Le chronogramme du résidu montre de longues séries de valeurs de même signe ; typiquement, ce résidu n'est pas stationnaire.
Le présupposé de variance constante ou au moins de variance obéissant à un mécanisme déterministe ne tient pas. On ne peut pas appliquer les MCO ou les MCG car la relation entre les deux indices n'est pas stable.

Cet exemple nous a conduit à effectuer une régression ayant une significativité illusoire : un R2 élevé et une régression apparemment très significative, mais qui en réalité débouche sur un résidu non stationnaire.

Exercice 5.4 (Régression d'une marche aléatoire)
Simuler deux marches aléatoires x et y indépendantes, par exemple à l'aide du code :

```
> set.seed(514); nobs=300
> y0=2+rnorm(nobs)
> y=diffinv(y0)
> x0=1-1.5*rnorm(nobs)
> x=diffinv(x0)
```

1. Dessiner le diagramme de dispersion de (x,y).
2. Superposer les chronogrammes des deux séries.
3. Que suggèrent ces graphiques ?
4. Effectuer la régression linéaire de y sur x.
5. Etudier la stationnarité du résidu et conclure sur la pertinence de cette régression.

Cadre de la régression fallacieuse. Le résultat que nous avons illustré se situe dans le contexte suivant. Quand on régresse une série y_t sur une série x_t, différentes situations peuvent survenir.

1. Les deux séries sont stationnaires, ou stationnaires à une tendance déterministe près, alors le modèle de régression classique tient.
2. Les deux séries sont intégrées d'ordres différents, alors une équation de régression les reliant n'a pas de sens.
3. Les deux séries sont intégrées de même ordre et le résidu de la régression de l'une sur l'autre est non stationnaire, alors cette régression n'a pas de sens. Un remède consiste à différencier les deux séries de la même façon pour les rendre stationnaires, mais si la régression sur les séries différenciées a un sens statistique, elle peut n'avoir pas le même sens concret que celle qu'on a dû abandonner sur les séries. Par exemple, on cherchait une relation entre `nikkei` et `nasdaq`, mais non entre leurs accroissements.
4. Les deux séries sont intégrées de même ordre et le résidu de la régression de l'une sur l'autre est stationnaire, alors la régression entre elles a un sens. On dit que les deux séries coïntègrent.

La régression fallacieuse et la coïntégration concernent la modélisation des séries vectorielles et sont étudiées dans la plupart des ouvrages d'économétrie. Par exemple, Enders (1995) en donne une présentation élémentaire mais non simpliste et Hamilton (1994), une présentation théorique.

Chapitre 6

Lissage exponentiel

Le lissage exponentiel est pratiqué depuis plus de 50 ans. Méthode d'abord purement intuitive, il a connu depuis une vingtaine d'années un développement théorique important et s'appuie sur un modèle basé associé à une représentation espace-état et sur le filtre d'innovation. Le fait qu'un modèle soit disponible permet de ne pas se contenter de prévisions ponctuelles, mais de calculer également des intervalles de prévision. De nombreuses méthodes récentes de lissage exponentiel sont disponibles dans `ets()` de **forecast**.

6.1 Lissage exponentiel

L'expression *lissage exponentiel* désigne un ensemble de méthodes de calcul de prédictions d'une série, centrées sur une mise à jour facile de la prédiction de la série quand une nouvelle observation est disponible. Ces méthodes partent d'une décomposition de série en tendance, saisonnalité et erreur, et proposent un mécanisme de mise à jour de la tendance et de la saisonnalité quand une nouvelle observation est disponible.
La prédiction de y_{t+h} connaissant le passé $y_t, y_{t-1}, \cdots$ de la série est l'espérance conditionnelle au passé de la série (cf. section 4.4). On note indifféremment cette prédiction :

$$\mathsf{E}(y_{t+h}|y_t, y_{t-1}, \cdots) \quad \text{ou} \quad y_{t+h|t}.$$

Pour l'horizon $h = 1$, on note aussi cette prédiction μ_t : $\mu_t \equiv y_{t+1|t}$.

6.1.1 Lissage exponentiel simple

Supposons une série $y_1, y_2, \cdots$, *sans saisonnalité* et montrant une *tendance localement constante*, c'est-à-dire que son niveau reste à peu près constant pour des dates proches. Nous voulons prédire la prochaine observation.

► **Présentation classique - approche descriptive** . Il est naturel de prédire y_{t+1} par une moyenne pondérée des valeurs passées $y_1, y_2, \cdots, y_t$:

$$\mu_t = c_0 y_t + c_1 y_{t-1} + c_2 y_{t-2} + \cdots .$$

Les c_i sont des poids positifs à définir. Comme la série évolue peu d'une date sur l'autre, on choisit de la prédire à un horizon h, pas trop élevé, par

$$y_{t+h|t} = \mu_t, \ \forall h \geq 1. \tag{6.1}$$

Comme la série évolue au cours du temps, il est logique d'affecter un poids plus important aux valeurs récentes qu'à celles du début de la série, on choisit des poids géométriques :

$$c_i = \alpha(1-\alpha)^i, i = 0, 1, \cdots$$

où α est une constante, $0 < \alpha < 1$. Ainsi, dans cette méthode de prévision, on choisit de prédire y_{t+1} par :

$$\mu_t = \alpha y_t + \alpha(1-\alpha) y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \cdots . \tag{6.2}$$

On peut voir une récurrence dans (6.2) :

$$\begin{aligned} \mu_t &= \alpha y_t + (1-\alpha)(\alpha y_{t-1} + \alpha(1-\alpha)^1 y_{t-2} + \cdots) \\ &= \alpha y_t + (1-\alpha)\mu_{t-1}. \end{aligned} \tag{6.3}$$

La deuxième ligne de (6.3) est la formule de mise à jour de la prédiction quand arrive une nouvelle observation. Posons :

$$e_t = y_t - \mu_{t-1}, \tag{6.4}$$

c'est l'erreur sur y_t quand on le prédit à partir de la série jusqu'en $t-1$. On a encore

$$\mu_t = \mu_{t-1} + \alpha e_t. \tag{6.5}$$

Vu (6.1), la prévision ne dépend pas de l'horizon et μ_{t-1} est aussi la prévision de y_{t+1}, connaissant la série jusqu'en $t-1$. La nouvelle prévision à l'horizon 1 est ainsi une combinaison convexe de l'ancienne prévision à l'horizon 2 et de la dernière valeur observée. L'écriture (6.5) est la forme dite à *correction d'erreur*. Ce modèle est aussi appelé *modèle à moyenne localement constante* et la prévision μ_t est le niveau local noté également l_t. L'équation (6.3) s'écrit donc

$$l_t = \alpha y_t + (1-\alpha) l_{t-1}$$

ou

$$l_t = l_{t-1} + \alpha e_t. \tag{6.6}$$

C'est une équation de mise à jour de la prévision quand arrive une nouvelle observation. Plus α est proche de 1,

– plus les poids décroissent rapidement et moins les valeurs anciennes interviennent dans la moyenne pondérée ;
– plus le passé récent est important par rapport au passé éloigné ;
– plus la correction de la prévision est importante quand arrive une nouvelle observation.

Bien qu'il y ait un nombre fini d'observations, on peut raisonner et calculer sur cette expression infinie sans perdre beaucoup de précision (voir la vignette `Anx6`).

Si on pose $\mu_1 = y_1$, α étant choisi, on dispose d'une formule complète de mise à jour de la prédiction à l'horizon 1 quand arrive une nouvelle observation. C'est une formule très simple. La procédure décrite par (6.3) s'appelle *lissage exponentiel simple* (*SES*) pour *Simple Exponential Smoothing* : *exponentiel* parce que les poids (qui forment une suite géométrique) décroissent exponentiellement. Notons enfin qu'on a parlé de « modèle » : il s'agit d'un modèle descriptif et non probabiliste.

On prend souvent α entre 0.1 et 0.3, les observations anciennes ont donc un poids élevé. On peut également l'estimer sur la série disponible : pour une valeur de α, on calcule la suite des erreurs $e_1,\ e_2, \cdots, e_N$ puis on forme $\sum e_i^2$. On retient la valeur de α qui minimise cette somme. Habituellement la courbe de la somme des carrés des erreurs en fonction de α est assez plate près de l'optimum et le choix n'a pas à être très précis. Si cette valeur optimale se trouve au bord de l'intervalle $(0, 1)$, il faut envisager une autre méthode de prévision.

► **Approche inférentielle du SES.** Considérons (6.4), e_t est l'erreur de prévision à l'horizon 1. Dans le cadre des séries linéaires et causales, e_t s'appellerait *innovation* (cf. section 4.4.3). e_t intervient également dans (6.6) pour mettre à jour la prévision à l'horizon 1 quand y_t est disponible. Cette lecture du lissage exponentiel simple amène à introduire un modèle probabiliste.

Définition 6.1 (Modèle de lissage exponentiel simple)
La série y_t obéit à un modèle de lissage exponentiel simple si elle vérifie :

$$\begin{aligned} l_t &= l_{t-1} + \alpha\epsilon_t \\ y_t &= l_{t-1} + \epsilon_t, \end{aligned} \tag{6.7}$$

où ϵ_t bruit blanc gaussien est l'innovation, l_t est l'état.

Ce modèle est décrit par deux équations :
– l'équation qui relie l'état à une date, à l'état à la date précédente, est l'**équation d'état** ou **de transition** ;
– l'équation qui donne la série observée en fonction de l'état l_{t-1} (ici unidimensionnel) non observé est l'**équation d'observation**.

C'est un exemple de modèle à *représentation espace-état*. Une représentation espace état est l'écriture d'un modèle de série temporelle à l'aide de deux équations : l'une, l'équation de transition, donne l'évolution de l'état, vecteur auxiliaire dont

certaines composantes peuvent présenter un intérêt, l'autre, l'équation d'observation, exprime la série observée en fonction de l'état et de l'innovation. Ce découpage est particulièrement commode pour la prévision et permet de construire un modèle en associant différents mécanismes (pour la tendance, la saisonnalité...).
On peut observer que les erreurs dans les deux équations de la représentation (6.7) dépendent du même bruit blanc, d'où le nom de *modèle à une seule source d'erreur* ou SSOE (*Single Source Of Error*).

Remarques
1. L'interprétation donnée à α dans l'approche descriptive demeure valable : plus il est élevé, plus le niveau l_t peut changer au cours du temps.
2. Il n'y a généralement pas une unique représentation espace-état pour un modèle donné.
3. Si l'on peut donner à un modèle une représentation espace-état, on retrouve une écriture standard et il est alors assez facile d'écrire la maximisation de vraisemblance. Sans le recours à une telle représentation, les moindres variations dans la définition du modèle peuvent donner lieu à des complications importantes dans l'écriture de l'estimation.
4. Il existe des modèles à représentation espace-état à plusieurs sources d'erreur ou MSOE (*Multiple Source Of Error*), le modèle BSM estimé dans R par `StructTS()` en est un exemple. Ils s'estiment par le filtre de Kalman.
5. On devrait rejeter (6.7) comme modèle de y_t si la série des résidus obtenus $\widehat{\epsilon}_t$ n'est pas un bruit blanc. Toutefois on se contente dans ces modèles d'un examen sommaire de la blancheur du résidu. C'est l'examen de la série qui a priori oriente vers le choix d'un tel modèle.

De l'équation (6.7) nous pouvons déduire une forme réduite du modèle de y_t. Par différenciation, nous voyons que y_t suit un ARIMA(0,1,1) :

$$(1 - \mathrm{B})y_t = \epsilon_t - (1 - \alpha)\epsilon_{t-1}. \tag{6.8}$$

Si la série suit effectivement un modèle ARIMA(0,1,1), la prévision par SES est optimale. Notons que $\alpha = 1$ correspond à une marche aléatoire.
En résumé, on est parti d'une idée assez intuitive de mécanisme de prédiction. Or on voit que ce mécanisme décrit également une représentation espace-état d'un ARIMA(0,1,1), d'où l'optimalité de ce mécanisme pour des séries de ce type.
L'approche traditionnelle du lissage exponentiel peut se faire par `HoltWinters()` de **stats**. Nous utilisons la fonction `ets()` de **forecast**, basée sur l'approche par représentation espace-état et qui donne des estimations des paramètres par maximum de vraisemblance. Le lissage exponentiel simple d'une série `y` est obtenu par `ets(y,model="ANN")` dont la syntaxe est expliquée après l'exemple qui suit.

Exemple 6.1 (`fmsales`) La série `fmsales` de **expsmooth** donne les ventes d'un produit sur 62 semaines à partir du début de 2003. On veut en faire la prévision à l'horizon 4 par SES. Le code nécessaire est :

```
> require(expsmooth)
> ets0=ets(fmsales,model="ANN")
> summary(ets0)
ETS(A,N,N)

Call:
 ets(y = fmsales, model = "ANN")

  Smoothing parameters:
    alpha = 0.7312

  Initial states:
    l = 23.4673

  sigma:  3.5496

     AIC     AICc      BIC
416.9693 417.1727 421.2236

In-sample error measures:
        ME       RMSE        MAE        MPE       MAPE
0.20127166 3.54958451 2.35036107 0.09804668 6.94976638
      MASE
0.94658312
```

La première lettre `A` de `model="ANN"` signifie que l'erreur est additive, la deuxième lettre concerne la tendance, `N` indique qu'il n'y en a pas, la troisième lettre concerne la saisonnalité, `N` indique qu'il n'y en a pas. Le modèle de lissage exponentiel est estimé par maximum de vraisemblance, par l'intermédiaire de la représentation espace-état. On obtient notamment :

– l'estimation de α ;
– l'estimation de l_1, état initial. On peut voir qu'il est de l'ordre de grandeur des premières valeurs de la série ;
– l'écart type du bruit ϵ_t dans (6.7) ;
– des critères d'information ;
– des mesures d'erreur intra-échantillon, voir la remarque ci-dessous.

La fonction `predict()` donne les prédictions à l'horizon qu'on choisit.

```
> (aaa=predict(ets0,4))
   Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
63       32.59211 28.04314 37.14109 25.63506 39.54917
64       32.59211 26.95672 38.22751 23.97352 41.21070
65       32.59211 26.04825 39.13597 22.58414 42.60008
66       32.59211 25.25136 39.93286 21.36541 43.81882
```

On observe que la prédiction est bien constante. La valeur 32.592 n'est autre que la prévision à l'horizon 1 à partir de la dernière observation (6.5). Elle est obtenue par

```
> e_t=fmsales[62]-ets0$fitted[62]
> ets0$fitted[62]+ets0$par[1]*e_t

   alpha
32.59211
```

Pour avoir l'ensemble des sorties de `predict()`, on examine `str(aaa)`. Les graphiques (fig. 6.1 et 6.2) de la série et de sa prévision avec, par défaut, des intervalles de prévision à 80 et 95%, s'obtiennent par :

```
> plot(fmsales,xlab='temps',ylab='Ventes',
+   main=expression(paste("Série",plain(fmsales))))
> plot(aaa,xlab='temps',ylab='Ventes',
+   main=expression(paste("Prédiction de ",plain(fmsales),"à l'horizon 4")))
```

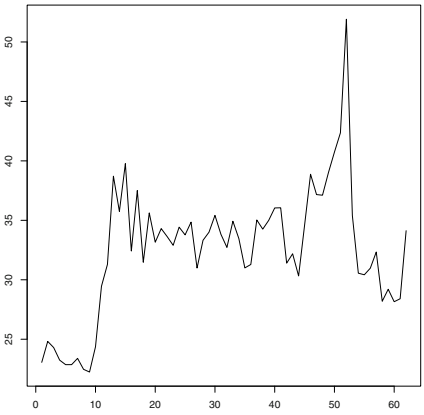

Fig. 6.1 – Série `fmsales`.

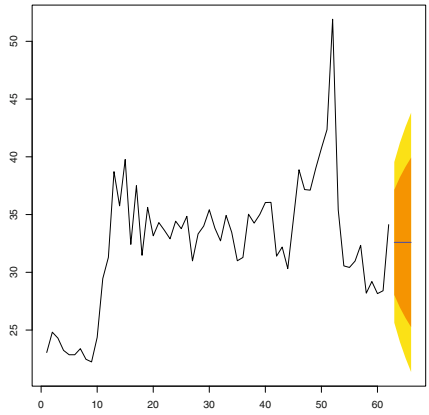

Fig. 6.2 – Prédiction par lissage exponentiel simple de `fmsales`.

Hyndman & Khandakar (2008) ou Hyndman *et al.* (2008) développent les questions théoriques et pratiques de cette approche, seulement esquissée ici.

Exercice 6.1 (Compléments sur `fmsales`)
Examiner la sortie `ets0` puis :

1. Repérer l'état initial, vérifier (6.7) sur quelques observations.
2. Expliquer pourquoi `ets0$mse` = `ets0$sigma2` (cf. section 4.4).
3. Donner les paramètres de ce modèle en plus de la variance du bruit.
4. Tester la blancheur du résidu.

Exercice 6.2 (Lissage exponentiel simple par la méthode de Holt-Winters)

1. Faire la prévision de `fmsale` à l'horizon 4 à l'aide de la fonction `HoltWinters()`.
2. Comparer dans les deux approches les valeurs du paramètre α, les vecteurs donnant le niveau.

Remarque (Mesures d'erreur intra-échantillon)
La fonction `summary()` appliquée à une sortie de `ets()` ou de `Arima()` fournit un certain nombre de *mesures d'erreur intra-échantillon* décrites par Hyndman *et al.* (2008). Ces mesures sont basées sur l'erreur de prévision à l'horizon 1 dans l'échantillon, c'est-à-dire sur $y_t - y_{t|t-1}$, l'innovation, notée e_t ou z_t, estimée par $\widehat{z}_t$.
Ainsi nous avons les mesures suivantes, exprimées dans les unités de `y` :
– ME $= (1/n)\sum e_t$, *Mean Error* ;
– RMSE $= \sqrt{(1/n)\sum e_t^2}$, *Root Mean Square Error* qui est évidemment la racine carrée de l'erreur quadratique moyenne, MSE ;
– MAE $= (1/n)\sum |e_t|$, *Mean Absolute Error*.

Les autres mesures sont données en pourcentage. Si nous notons l'erreur en pourcentage $p_t = 100e_t/y_t$, alors les autres mesures proposées s'écrivent :
– MPE $= (1/n)\sum p_t$, *Mean Percentage Error* ;
– MAPE $= (1/n)\sum |p_t|$, *Mean Absolute Percentage Error*.

Ces erreurs relatives sont utiles pour comparer des ajustements sur des données différentes, mais si y_t est proche de 0, l'erreur p_t se trouve amplifiée. La dernière mesure, le MASE, corrige ce problème. Posons

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n} |y_i - y_{i-1}|}$$

le dénominateur de q_t est, en quelque sorte, l'erreur moyenne absolue si on prédit une observation par l'observation précédente, alors,
– MASE $= (1/n)\sum q_t$.

6.1.2 Lissage exponentiel double

Dans cette section, nous envisageons des modèles à moyenne localement linéaire et la méthode de Holt. Nous supposons que la série à prédire montre une *tendance localement linéaire* : sur de courts intervalles de temps, elle évolue à peu près comme une droite dont l'équation peut changer légèrement au cours du temps. Il est sensé prédire une telle série à l'horizon h par une droite :

$$y_{t+h|t} = l_t + hb_t,\ h = 1, 2, \cdots, \tag{6.9}$$

où on utilise la dernière pente estimée b_t et la dernière ordonnée à l'origine estimée l_t pour prédire le niveau en $t + h$. La prévision à l'horizon 1, en $t - 1$, c'est-à-dire la prévision de y_t, est donc :

$$\mu_t = l_{t-1} + b_{t-1} \tag{6.10a}$$

et l'erreur de prédiction à l'horizon 1 :

$$e_t = y_t - \mu_t. \tag{6.10b}$$

On appelle b_t la pente et l_t le niveau, qu'on peut également comprendre comme l'ordonnée à l'origine de la droite donnant la prévision à partir de la date t.
Dans la méthode de Holt, une fois y_t disponible, on met à jour le niveau par :

$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}), \tag{6.11a}$$

où $\alpha \in (0,1)$ doit être choisi. l_t est en quelque sorte une prévision de y_t où l'on n'utilise que partiellement la connaissance de y_t. Le nouveau niveau est ainsi une combinaison convexe de sa prédiction et de la nouvelle valeur disponible. On l'écrit comme :

$$l_t = l_{t-1} + b_{t-1} + \alpha(y_t - (l_{t-1} + b_{t-1})) = l_{t-1} + b_{t-1} + \alpha e_t,$$

il obéit à un mécanisme de mise à jour basé sur l'erreur de prédiction à l'horizon 1. On met à jour la pente par :

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}, \tag{6.11b}$$

où $\beta^* \in (0,1)$. Enfin, faisons apparaître e_t dans ce mécanisme :

$$b_t = b_{t-1} + \beta e_t, \tag{6.11c}$$

où $\beta = \alpha\beta^*$. Si α est proche de 1, il provoque de fortes corrections de niveau, s'il est proche de 0, le niveau évolue peu. Un β^* proche de 1 provoque des variations fortes de la pente et donc du niveau.
En résumé, quand l'observation y_t devient disponible, on calcule e_t et la mise à jour de la prédiction s'effectue ainsi. On met d'abord à jour le niveau :

$$l_t = l_{t-1} + b_{t-1} + \alpha e_t, \tag{6.12a}$$

puis la pente :

$$b_t = b_{t-1} + \beta e_t, \tag{6.12b}$$

enfin la prédiction à l'horizon h :

$$y_{t+h|t} = l_t + hb_t. \tag{6.12c}$$

Nous pouvons maintenant associer un modèle à représentation espace-état à cette démarche encore purement descriptive. On choisit comme état le vecteur des deux composantes de la prédiction :

$$\mathbf{x}_t = \begin{bmatrix} l_t \\ b_t \end{bmatrix}.$$

Introduisant les matrices :

$$\mathbf{w} = \begin{bmatrix} 1 & 1 \end{bmatrix}, \ \mathbf{F} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \ \mathbf{g} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \tag{6.13}$$

nous pouvons définir le modèle de lissage exponentiel double basé sur une représentation espace-état SSOE.

Définition 6.2 (Modèle de lissage exponentiel double)
La série y_t obéit à un modèle de lissage exponentiel double si elle vérifie :

$$y_t = \mathbf{w}\boldsymbol{x}_{t-1} + \epsilon_t \tag{6.14a}$$

$$x_t = \mathbf{F}x_{t-1} + \mathbf{g}\epsilon_t, \tag{6.14b}$$

où ϵ_t est l'innovation, bruit blanc gaussien. $\mathbf{F}$ est la matrice de transition.

Nous donnons quelques notions intuitives sur cette question. Le passé jusqu'en t s'exprime indifféremment à l'aide de $y_t, y_{t-1}, y_{t-2}, \cdots$ ou de $\epsilon_t, \epsilon_{t-1}, \epsilon_{t-2}, \cdots$. Par (6.14a), la prévision à l'horizon 1 est :

$$\mu_t = \mathbf{w}\mathbf{x}_{t-1} = l_{t-1} + b_{t-1}.$$

De même :

$$y_{t+1|t-1} = \mathbf{w}x_{t|t-1} + 0 = \mathbf{w}\mathbf{F}x_{t-1}; \cdots \; y_{t+h|t-1} = \mathbf{w}\mathbf{F}^h x_{t-1}.$$

Comme

$$\mathbf{F}^h = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}^h = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \quad h > 0,$$

on obtient :

$$y_{t+h|t-1} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \begin{bmatrix} l_{t-1} \\ b_{t-1} \end{bmatrix} = l_{t-1} + (h+1)b_{t-1}$$

qui concorde avec (6.9). Pour utiliser cette méthode par `ets()`, on doit choisir l'option `model="AAN"`.

Exemple 6.2 Ajustons un modèle à moyenne localement linéaire à `fmsales` pour laquelle on a vu qu'un modèle de lissage exponentiel simple, modèle à moyenne localement constante, convient.

```
> ses.2=ets(fmsales,model="AAN")
> summary(ses.2)

ETS(A,A,N)

Call:
 ets(y = fmsales, model = "AAN")

  Smoothing parameters:
    alpha = 0.7382
    beta  = 1e-04

  Initial states:
    l = 23.7075
    b = 0.1667
```

```
  sigma:  3.5447

     AIC     AICc      BIC
420.7979 421.4997 429.3065

In-sample error measures:
        ME       RMSE        MAE        MPE       MAPE
-0.0302450  3.5446820  2.3676023 -0.6372273  7.0581764
      MASE
 0.9535269
```

La prédiction (à l'horizon 4, encore) s'obtient ensuite par `predict()` appliquée à l'objet `ses.2`.

```
> predict(ses.2,h=4)

   Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
63       32.85680 28.31411 37.39950 25.90935 39.80425
64       33.02335 27.37640 38.67030 24.38708 41.65962
65       33.18989 26.62155 39.75824 23.14448 43.23531
66       33.35644 25.98072 40.73216 22.07625 44.63663
```

En différenciant deux fois y_t obéissant à (6.14), on obtient que y_t obéit à un ARIMA(0,2,2) :

$$(1-\mathrm{B})^2 y_t = (1-\theta_1 \mathrm{B} - \theta_2 \mathrm{B}^2)\epsilon_t,$$

où $\theta_1 = \alpha + \beta - 2$ et $\theta_2 = 1 - \alpha$. Pour plus de détails, le lecteur intéressé pourra consulter Hyndman *et al.* (2008, p. 169).

6.1.3 Méthode de Holt-Winters et modèle de lissage correspondant

Nous envisageons maintenant le cas de séries présentant deux composantes additives : une tendance et une saisonnalité de période m. Comme précédemment, nous rappelons la méthode de lissage exponentiel de Holt-Winters, traditionnelle pour de telles séries, avant d'examiner la version inférentielle du même lissage.
La méthode de Holt-Winters prédit y_t connaissant la série jusqu'en $t-1$ par :

$$y_{t|t-1} = l_{t-1} + b_{t-1} + s_{t-m}, \tag{6.15}$$

où s_{t-m} est la composante saisonnière à la date $t-m$. L'erreur de prévision s'écrit :

$$e_t = y_t - y_{t|t-1}.$$

Quand l'observation y_t devient disponible, les composantes sont mises à jour en commençant par le niveau :

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}). \tag{6.16a}$$

Notre connaissance de la saisonnalité pour la date t remonte en effet à $t-m$. Ensuite on met à jour la pente :

$$b_t = \beta^*(l_t - l_{t-1}) + (1-\beta^*)b_{t-1}, \tag{6.16b}$$

enfin la saisonnalité et la prévision :

$$s_t = \gamma^*(y_t - l_{t-1} - b_{t-1}) + (1-\gamma^*)s_{t-m}. \tag{6.16c}$$

La prévision à l'horizon h est donnée par :

$$y_{t+h|t} = l_t + b_t h + s_{t-m+h_m^+}, \tag{6.16d}$$

où $h_m^+ = [(h-1) \mod m]+1$. Cette méthode est aussi appelée *lissage exponentiel triple*. On peut trouver les formules de mise à jour des composantes pour cette méthode, par exemple dans Gourieroux & Monfort (1995, p. 118).
Faisant apparaître l'erreur de prédiction e_t dans les différentes équations, on peut écrire un modèle probabiliste parallèle au modèle descriptif précédent.

$$\begin{aligned} y_t &= l_{t-1} + b_{t-1} + s_{t-m} + \epsilon_t && (6.17a)\\ l_t &= l_{t-1} + b_{t-1} + \alpha\epsilon_t && (6.17b)\\ b_t &= b_{t-1} + \beta\epsilon_t && (6.17c)\\ s_t &= s_{t-m} + \gamma\epsilon_t, && (6.17d) \end{aligned}$$

où : $\beta = \alpha\beta^*$, $\gamma = (1-\alpha)\gamma^*$ et ϵ_t est une innovation. Explicitons maintenant la représentation espace-état de (6.17) quand $m = 4$. D'abord nous définissons l'état :

$$\mathbf{x}_t = \begin{bmatrix} l_t & b_t & s_t & s_{t-1} & s_{t-2} & s_{t-3} \end{bmatrix}',$$

puis

$$\mathbf{w} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{F} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \mathbf{g} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

La représentation espace-état est :

$$\begin{aligned} y_t &= \mathbf{w}\mathbf{x}_{t-1} + \epsilon_t \\ x_t &= \mathbf{F}x_{t-1} + \mathbf{g}\epsilon_t. \end{aligned}$$

Elle a la même forme que (6.14). Les paramètres sont l'état initial, la variance de l'innovation et $\mathbf{g}$. L'estimation et la prévision associée sont désignées sous le nom de *filtre d'innovation*. Encore une fois, on renvoie à Hyndman & Khandakar (2008) pour l'étude détaillée de ces modèles.

Remarques

1. Dans les différents modèles de lissage exponentiel, α, β et γ ont la même interprétation : plus l'un d'eux est élevé, plus le passé récent est important dans la mise à jour de la composante concernée. On peut faire le même constat sur la représentation espace-état. Par exemple, dans (8.5a), les trois premiers éléments de $\mathbf{g}\ \epsilon_t$ donnent l'erreur pour chaque composante. L'erreur est d'autant plus importante en écart type que l'est le passé récent dans la mise à jour.
2. La saisonnalité reconnue par le logiciel est celle donnée par la fréquence dans la définition de la série dans le type `ts`. Les composantes de l'état concernant la saisonnalité sont normalisées pour être identifiées et leur examen n'aide pas à la description de la série (cf. Hyndman *et al.* (2008, chap. 8)).

Tableau 6.1 – Modèles ajustés par `ets()`.

Composante tendancielle	Composante saisonnière		
	N (Absente)	A (Add.)	M (Multip.)
N (Absente)	N,N	N,A	N,M
A(Additive)	A,N	A,A	A,M
A_d (Additive amortie)	A_d,N	A_d,A	A_d,M
M (Multiplicative)	M,N	M,A	M,M
M_d (Multiplicative amortie)	M_d,N	M_d,A	M_d,M

3. Nous avons présenté les modèles correspondant à trois méthodes classiques de lissage exponentiel. D'autres mécanismes de mise à jour donnent des modèles proches et bien adaptés à certains types de données. Nous rencontrerons un modèle saisonnier à composantes multiplicatives dans la prévision du trafic passager à l'aéroport de Toulouse-Blagnac (chap. 8). L'article de Hyndman & Khandakar (2008) présente **forecast** et les différents modèles que peut ajuster `ets()` (tableau 6.1). Pour une présentation du filtre d'innovation, la méthode d'estimation des paramètres et pour bien d'autres questions, Hyndman, Koehler, Ord & Snyder (2008) constitue une référence complète.
4. La fonction `estMaxLik()` de **dse1** permet d'estimer un modèle en représentation espace-état par le filtre d'innovation.

Chapitre 7

Simulation

La simulation, c'est-à-dire la reconstitution d'événements aléatoires obéissant à un mécanisme choisi, est très utile sinon indispensable en statistique. Elle permet :
- de vérifier qu'on a compris les phénomènes aléatoires qu'on étudie ;
- d'apprécier ce que devient l'efficacité de méthodes d'estimation quand elles sont utilisées sur des données ne vérifiant pas les présupposés sous lesquelles elles se montrent optimales ;
- d'estimer des fonctions complexes des v.a. sans effectuer des calculs théoriques.

Ici on commence par rappeler des principes généraux de simulation. On détaille ensuite la simulation de différents modèles de séries temporelles et le choix de tels modèles selon qu'on doit simuler une série stationnaire ou non. Au passage apparaissent les fonctions de R pour la simulation de séries temporelles. En fin de chapitre figurent les modèles d'intervention.

7.1 Principe

Les ordinateurs peuvent fabriquer des nombres pseudo-aléatoires, c'est-à-dire obtenus par des opérations déterministes, nombres qui peuvent donc réapparaître au cours des simulations, mais avec une période très longue. Ils sont tirés dans la loi uniforme sur $(0, 1)$. Des transformations permettent ensuite d'obtenir des nombres tirés dans différentes lois : normale, Poisson, exponentielle, etc. Le démarrage d'une simulation se fait à partir d'un entier positif, la *graine* (*seed*). Si l'utilisateur choisit cette graine, il peut recommencer, si nécessaire, les mêmes tirages. Alternativement on peut laisser l'ordinateur la choisir. Pour aller plus loin, le lecteur pourra consulter Bouleau (2002), qui donne des bases théoriques de la simulation, Kennedy & Gentle (1980), qui présentent les transformations des nombres pseudo-aléatoires vers les lois de probabilité, et bien d'autres ouvrages.
Le générateur par défaut de nombres pseudo-aléatoires a une période égale à $2^{19937} - 1 \simeq e^{13819}$. R propose par ailleurs des fonctions permettant de simuler sui-

vant les lois uniforme, normale, de Cauchy... (`runif()`, `rnorm()`, `rcauchy()`...), de tirer avec ou sans remise un échantillon dans une population, `sample()`. Dans R, la graine est choisie par l'intermédiaire de `set.seed()`, par exemple `set.seed(1984)`. Une fois la graine choisie, on peut lancer un code pour simuler un ou plusieurs échantillons. Chaque fois qu'on relance un code de simulation à partir d'une même valeur de graine, on obtient la même série de valeurs. Dans ce livre on a systématiquement utilisé `set.seed()` pour que le lecteur puisse retrouver les résultats imprimés. Si l'utilisateur ne choisit pas de graine, elle est créée à partir de l'heure courante. Dans la pratique, avant de fixer la graine, il est important de faire tourner une simulation pour pouvoir observer la variété des résultats.

7.2 Simulation de séries temporelles

Nous supposons qu'on dispose d'un générateur de bruit blanc, gaussien dans la plupart de nos exemples, et nous nous intéressons à la simulation de trajectoires de séries obéissant à un modèle bâti à partir d'un bruit blanc. Considérons quelques situations avant d'entrer dans le détail de simulations particulières. Dans cette section, tous les paramètres ϕ, θ et la loi du bruit blanc sont donnés, soit arbitrairement, soit parce qu'on se situe après l'estimation d'un modèle.

7.2.1 Principe

Simulation d'un autorégressif d'ordre 2. Considérons un AR d'ordre 2 :

$$(1 - \phi_1 \mathrm{B} - \phi_2 \mathrm{B}^2) y_t = c + z_t$$

où c, ϕ_1, ϕ_2 et σ_z^2, la variance du bruit blanc gaussien z_t, sont connus.
On doit simuler une trajectoire de n valeurs de y_t. On voit qu'il faut écrire la récurrence :

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + z_t,\ t = -1, 0, 1, \cdots, n$$

en se fixant 2 valeurs initiales : y_{-1}, y_0 et en tirant n valeurs i.i.d. $\mathcal{N}(0, \sigma_z^2)$ à l'aide de `rnorm()`. Une fois tirées les n valeurs, on fabrique y, observation par observation : pour chaque t, on ajoute à z_t, $c + \phi_1 y_{t-1} + \phi_2 y_{t-2}$ ce qui donne y_t.
Comme on l'a rappelé au chapitre 4, le processus simulé est (1) stationnaire, si les racines de $1 - \phi_1 z - \phi_2 z^2 = 0$ sont de module strictement supérieur à 1, ou (2) non stationnaire et ARIMA, si une racine au moins est égale à 1 et aucune n'est inférieure à 1 en module, ou (3) non stationnaire et non ARIMA, si au moins une racine est inférieure à 1 en module. Si le processus est stationnaire, on fait fonctionner le mécanisme de la récurrence un certain temps pour s'affranchir des conditions initiales et atteindre le régime stationnaire avant de stocker les simulations. Ainsi, les premières valeurs stockées sont elles-même des observations du processus. Si le processus n'est pas stationnaire, il faut aussi fixer deux valeurs initiales.

Simulation d'un ARMA$(1,2)$. Considérons un ARMA(1,2) :

$$y_t = \mu + \frac{1+\theta_1 \mathrm{B} + \theta_2 \mathrm{B}^2}{1-\phi_1 \mathrm{B}} z_t, \qquad z_t \sim \mathrm{BBN}(0, \sigma_z^2), \tag{7.1}$$

où $\mu, \phi_1, \theta_1, \theta_2$ et σ_z^2, la variance du bruit blanc gaussien z_t, sont connus, $|\phi_1| < 1$. On veut simuler une trajectoire de n valeurs $y_1, \cdots, y_n$ de y_t. On voit qu'il faut écrire la récurrence suivante :

$$y_t = c + \phi_1 y_{t-1} + z_t + \theta_1 z_{t-1} + \theta_2 z_{t-2},\ t = \cdots, -1, 0, 1, \cdots, n \tag{7.2}$$

où $c = \mu(1-\phi_1)$. On doit se fixer une valeur initiale y_0 puis tirer $n+2$ valeurs z_t i.i.d. $\mathcal{N}(0, \sigma_z^2)$. A partir de ces valeurs, on obtient le bruit $w_t = z_t + \theta_1 z_{t-1} + \theta_2 z_{t-2},\ t = 1, \cdots, n$. Ensuite on programme une boucle qui fabrique y, observation par observation : pour chaque t, on ajoute à w_t, $c + \phi_1 y_{t-1}$. On aurait pu également simuler l'ARMA centré $(1+\theta_1 \mathrm{B} + \theta_2 \mathrm{B}^2)/(1-\phi_1 \mathrm{B}) z_t,\ t = 1, \cdots, n$ et ajouter μ ensuite. Mais rien n'empêche d'ajouter une fonction du temps plutôt qu'une constante. C'est ainsi qu'on peut simuler les modèles ARMAX et les modèles d'intervention.

Remarques

- Pour simuler un SARMA, c'est-à-dire un ARMA dont la partie autorégressive ou la partie moyenne mobile s'exprime comme un produit de deux polynômes, on peut effectuer ce produit puis écrire la récurrence ou dans les cas simples, faire appel à une fonction de R.
- Pour simuler un ARIMA(p,d,q) on peut simuler le processus différencié d fois, c'est-à-dire le processus stationnaire correspondant, puis l'intégrer d fois par `diffinv()`.

Dans la section suivante, nous illustrons ces situations en suivant ces étapes pas à pas ou, quand c'est possible, en utilisant les fonctions de R généralement optimisées pour cette tâche.

7.2.2 Illustration numérique

Dans les exemples qui suivent, on se donne un mécanisme qui décrit un processus $\{y_t\}$ et on veut simuler une trajectoire, $y_t,\ t = 1, \cdots, 200$, de la série obéissant à ce mécanisme. La série des valeurs obtenues est notée $y_t,\ t = 1, \cdots, 200$.

Exemple 7.1 (Simulation d'un ARMA$(1,2)$) Nous souhaitons simuler une série temporelle de $T = 200$ observations obéissant au modèle ARMA(1,2) :

$$y_t = -0.5 + \frac{1 - 0.3\mathrm{B} + 0.6\mathrm{B}^2}{1 + 0.8\mathrm{B}} z_t, \qquad z_t \sim \mathrm{BBN}(0, 1.5). \tag{7.3}$$

La série est stationnaire, de moyenne -0.5. En multipliant des deux côtés par le dénominateur on obtient :

$$y_t = -0.9 - 0.8 y_{t-1} + z_t - 0.3 z_{t-1} + 0.6 z_{t-2}. \tag{7.4}$$

Examinons deux façons de réaliser cette simulation : sans fonctions de simulation propres à cette tâche, puis avec les fonctions de R prévues à cet effet.

A. On simule d'abord l'erreur $v_t = z_t - 0.3z_{t-1} + 0.6z_{t-2}$ à partir d'une série simulée de bruit blanc :

```
> require(caschrono)
> set.seed(951)
> n2=250
> z0=rnorm(n2+2,sd=sqrt(1.5))
> vt=z0-0.3*Lag(z0,1)+0.6*Lag(z0,2)
> str(vt)

 num [1:252] NA NA 0.315 0.214 -0.175 ...

> vt=vt[-(1:2)]
```

Comme la partie MA est d'ordre 2, nous avons simulé 250+2 valeurs et éliminé les deux premières, qui sont d'ailleurs manquantes. Nous devons maintenant calculer par récurrence $y_t = -0.8 * y_{t-1} + v_t - 0.9$. D'abord nous formons $w_t = v_t - 0.9$, noté `wt`, puis nous calculons la récurrence $y_t = -0.8 * y_{t-1} + w_t$:

```
> moy=-.5; cc=moy*(1+0.8)
> wt=vt+cc
> y.n=filter(wt[-1],c(-0.8),method='recursive')
> y.n=y.n[-(1:50)]
```

Nous n'avons pas donné de valeur initiale pour y, `filter()` choisit alors 0. Enfin nous avons abandonné les 50 premières valeurs, qu'on peut considérer comme une période de rodage du mécanisme.

B. Nous pouvons simuler ce processus ARIMA à l'aide de `arima.sim()`. Tenant compte du fait qu'on a déjà simulé le bruit blanc dans `z0` nous pouvons l'utiliser dans la fonction :

```
> yc.n=moy+arima.sim(n=200,list(ar=-0.8,ma=c(-0.3,0.6)),
+  innov=z0[53:252],n.start=50,start.innov=z0[1:50])
```

On voit que la période de rodage est précisée par `n.start=50`. Si on n'utilise pas un bruit blanc précédemment simulé, on peut obtenir directement la simulation par :

```
> set.seed(281)
> yd.n=moy+arima.sim(n=200,list(ar=-0.8,ma=c(-0.3,0.6)),
+           sd = sqrt(1.5),n.start=50)
```

Si l'on compare la simulation obtenue en utilisant le bruit blanc fabriqué précédemment, notée `yc.n`, avec `y.n` n'utilisant pas `arima.sim()`, on ne trouve pas les mêmes valeurs. En effet, la fonction `arima.sim()` n'utilise pas les mêmes valeurs initiales.

C. Nous pouvons aussi utiliser la fonction `simulate()` de **dse** pour cette tâche. Cette fonction peut gérer des modèles multidimensionnels où y_t a p composantes. Un modèle ARMA multidimensionnel doit y être écrit sous la forme

$$A(\mathrm{B})y_t = B(\mathrm{B})z_t + C(\mathrm{B})u_t, \tag{7.5}$$

où :

– $A(a \times p \times p)$ est l'array contenant le polynôme d'autorégression, de retard maximum $a-1$;

– $B(b \times p \times p)$ est l'array contenant le polynôme de moyenne mobile, de retard maximum $b-1$;

– $C(c \times p \times m)$ donne les coefficients d'un input u_t à m composantes et c retards.

La fonction `ARMA()` définit ensuite le modèle à partir de ces arrays. Pour les séries unidimensionnelles, $p = 1$. Observons que cette fonction se contente de décrire l'algèbre d'une équation de récurrence telle que (7.5), indépendamment du type des séries z_t et u_t. L'aide en ligne de `ARMA()` de **dse** donne des détails indispensables. Le tableau 7.1 résume et compare les fonctions `arima.sim()` et `simulate()`. Les `...` sont précisés dans le code ci-dessous.

Tableau 7.1 – Fonctions de simulation dans R.

Fonction	Simulation de $y_t = \phi_1 y_{t-1} + z_t + \theta_1 z_{t-1} + \theta_2 z_{t-2} + c$
`arima.sim()`	$y_t = \underbrace{\phi_1 y_{t-1}}_{\mathrm{ar}=c(\phi_1)} + z_t + \underbrace{\theta_1 z_{t-1} + \theta_2 z_{t-2}}_{\mathrm{ma}=c(\theta_1,\theta_2)} + \underbrace{c}_{\mathrm{mean}=c}$
`simulate()`	$\underbrace{(1-\phi_1 \mathrm{B})}_{\mathrm{A}=\ldots} y_t = \underbrace{z_t + \theta_1 z_{t-1} + \theta_2 z_{t-2}}_{\mathrm{B}=\ldots} + \underbrace{\mathrm{C}\, u_t}_{\mathrm{C}=1, u_t=c}$

Pour simuler une trajectoire suivant (7.4), nous commençons par écrire le modèle avec les conventions de **dse**. Nous définissons donc la partie autorégressive, `AR`, la partie moyenne mobile, `MA`, et la matrice `C` que nous notons `Cmat`, coefficient de l'input. Ceci nous donne le modèle `mod1` :

```
> require(dse)
> vecauto=c(1,0.8)
> (AR =array(vecauto,c(length(vecauto),1,1)))

, , 1

     [,1]
[1,]  1.0
[2,]  0.8
```

```
> vecma=c(1,-0.3,0.6)
> MA=array(vecma,c(length(vecma),1,1))
> MA

, , 1

     [,1]
[1,]  1.0
[2,] -0.3
[3,]  0.6

> Cmat=1
> mod1=ARMA(A=AR,B=MA,C=Cmat)
```

Ensuite, on définit l'input : c'est la constante de (7.4). La simulation est maintenant bien définie.

```
> sampleT.a=250
> ut=cc*matrix(1,ncol=1,nrow=sampleT.a)
> y0.a=c(-.5,-.5) # on affecte la moyenne de la série aux 2 valeurs initiales
> set.seed(951)
> bb=rnorm(sampleT.a*sqrt(1.5))
> asim=simulate(mod1,y0=y0.a,noise=as.matrix(bb),input=ut,
+          sampleT=sampleT.a,nrow=sampleT.a)
> mean(asim$output)

[1] -0.5330506

> (m3=Arima(asim$output,order=c(1,0,2)))

Series: asim$output
ARIMA(1,0,2) with non-zero mean
...
Coefficients:
          ar1      ma1    ma2  intercept
      -0.7584  -0.4349  0.594    -0.5296
s.e.   0.0449   0.0511  0.059     0.0400

sigma^2 estimated as 0.923:  log likelihood = -346.14
AIC = 702.28   AICc = 702.53   BIC = 719.89
```

Les deux dernières lignes de code permettent des vérifications sommaires : on vérifie ainsi que la moyenne de la série simulée est proche de la moyenne théorique et que le modèle estimé est proche de celui qui a gouverné la simulation. Ce que R nomme `intercept` est ici la moyenne de la série (stationnaire).

Exemple 7.2 (Simulation d'un ARIMA$(p, 1, q)$) Pour simuler un ARIMA dont la série différenciée une fois obéit à (7.4), il suffit d'intégrer la série précédemment simulée :

```
y.int = diffinv(y.n)
```

ou bien directement, en s'inspirant de l'équation (5.6, chap. 5) :

```
y2.int=moy*(0:199)+arima.sim(n=199,list(order=c(1,1,2),
       ar=-0.8,ma=c(-0.3, 0.6)),sd=sqrt(1.5),n.start=50)
```

Notons que `moy` est la moyenne pour le processus ARMA (7.1). Pour le processus ARIMA, où la moyenne n'est pas définie, c'est la dérive. `arima.sim()` simule une série de longueur `n +` l'ordre d'intégration.

Exemple 7.3 (Simuler un SARMA) Nous allons simuler 200 observations de

$$y_t = 4 + \frac{1 + 0.6\mathrm{B}^2}{(1 + 0.8\mathrm{B})(1 - 0.7\mathrm{B}^4)} z_t, \qquad z_t \sim \mathrm{BBN}(0, 1.5).$$

C'est un processus stationnaire saisonnier. Il n'est pas possible d'utiliser directement `arima.sim()` ou `simulate()`. Se présentent deux possibilités : (1) effectuer d'abord le produit des polynômes décrivant l'autorégression ou (2) suivre les étapes par lesquelles on a présenté les SARMA.

- Première méthode : on effectue le produit des polynômes d'autorégression de façon à entrer dans les possibilités de la fonction `arima.sim()`, ce qui nous donne l'objet `autopol`. Au passage nous calculons les modules des racines du polynôme d'autorégression. Ensuite on peut simuler par `arima.sim()`, ce qui nous donne :

```
> require(polynom)
> autopol=polynomial(c(1,0.8))*polynomial(c(1,0,0,0,-0.7))
> #vérification (facultative) de la stationnarité
> Mod(polyroot(autopol))

[1] 1.093265 1.093265 1.093265 1.093265 1.250000

> #simulation
> ys=4+arima.sim(n=200,list(ar=-autopol[-1],ma=c(0,0.6)),sd=sqrt(1.5))
```

  Pour la fonction `arima.sim()`, y_t doit être exprimé en fonction des y retardés comme dans (7.2). C'est pourquoi on a donné dans `ar=-autopol[-1]` les coefficients du polynôme changés de signe.

- Deuxième méthode : nous reproduisons la démarche qui a introduit les SARMA au chapitre 4. L'objet `bt` du code ci-dessous correspond à b_t de l'équation (4.41) reproduite ici :

$$b_t = \frac{\Theta_s(\mathrm{B}^s)}{\Phi_s(\mathrm{B}^s)} z_t,$$

```
> bt=arima.sim(n=230,list(ar=c(0,0,0,+.7)),sd=sqrt(1.5))
> yt=4+arima.sim(n=200,list(ar=-0.8,ma=c(0,0.6)),innov=bt[31:230],
+                  n.start=30,start.innov=bt[1:30])
```

  Pour estimer le modèle de la série simulée, on doit imposer la nullité du coefficient MA d'ordre 1. On le fait par l'option `fixed=` de `arima()`, en indiquant pour chaque coefficient s'il est libre (`NA`) ou la valeur qu'il prend. On a rencontré cette démarche (section 4.5.1) pour l'estimation du modèle de `y2`.

7.3 Construction de séries autorégressives

Dans la section précédente, les modèles des séries à simuler étaient entièrement spécifiés. Quand on veut s'assurer des qualités d'une méthode ou vérifier qu'on a compris son fonctionnement, on est amené à simuler des séries ayant des caractéristiques particulières. Dans cette perspective, nous examinons la simulation de séries autorégressives d'après les racines du polynôme d'autorégression.
Une autorégression s'écrit :

$$(1 - \phi_1 \mathrm{B} - \phi_2 \mathrm{B}^2 - \cdots - \phi_p \mathrm{B}^p) y_t = z_t.$$

La stationnarité de y_t dépend des racines du polynôme d'autorégression. Pour fabriquer une série stationnaire, les racines doivent être en module > 1 (chap. 4, section 4.2). Dès que l'ordre dépasse 2, il est commode de se donner les modules des racines puis d'en tirer le polynôme.

Exemple 7.4 Simulons une autorégression dont le polynôme d'autorégression admet les racines 1.4, -1.2 et 2. (On prendra comme erreur un bruit blanc gaussien de variance 1.5^2). Nous construisons d'abord le polynôme d'autorégression par l'intermédiaire de ses facteurs :

$$(1 - \frac{\mathrm{B}}{1.4})(1 + \frac{\mathrm{B}}{1.2})(1 - \frac{\mathrm{B}}{2}).$$

Le polynôme d'autorégression est :

```
> (autopol=polynomial(c(1,-1/1.4))*polynomial(c(1,1/1.2))*
+          polynomial(c(1,-1/2)))

1 - 0.3809524*x - 0.6547619*x^2 + 0.297619*x^3
```

enfin on peut simuler l'ARMA correspondant :

```
asim7 = arima.sim(n=200, list(ar = -autopol[-1]), sd=1.5)
```

Exercice 7.1 (ARIMA)
On veut simuler une série de 200 valeurs d'une autorégression dont le polynôme a deux racines strictement supérieures à 1 et une racine égale à 1 :

$$(1 - \frac{\mathrm{B}}{1.4})(1 - \mathrm{B})(1 - \frac{\mathrm{B}}{1.9})$$

et la variance du bruit est égale à 1.
- Calculer le polynôme d'autorégression.
- Si on essaie de simuler cette série directement à l'aide de `arima.sim()`, qu'observe-t-on ? La série obéit à un ARIMA$(2, 1, 0)$. Après avoir consulté l'aide en ligne de cette fonction, reformuler la simulation pour pouvoir utiliser `arima.sim()`.
- Simuler la série à l'aide de `simulate()`.

Exercice 7.2 (Simulation d'un SARMA)
On veut simuler une série obéissant à (1.3).
- Tirer d'abord 290 observations i.i.d. suivant la loi de z_t.
- Simuler d'après cette série une série de 240 valeurs obéissant à (1.3).

7.4 Construction de séries subissant une intervention

Une intervention est une action brutale, un choc, sur le niveau moyen d'une série. Schématiquement cette action peut avoir :

1. un effet ponctuel : à une certaine date le niveau moyen fait un saut et revient à la valeur habituelle à la date suivante ;
2. un effet brutal qui s'atténue progressivement : à une certaine date le niveau moyen fait un saut et revient progressivement à la valeur habituelle ;
3. un effet brutal et durable : le niveau moyen fait un saut et reste à cette nouvelle valeur ;
4. un effet brutal complété par un effet progressif : il y a d'abord une variation brutale de la moyenne qui atteint ensuite progressivement une nouvelle valeur d'équilibre.

On fait par ailleurs l'hypothèse que la série est la somme d'une fonction du temps qui décrit notamment ces interventions et d'une série stationnaire de moyenne nulle dont le modèle ne change pas au cours de la période étudiée. En somme la série suit un modèle ARMAX avec une moyenne qui a une dynamique.
Nous allons maintenant donner une expression quantitative aux quatre situations ci-dessus. Ensuite nous simulerons un tel mécanisme et nous estimerons le modèle.

7.4.1 Réponses typiques à une intervention

Notons t_0 la date du choc. Une intervention ponctuelle est représentée par la fonction impulsion :

$$P_t^{t_0} = \begin{cases} 1 & \text{si } t = t_0 \\ 0 & \text{si } t \neq t_0. \end{cases}$$

Une intervention qui dure est représentée par la fonction échelon :

$$S_t^{t_0} = \begin{cases} 0 & \text{si } t < t_0 \\ 1 & \text{si } t \geq t_0. \end{cases}$$

On observe que :

$$P_t^{t_0} = S_t^{t_0} - S_{t-1}^{t_0} = (1 - \mathrm{B}) S_t^{t_0}.$$

Classiquement on peut schématiser l'effet d'une intervention ponctuelle en t_0, d'abord important puis qui tend progressivement vers 0, par une suite géométrique : $1, \delta, \delta^2, \cdots$ où $\delta \in (0, 1)$ est un facteur d'amortissement. Comme l'effet est nul avant t_0, on peut décrire cet effet sur toute la période d'observation par la fonction :

$$\omega \left(1 + \delta \mathrm{B} + \delta^2 \mathrm{B}^2 + \cdots\right) P_t^{t_0} = \frac{\omega}{1 - \delta \mathrm{B}} P_t^{t_0} \tag{7.6}$$

où ω est positif ou négatif suivant le sens de l'intervention. De même un effet qui augmente progressivement à partir de t_0 peut être schématisé par

$$\omega\,(1+\delta \mathrm{B}+\delta^2\mathrm{B}^2+\cdots)S_t^{t_0}=\frac{\omega}{1-\delta \mathrm{B}}S_t^{t_0} \tag{7.7}$$

Le tableau 7.2 illustre ces effets avec $\omega = 1$.

Tableau 7.2 – Effets progressifs, par $1/(1-\delta \mathrm{B})$, d'une impulsion et d'un échelon.

instant	t_0-1	t_0	t_0+1	t_0+2	t_0+3	$\cdots$
effet sur $P_t^{t_0}$	0	1	δ	δ^2	δ^3	$\cdots$
effet sur $S_t^{t_0}$	0	1	$1+\delta$	$1+\delta+\delta^2$	$1+\delta+\delta^2+\delta^3$	$\cdots$

Notons que les mécanismes déterministes (7.6 ou 7.7) sont exactement ceux d'une autorégression, appliqués à un échelon ou une impulsion et non à un bruit blanc.

Exemple 7.5 Calculons l'effet d'une intervention qui commence en $t_0 = 3$ par une impulsion de 4 et qui s'amortit ensuite avec un facteur d'amortissement de .8 sur une série de 20 dates consécutives. Nous posons donc $\omega = 4$ et $\delta = .8$. Il nous faut aussi définir une fonction dans R dont les arguments sont une suite $1, 2 \cdots, n$, et t0 la date dans la suite où a lieu l'impulsion et qui renvoie l'impulsion, c'est-à-dire une suite de n termes tous nuls, sauf celui d'indice t0. La fonction `imp.fun()` ci-dessous fait ce travail. Il nous faut enfin décrire (7.6) avec le vocabulaire de la fonction `ARMA()`. Nous illustrons avec $n = 7$.

```
> imp.fun=function(temps,t0){a=rep(0,length(temps));a[t0]=1;a}
> t0=3; dates=1:20; ldat=length(dates)
> y.imp=imp.fun(dates,t0)
> delta=.8; omega=4; vecauto=c(1,-delta)
> AR=array(vecauto,c(length(vecauto),1,1))
> vecma=1; MA=array(vecma,c(length(vecma),1,1))
> mod1=ARMA(A=AR,B=MA)
> y1=as.vector(simulate(mod1,y0=0,noise=as.matrix(omega*y.imp),
+         sampleT=ldat)$output)
> y1[1:6]
```

```
[1] 0.000 0.000 4.000 3.200 2.560 2.048
```

On observe bien l'impulsion de 4 à la date 3 qui s'amortit ensuite. Si l'effet au lieu de s'atténuer se cumule, on pourra décrire la situation par le même mécanisme mais appliqué à un échelon. Nous devons écrire d'abord la fonction qui calcule l'échelon : `ech.fun()`.

```
> ech.fun=function(temps,t0){a=rep(0,length(temps));a[temps >= t0]=1;a}
> y.ech=ech.fun(dates,t0)
> y2=as.vector(simulate(mod1,y0=0,noise=as.matrix(omega*y.ech),
+           sampleT=ldat)$output)
> y2[1:6]
```

```
[1]  0.000  0.000  4.000  7.200  9.760 11.808
```

Les effets de l'impulsion et de l'échelon sont représentés sur la figure 7.1.

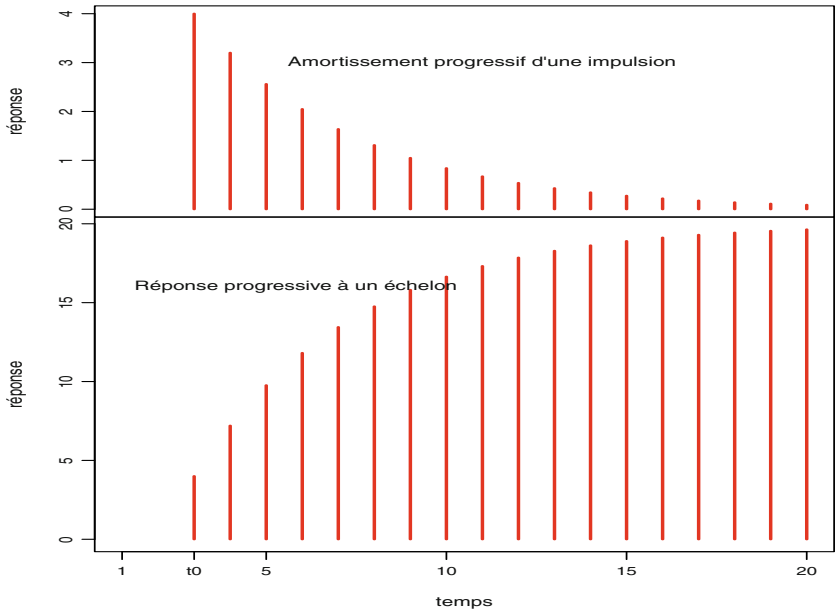

Fig. 7.1 – Action progressive d'une impulsion et d'un échelon.

Plus généralement, une intervention peut être décrite par une fraction rationnelle :

$$y_t = \frac{\omega(\mathrm{B})}{\delta(\mathrm{B})} x_t + u_t,$$

ou par une somme de fractions rationnelles :

$$y_t = \sum_{k=1}^{m} \frac{\omega_k(\mathrm{B})}{\delta_k(\mathrm{B})} x_{kt} + u_t,$$

où les $\omega_k(\mathrm{B})$ et $\delta_k(\mathrm{B})$ sont des polynômes de l'opérateur retard.
Nous avons passé en revue quelques mécanismes permettant de décrire la moyenne d'une série quand elle a une dynamique. Ces questions sont de nature purement déterministe.

7.4.2 Simulation d'une intervention

Cet exemple est inspiré d'une étude sur la piraterie aérienne (Enders, 1995). A la fin des années 60, on observe une augmentation du nombre mensuel d'actes de

piraterie aérienne aux Etats-Unis. Des portiques détecteurs de métaux sont mis en place progressivement à partir d'une date t_0 dans les différents aéroports. La mesure, mise en place de portiques, est permanente : on la schématise donc par un échelon, mais son effet, au niveau de l'ensemble des aéroports, est progressif : le nombre moyen trimestriel d'actes de piraterie décroît à mesure que le nombre d'aéroports équipés augmente, jusqu'à atteindre un niveau plus faible. L'action de l'intervention est finalement décrite par une fraction rationnelle $\frac{\omega_0}{1-\delta \mathrm{B}}$ appliquée à l'échelon $S_t^{t_0}$ avec ω_0 négatif. Pour conduire la simulation on se fixe : $\omega_0 = -2$ et $\delta = 0.6$. Par ailleurs, avant cette intervention et loin après, le nombre d'actes de piraterie est approximativement un AR(1) de paramètre $\phi = 0.8$ et de moyenne $b_0 = 3.1$, et la variance du bruit blanc est 0.55. On simule une série de 100 valeurs et l'intervention est supposée prendre place à la date $t_0 = 51$. Finalement le modèle à simuler est

$$y_t = 3.1 - \frac{2}{1-0.6\mathrm{B}} S_t^{t_0} + \frac{1}{1-0.8\mathrm{B}} z_t, \qquad z_t \sim \mathrm{BBN}(0, 0.55). \tag{7.8}$$

Pour effectuer cette simulation, on peut écrire la récurrence en multipliant à gauche et à droite par le dénominateur ou bien, et c'est ce que nous ferons, calculer la moyenne et lui superposer un AR(1). La moyenne de y_t est $3.1 - \frac{2}{1-0.6\mathrm{B}} S_t^{t_0}$, qu'on calcule à l'aide de `simulate()`.

```
> temps=1:100; t0=51
> echel=rep(1,length(temps))*(temps >= t0)
> delta=.6; omega=-2
> vecauto=c(1,-delta)
> (AR=array(vecauto,c(length(vecauto),1,1)))

, , 1
     [,1]
[1,]  1.0
[2,] -0.6

> vecma=-2
> (MA=array(vecma,c(length(vecma),1,1)))

, , 1
     [,1]
[1,]   -2

> mod2=ARMA(A=AR,B=MA)
> moy0=as.matrix(rep(3.1,100))
> moy=moy0+simulate(mod2,y0=0,noise=as.matrix(echel),sampleT=100)$output
```

On simule maintenant le processus centré de l'erreur pour différentes valeurs de l'écart type du bruit blanc en utilisant `simulate()` avec un bruit blanc en entrée.

```
> # bruit AR(1)
> set.seed(329)
> vecauto1=c(1,-.8)
```

```
> mod3=ARMA(A=array(vecauto,c(length(vecauto),1,1)),B=1)
> bruit1=simulate(mod3,y0=0,sd=.5,sampleT=100)
> bruit2=simulate(mod3,y0=0,sd=1,sampleT=100)
> bruit3=simulate(mod3,y0=0,sd=1.5,sampleT=100)
```

Enfin, on additionne moyenne et bruit pour obtenir la série.

```
> # datation de la série
> ts.temps=ts(temps,start=c(60,3),frequency=4)
> str(ts.temps)
 Time-Series [1:100] from 60.5 to 85.2: 1 2 3 4 5 6 7 8 9 10 ...
> signal1=moy+bruit1$output
> signal2=moy+bruit2$output
> signal3=moy+bruit3$output
> ser2=cbind(moy,signal1,signal2,signal3)
> colnames(ser2)=c('Moyenne','sigma=0.5','sigma=1','sigma=1.5')
> sign.br=ts(ser2,start=c(60,3),frequency=4)
```

Pour dater la série sans tâtonnement, de façon que la 51e valeur corresponde bien au 1er trimestre 73, on a créé une série contenant le temps $(1, 2, 3 \dots)$. Il est ainsi facile de vérifier notre datation.

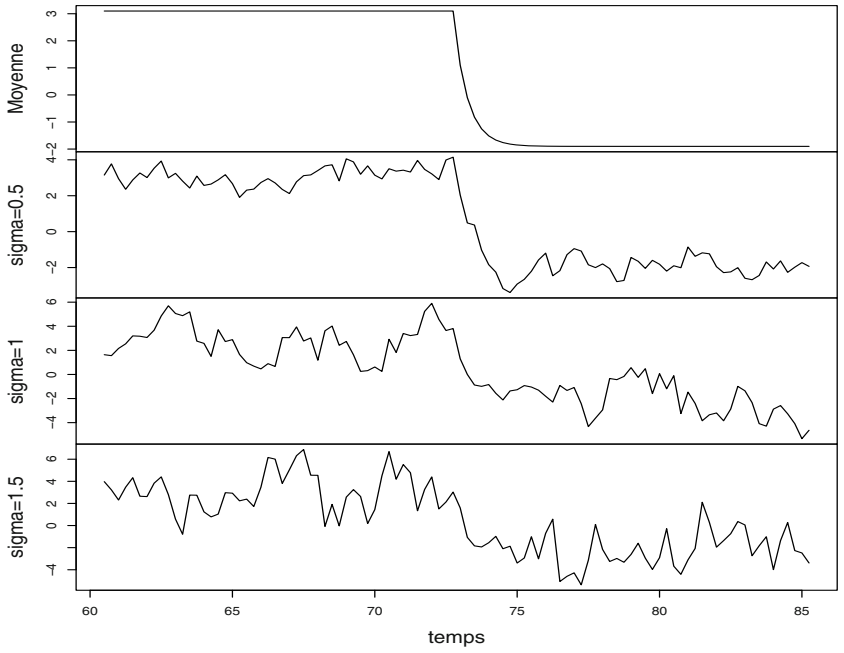

Fig. 7.2 – Intervention : moyenne de la série et trois simulations.

On peut voir (fig. 7.2) que plus l'écart type de l'erreur augmente (de haut en bas, `sd=` .5, 1, 1.5) plus il est difficile de mettre en évidence le mécanisme d'intervention.

7.4.3 Estimation d'une intervention

Remarquons d'abord la ressemblance formelle entre (7.8) et la description du mécanisme ARMA, ressemblance que nous avons exploitée pour la simulation. Pour estimer des modèles d'intervention, on utilisera `arimax()` de **TSA**. Cette fonction dispose, en plus des options habituelles de `arima()`, de `xtransf=` qui indique la matrice des données affectant la série, une colonne par variable, et de `transfer=` qui donne les ordres, à la façon d'un ARMA, de la fraction rationnelle qui décrit l'intervention.
Estimons les paramètres du modèle figurant dans l'équation (7.8) grâce à la série simulée `signal1`. Nous devons indiquer l'échelon ou l'impulsion, c'est la série `echel1` ci-dessous. Ensuite l'intervention dans (7.8), terme $-\frac{2}{1-0.6\mathrm{B}}S_t^{t_0}$ a un mécanisme AR et l'option à préciser est `transfer=list(c(1,0))`.

```
> require(TSA)
> temps=1:length(signal1)
> echel1=rep(1,length(temps))*(temps>=51)
> (pirate.m1=arimax(signal1,order=c(1,0,0),xtransf=data.frame(echel1),
+         transfer=list(c(1,0)),method='ML'))

Series: signal1
ARIMA(1,0,0) with non-zero mean
...
Coefficients:
         ar1  intercept  echel1-AR1  echel1-MA0
      0.5815     3.1421      0.6418     -1.8406
s.e.  0.0803     0.1503      0.0565      0.2874

sigma^2 estimated as 0.2084:  log likelihood = -63.68
AIC = 135.36   AICc = 136   BIC = 148.39
```

On obtient bien un résultat proche du modèle.
Les modèles d'intervention sont présentés dans Box *et al.* (1999), Wei (2006) et Cryer & Chan (2008) qu'accompagne **TSA**.

Chapitre 8

Trafic mensuel de l'aéroport de Toulouse-Blagnac

Nous nous intéressons au trafic passager mensuel de l'aéroport, c'est-à-dire à la somme des arrivées et des départs de chaque jour sur le mois. Nous étudierons en particulier l'influence des attentats de septembre 2001 sur le trafic au cours des mois suivants (nous utilisons la date abrégée 9/11 par la suite). La série couvre plus de 14 ans au cours desquels sa dynamique évolue. Aussi pourrons-nous, en fonction de notre objectif, n'en utiliser qu'une partie : pour une étude historique il est intéressant de travailler avec toute la série, mais pour une prévision à quelques mois il est préférable de ne pas se baser sur un passé trop ancien et une série de quatre ou cinq ans est largement suffisante.
L'exploration de la série permet d'en voir les traits globaux : nature de la tendance et de la saisonnalité, sans attention particulière à des questions d'autocorrélation. En revanche, pour la modélisation ou la prévision, les phénomènes d'autocorrélation d'un mois sur l'autre doivent être pris en compte. Nous commencerons par fabriquer les séries annuelles et mensuelles à partir de la série du trafic quotidien. Puis nous explorerons la série sur toute la période disponible dans la section 8.2. Nous la modéliserons ensuite avant septembre 2001, section 8.3. Le modèle obtenu nous servira à établir la prédiction de ce qu'aurait été le trafic en 2002 en l'absence des attentats du 11 septembre (section 8.4). Afin de fournir une estimation de perte de trafic non seulement mensuelle mais également sur d'autres périodes, nous simulons un grand nombre de trajectoires de trafic, d'où nous tirerons une estimation de la distribution de la perte annuelle.

8.1 Préparation des données

La série du trafic quotidien est contenue dans le fichier texte `trafquoti.txt`. La première colonne est le trafic quotidien en nombre de passagers et la deuxième, la

date sous la forme année-mois-jour. Nous chargeons **caschrono** et lisons ensuite les données, examinons leur structure et procédons à quelques vérifications :

```
> require(caschrono)
> aa=read.table(file=system.file("/import/trafquoti.txt",package="caschrono"),
+ header=FALSE,quote="",sep="",colClasses=c('numeric','character'),
+ col.names=c('trafic','date'))
> str(aa)

'data.frame':        5417 obs. of  2 variables:
 $ trafic: num  3542 9788 13204 9342 6792 ...
 $ date  : chr  "1993-01-01" "1993-01-02" "1993-01-03" "1993-01-04" ...

> summary(aa)

     trafic          date
 Min.   : 1915   Length:5417
 1st Qu.:10345   Class :character
 Median :13115   Mode  :character
 Mean   :13174
 3rd Qu.:15729
 Max.   :27231
```

Sur la période, il apparaît une moyenne de 13 174 passagers par jour. Nous voyons qu'il faut convertir la variable `date` lue comme une chaîne de caractères en dates, puis vérifier qu'il n'y a pas de jours manquants. Pour s'en assurer, on mesure la longueur de la série : elle doit être égale à la longueur d'une série de dates calculées séquentiellement, commençant et finissant aux mêmes dates que `aa` :

```
> date.1=as.Date(aa$date)
> date.1[1:10]

 [1] "1993-01-01" "1993-01-02" "1993-01-03" "1993-01-04"
 [5] "1993-01-05" "1993-01-06" "1993-01-07" "1993-01-08"
 [9] "1993-01-09" "1993-01-10"

> date.2=seq(from=as.Date("1993-01-01"),to=as.Date("2007-10-31"),by="day")
> c(length(date.1),length(date.2))

[1] 5417 5417
```

Les longueurs des deux séries de dates sont bien égales : il n'existe donc ni manquants ni doublons dans la série. Un chronogramme de la série du trafic quotidien sur les 5417 points qu'elle compte est à peu près illisible, à cause du grand nombre de points représentés et de la grande variabilité du trafic quotidien (SiteST).
Passons du trafic quotidien aux trafics annuel et mensuel. Il faut alors agréger par année puis par couple (année, mois) en gardant l'ordre initial.

Agrégation par an et par mois. Nous partons de la chaîne de caractères `aa$date`. On en extrait l'année et le mois. On agrège sur l'année pour obtenir le trafic annuel, `trafan`, qu'on exprime en milliers de passagers et qu'on convertit en une série de type `ts()`. Enfin, on enlève du trafic annuel l'observation incomplète de l'année 2007 en fabriquant une sous-série à l'aide de `window()` :

```
> an=substr(aa$date,1,4)
> mois=substr(aa$date,6,7)
> trafan=aggregate(aa$traf,list(An=an),sum)
> str(trafan)

'data.frame':        15 obs. of  2 variables:
 $ An: chr  "1993" "1994" "1995" "1996" ...
 $ x : num  3119216 3271363 3667372 4088480 4299624 ...

> trafan=ts(trafan$x/1000,start=1993,frequency=1)
> trafan.1=window(trafan,end=2006)
```

Pour obtenir le trafic mensuel, on fabrique `mois.an`, variable numérique qui augmente de valeur tous les mois et tous les ans. En agrégeant sur cette variable on obtient `trafmens`, le trafic mensuel. On l'exprime en milliers de passagers et on lui donne une structure de série temporelle.

```
> mois.an=as.numeric(paste(an,mois,sep=""))
> trafmens=aggregate(aa$traf,list(Mois.An=mois.an),sum)
> str(trafmens)

'data.frame':        178 obs. of  2 variables:
 $ Mois.An: num  199301 199302 199303 199304 199305 ...
 $ x      : num  245872 238014 263227 277991 286691 ...

> trafmensu=ts(trafmens$x/1000,start=c(1993,1),frequency=12)
```

On peut le vérifier : `mois.an` augmente de mois en mois puis d'année en année.

8.2 Exploration

8.2.1 Décomposition de la série en tendance, saisonnalité et erreur

Nous faisons une décomposition du trafic mensuel en tendance, saisonnalité et erreur à l'aide de `decompose()`. La figure 8.1 obtenue par

```
> dec.m=decompose(trafmensu)
> plot(dec.m)
> abline(v=2001.75)
```

donne de haut en bas : le trafic mensuel brut, la tendance, la composante saisonnière et l'erreur. Septembre 2001 est marqué par un trait vertical. Examinons ces chronogrammes. Le premier montre la croissance du trafic et sa saisonnalité prononcée qui rend peu lisible l'effet du 9/11. Par contre, le graphique de la tendance met en évidence la chute de trafic due au 9/11, puis la reprise progressive de la croissance par la suite. La composante saisonnière est importante mais `decompose()`, de même que `stl()`, en fournit une version « moyenne » sur la période étudiée, qui gomme toute évolution de la saisonnalité.

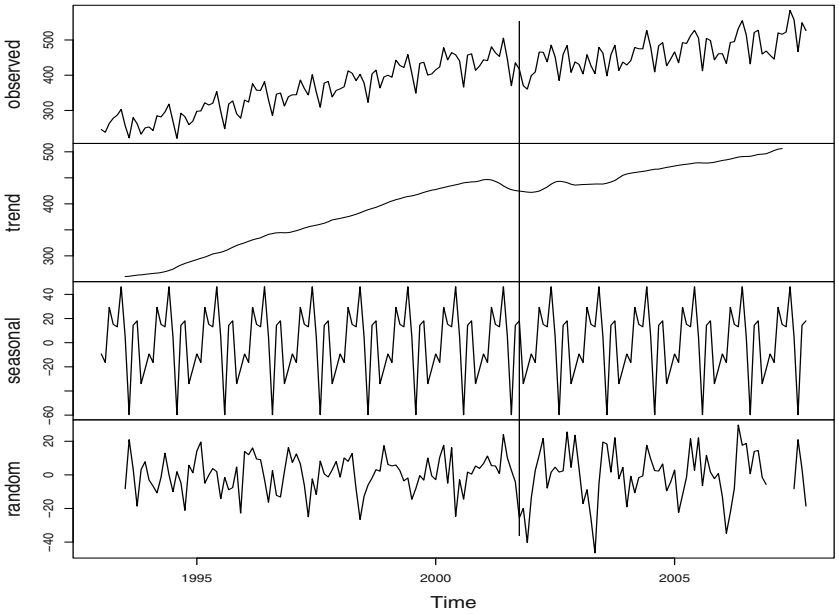

Fig. 8.1 – Trafic passager mensuel et sa décomposition additive, unité 10^3 passagers.

8.2.2 Month plot

Pour mieux comprendre la saisonnalité, nous examinons le month plot du trafic. La série étant de type `ts`, de fréquence 12, nous l'obtenons par :

```
> monthplot(trafmensu)
```

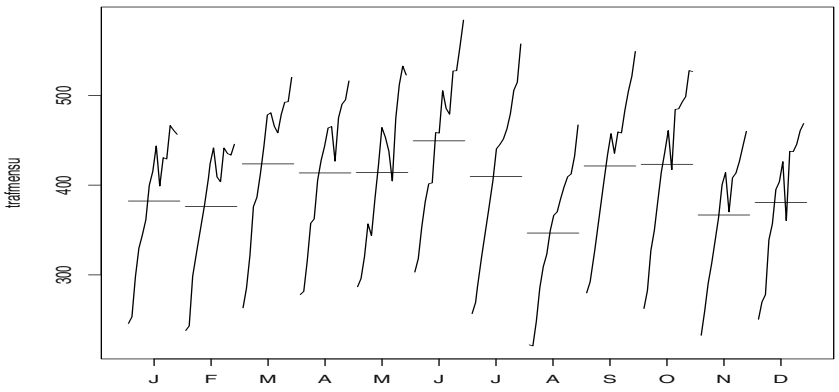

Fig. 8.2 – Month plot du trafic passager : janvier 1993-octobre 2007.

La figure 8.2 montre un diagramme séquentiel pour chacune des saisons supposées de la série. S'il n'y avait pas d'effet saisonnier, ces 12 chronogrammes se ressembleraient. On note que juin montre la plus forte activité. Janvier, février et août sont les mois de plus faible activité. Les baisses de trafic traduites par des décrochements dans les séries sont consécutives aux attentats du 9/11. On le vérifie en imprimant les années 2000 à 2002 :

```
> window(trafmensu,start=c(2000,1),end=c(2002,12))

         Jan     Feb     Mar     Apr     May     Jun
2000 415.292 423.665 478.207 443.548 464.162 457.944
2001 443.700 441.499 480.649 463.680 453.372 505.190
2002 398.975 409.142 465.646 465.236 437.930 485.439
         Jul     Aug     Sep     Oct     Nov     Dec
2000 440.436 366.272 457.318 460.735 413.933 426.097
2001 445.332 370.211 435.473 417.169 370.169 360.457
2002 451.417 385.078 459.356 484.329 408.187 437.763
```

En examinant cette sous-série, on note qu'en juillet et août il n'y a pas de baisse par rapport à l'année précédente. La décroissance du trafic a donc duré 10 mois mais on voit que la croissance a repris ensuite à un rythme plus lent. Ces graphiques ne nous renseignent pas sur un éventuel changement de la dynamique de la série après le 9/11. Il nous faut donc compléter cette exploration.

8.2.3 Lag plot

Nous avons observé, d'une part une forte saisonnalité, d'autre part un changement de régime après le 9/11. Nous essayons de nous représenter ce changement en comparant les lag plots des séries avant le 9/11 et après. Nous fabriquons ces sous-séries :

```
> trafav=window(trafmensu,start=c(1996,1),end=c(2001,8))
> traf.apr01=window(trafmensu,start=c(2001,10),end=c(2007,10))
> time.apr01=time(traf.apr01)
```

puis les lag plots pour 12 retards :

```
> lag.plot(rev(trafav),set.lags=1:12,asp=1,diag=TRUE,diag.col="red",
+     type="p",do.lines=FALSE)
> lag.plot(rev(traf.apr01),set.lags=1:12,asp=1,diag=TRUE,
+     diag.col="red",type="p",do.lines=FALSE)
```

On voit que les autocorrélations sont moins marquées sur la série après (fig. 8.4) que sur la série avant (fig. 8.3) et ce, quels que soient les retards. On peut en déduire que si la série après le 9/11 se révèle non stationnaire, alors il en sera de même pour la série avant. On peut observer également que les points sont majoritairement au-dessus de la diagonale, en particulier au retard 12. Le modèle obtenu pour cette série (8.1) expliquera cette situation.

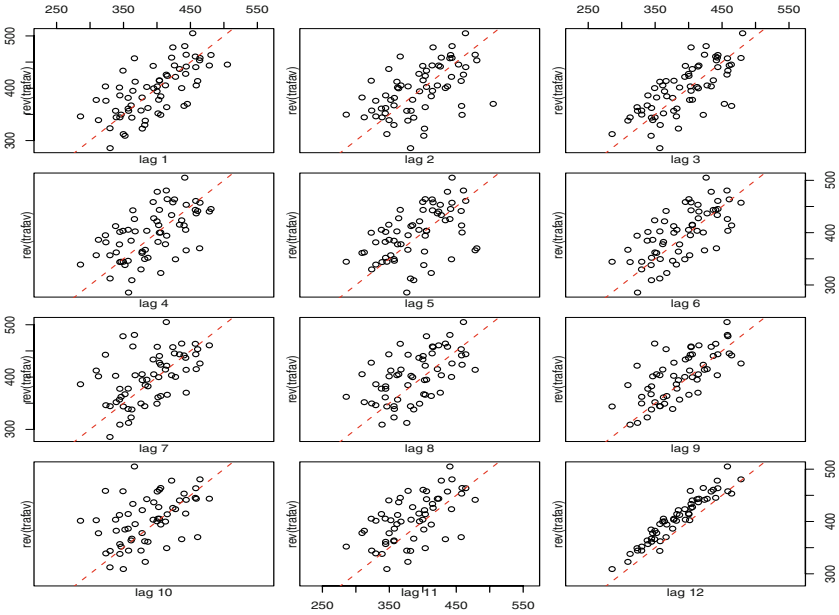

Fig. 8.3 – Lag plot du trafic passager avant le 9/11.

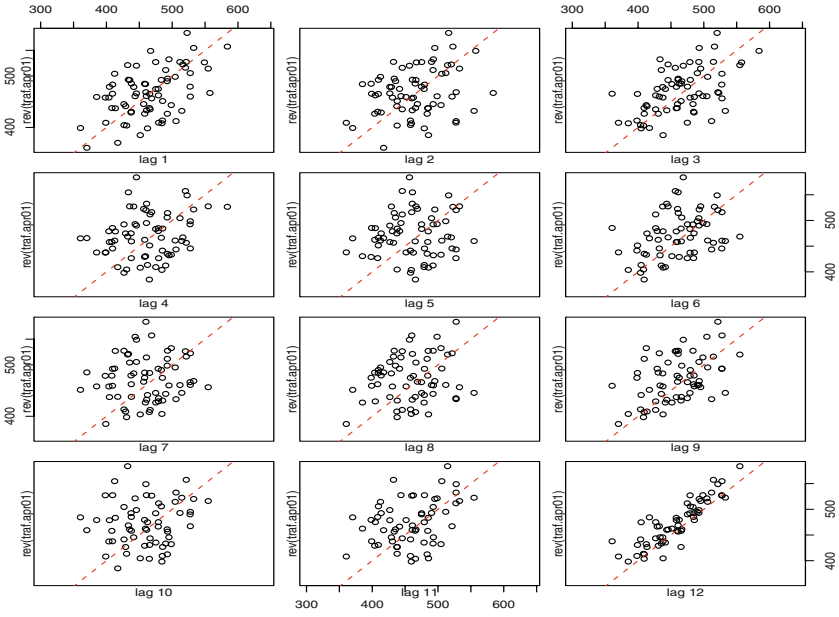

Fig. 8.4 – Lag plot du trafic passager après le 9/11.

En résumé, le trafic mensuel est une série croissante qui présente un aspect saisonnier marqué. Les attentats du 9/11 ont modifié l'évolution de cette série : tendance moins marquée après qu'avant, autocorrélations jusqu'au retard 12, moins fortes après qu'avant.
La suite de ce chapitre est consacrée d'abord à la modélisation ARIMA du trafic avant le 9/11. Après cette date, la série met plus d'un an à trouver une nouvelle régularité. Nous nous intéresserons alors à la prévision du trafic par lissage exponentiel.

8.3 Modélisation avant septembre 2001

Notre objectif est d'évaluer l'influence du 9/11 sur le volume de trafic pendant l'année 2002. Pour mesurer cet impact, nous allons d'abord modéliser la série avant cette date, prédire la série pour l'année 2002 et mesurer l'écart entre la réalisation et la prévision. Evidemment, cette démarche s'appuie sur le présupposé qu'*en l'absence de ces attentats, l'évolution de la série jusqu'en 2002 aurait obéi au même mécanisme qu'au cours des années antérieures.* Autrement dit, la série n'a pas subi d'autres influences, entre septembre 2001 et décembre 2002, susceptibles de modifier sa dynamique dans cet intervalle de temps.
Nous allons modéliser la série sur une période qui finit en août 2001. Il n'est pas pertinent de prendre une trop longue série, car son modèle évolue vraisemblablement. Nous choisissons de modéliser la sous-série de janvier 1996 à août 2001.

Normalité. Avant de commencer, nous vérifions sa normalité, par la version omnibus du test de D'Agostino :

```
> require(fBasics)
> aa=dagoTest(trafav)
> aa@test$p.value[1]

Omnibus  Test
    0.2693348
```

La p-value est élevée, il n'y a donc pas de raison de rejeter l'hypothèse de normalité de la série.

Première tentative. Le trafic avant (fig. 8.5 haut) montre une tendance croissante. La série est manifestement non stationnaire, au moins par la présence d'une tendance qui peut être déterministe ; ici elle serait représentée par une droite. D'autre part, sur le lag plot avant, on voit que l'autocorrélation empirique au retard 12 est bien supérieure à l'autocorrélation au retard 1. Ceci suggère que la non-stationnarité est d'abord de nature saisonnière. Mais essayons d'abord d'ajuster une tendance linéaire déterministe à la série. Nous régressons la série sur le temps et conservons les résidus comme série temporelle :

```
> temps=time(trafav)
> (mod1=lm(trafav~temps))
```

```
Call:
lm(formula = trafav ~ temps)

Coefficients:
(Intercept)        temps
  -45808.72        23.12
```

```
> resid= ts(residuals(mod1),start=c(1996,1),frequency=12)
```

Nous représentons en plus de la série, la série des résidus et leur ACF :

```
> plot(trafav,xlab="",ylab="trafic",xaxt="n")
> plot(resid,xlab="temps",ylab="résidu MCO")
> abline(h=0)
> acf(resid,lag.max=50, xlab="retard",main="")
```

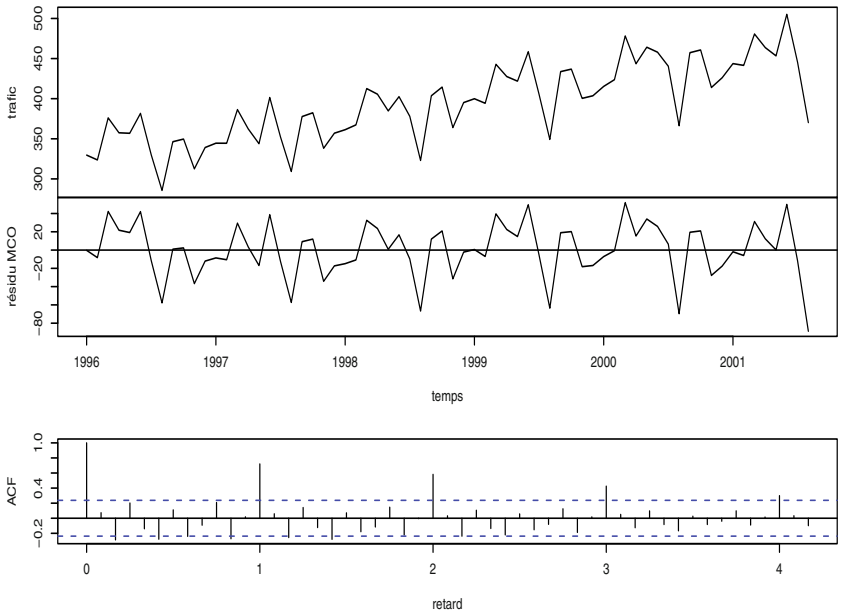

Fig. 8.5 – Ajustement d'une tendance linéaire : résidus et ACF des résidus.

On note (fig. 8.5 milieu) que la tendance a bien été prise en compte. Par contre, la décroissance de l'ACF de 12 en 12 est assez lente, symptôme de non-stationnarité dans la saisonnalité. De plus, le graphique du résidu montre une asymétrie par rapport à la moyenne 0.
En fait, notre observation du lag plot nous avait préparé à cette impasse. C'est bien la saisonnalité qui est le trait majeur de cette série. Comme le chronogramme de la série ne montre pas de régularité, au contraire, par exemple, de celui de `nottem` (fig. 1.8), nous pouvons conclure que la modélisation de cette série passe par une différenciation saisonnière.

8.3.1 Modélisation manuelle

Nous examinons donc la série différenciée saisonnièrement `diff(trafav,12)`, ainsi que ses ACF et PACF. On observe (fig. 8.6, graphique supérieur) que cette série n'est pas de moyenne nulle : il faut donc introduire une dérive dans le modèle de `trafav` (voir au chapitre 5 la discussion autour de 5.1).

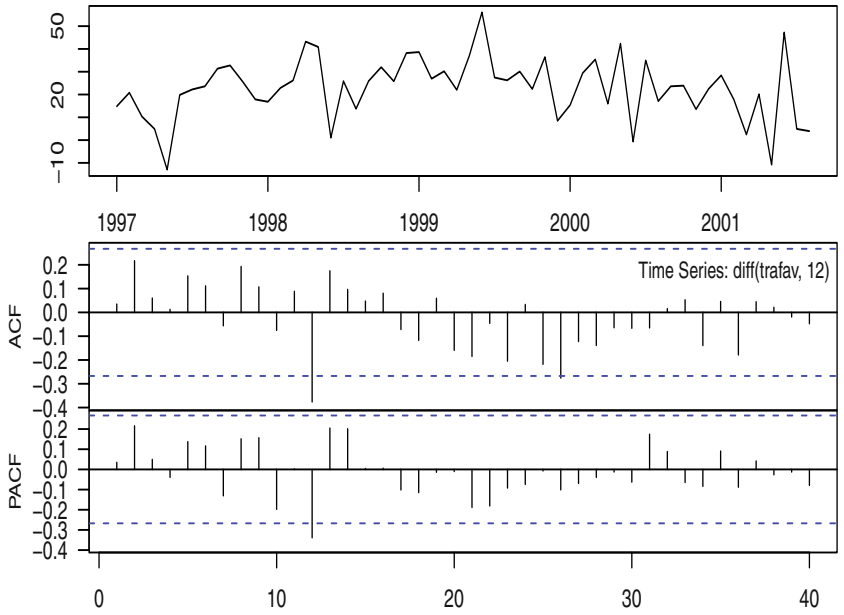

Fig. 8.6 – Trafic différencié saisonnièrement, chronogramme et ACF.

Sur les graphiques inférieurs, on note que les autocorrélations sont significatives principalement au retard 12. De plus, la PACF semble s'atténuer un peu plus rapidement que l'ACF après 12. On privilégie donc un autorégressif saisonnier d'ordre 1 :

```
> (mod2=Arima(trafav,seasonal=list(order=c(1,1,0),period=12),
+   include.drift=TRUE))

Series: trafav
ARIMA(0,0,0)(1,1,0)[12] with drift
...
Coefficients:
         sar1   drift
      -0.5417  1.9790
s.e.   0.1253  0.0919

sigma^2 estimated as 133.8:  log likelihood = -218.64
AIC = 443.27   AICc = 443.74   BIC = 449.35
```

On constate (fig. 8.7) qu'il reste encore de l'autocorrélation significative au retard 2 et à des retards de l'ordre de 24.
Commençons par traiter l'autocorrélation au retard 2 en introduisant des termes autorégressifs jusqu'à l'ordre 2.

```
> (mod3=Arima(trafav,order=c(2,0,0),seasonal=list(order=c(1,1,0),period=12),
+  method='ML',include.drift=TRUE))

Series: trafav
ARIMA(2,0,0)(1,1,0)[12] with drift
...
Coefficients:
         ar1     ar2     sar1   drift
      0.1958  0.3459  -0.6464  1.9341
s.e.  0.1317  0.1322   0.1066  0.1646

sigma^2 estimated as 107.3:  log likelihood = -213.79
AIC = 437.59   AICc = 438.79   BIC = 447.72
```

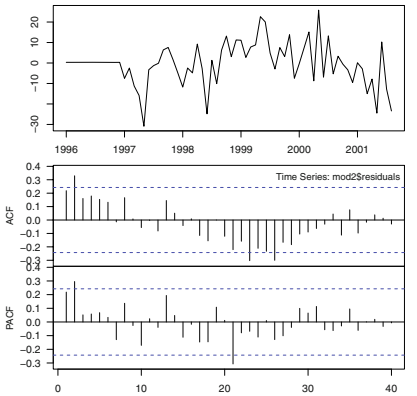

Fig. 8.7 – SARIMA(0,0,0)(1,1,0) : résidu, ACF et PACF.

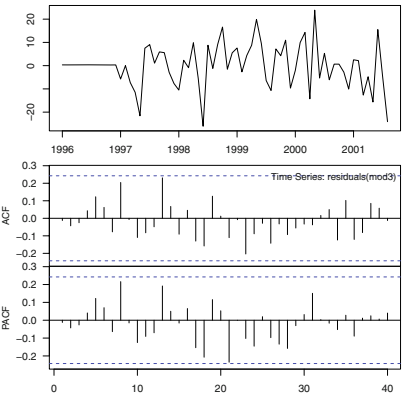

Fig. 8.8 – SARIMA(2,0,0)(1,1,0) : résidu, ACF et PACF.

Les résidus ne montrent que peu d'autocorrélation (fig. 8.8) :

```
> xy.acfb(residuals(mod3),numer=FALSE)
```

Le test de blancheur (cf. section 4.1.2) :

```
> ret= seq(6,30,6)
> t(Box.test.2(residuals(mod3),ret,type="Ljung-Box",fitdf=4,decim=4))

          [,1]    [,2]    [,3]    [,4]    [,5]
Retard  6.0000 12.0000 18.0000 24.0000 30.0000
p-value 0.9531  0.9012  0.7446  0.7501  0.8746
```

donne des résultats très satisfaisants. On voit également que le coefficient ar1 est du même ordre que son écart type : il n'est donc pas très significatif. On peut le vérifier :

```
> t_stat(mod3)

            ar1      ar2      sar1    drift
t.stat 1.486091 2.617070 -6.063581 11.74794
p.val  0.137255 0.008869  0.000000  0.00000
```

On supprime donc le terme d'autorégression d'ordre 1, en le contraignant à 0, on re-estime le modèle et on examine la blancheur du résidu.

```
> mod4=Arima(trafav,order=c(2,0,0),seasonal=list(order= c(1,1,0),period=12),
+   fixed=c(0,NA,NA,NA),method='ML',include.drift=TRUE)
> summary(mod4)

Series: trafav
ARIMA(2,0,0)(1,1,0)[12] with drift
...
Coefficients:
      ar1     ar2     sar1   drift
        0  0.3819  -0.6168  1.9533
s.e.    0  0.1339   0.1128  0.1279

sigma^2 estimated as 113.1:  log likelihood = -214.88
AIC = 437.76   AICc = 438.96   BIC = 447.88

In-sample error measures:
         ME        RMSE         MAE         MPE        MAPE
-0.13085828  9.65187341  6.72840847 -0.09643903  1.65978397
       MASE
 0.21837355

> ret6= seq(6,30,6)
> t(Box.test.2(residuals(mod4),ret6,type="Ljung-Box",fitdf=3,decim=4))

          [,1]    [,2]    [,3]    [,4]    [,5]
Retard  6.0000 12.0000 18.0000 24.0000 30.0000
p-value 0.6674  0.7823  0.6169  0.5379  0.5932
```

La suppression de l'autorégression sur l'ordre 1 n'a pas diminué la qualité de l'ajustement. Tous les coefficients sont significatifs :

```
> t_stat(mod4)

            ar2      sar1    drift
t.stat 2.853136 -5.468181 15.27072
p.val  0.004329  0.000000  0.00000
```

Le modèle finalement retenu est

$$\begin{aligned}(1-\mathrm{B}^{12})y_t &= 23.44 + u_t,\ t = 1, \cdots, 168 \\ u_t &= \frac{1}{(1-0.3819\mathrm{B}^2)(1+0.6168\mathrm{B}^{12})} z_t, \quad z_t \sim \mathrm{BBN}(0, 113.1).\end{aligned} \tag{8.1}$$

8.3.2 Modélisation automatique

La modélisation précédente ayant été assez laborieusement obtenue, considérons ce que propose une procédure automatique de sélection de modèle ARIMA pour cette série. Précisément, comparons notre estimation à celle proposée par `auto.arima()` de **forecast**. L'aide d'`auto.arima()` nous apprend que cette fonction cherche le meilleur modèle ARIMA suivant un des trois critères d'information AIC, AICc où BIC. La fonction cherche parmi les modèles possibles sous les contraintes d'ordre qu'on indique.
Le paramètre d est l'ordre de différenciation simple. S'il n'est pas précisé, le test de KPSS en choisit une valeur.
Le paramètre D est l'ordre de différenciation saisonnière. S'il n'est pas précisé, le test de Canova-Hansen en choisit une valeur. Ce test est une adaptation du test de KPSS à une série présentant une saisonnalité. L'hypothèse nulle y est : « la série a une composante saisonnière déterministe » et l'alternative : « la série présente une racine unité saisonnière ». Hyndman & Khandakar (2008) donnent quelques détails sur la procédure.
Il faut ensuite donner des ordres maximum d'autorégression et de moyenne mobile, simples et saisonnières. Par défaut ces ordres maximum sont max.p = 5, max.q = 5 et pour les termes saisonniers max.P = 2, max.Q = 2. Notre connaissance de la série nous montre que ces maximums sont satisfaisants. De plus nous savons qu'il faut différencier saisonnièrement la série. Enfin, retenons le critère AIC. L'appel d'`auto.arima()` ci-dessous sélectionne le meilleur modèle pour ce critère.

```
> best.av = auto.arima(trafav, D = 1)
> summary(best.av)

Series: trafav
ARIMA(1,0,1)(0,1,1)[12] with drift

Call: auto.arima(x = trafav, D = 1)

Coefficients:
         ar1      ma1     sma1   drift
      0.9308  -0.7845  -0.8768  1.8925
s.e.  0.0980   0.1130   0.5390  0.1554

sigma^2 estimated as 95.51:  log likelihood = -214.15
AIC = 438.31   AICc = 439.51   BIC = 448.43

In-sample error measures:
         ME        RMSE         MAE         MPE        MAPE
 0.12853310  8.87000156  6.39638588 -0.03309488  1.57803631
       MASE
 0.20759761

> t(Box.test.2(residuals(best.av),ret6,type="Ljung-Box",decim=4,fitdf=4))
```

```
          [,1]    [,2]    [,3]    [,4]    [,5]
Retard  6.0000 12.0000 18.0000 24.0000 30.0000
p-value 0.8591  0.8735  0.8933  0.8718  0.9302
```

```
> t_stat(best.av)
```

```
            ar1       ma1      sma1    drift
t.stat 9.493126 -6.941877 -1.626687 12.17682
p.val  0.000000  0.000000  0.103804  0.00000
```

Le modèle obtenu s'écrit :

$$\begin{aligned}(1-\mathrm{B}^{12})y_t &= 22.7102 + u_t,\ t = 1, \cdots, 168 \\ u_t &= \frac{(1-0.7845\,\mathrm{B})(1-0.8768\,\mathrm{B}^{12})}{1-0.9308\,\mathrm{B}} z_t, \qquad z_t \sim \mathrm{BBN}(0, 95.513).\end{aligned} \tag{8.2}$$

Comparons les deux modèles. Tout d'abord, curieusement, l'AIC est plus fort (438.31) dans le modèle censé minimiser ce critère que dans le modèle ajusté manuellement (437.76) ; les valeurs sont toutefois très proches. Ensuite, dans le modèle automatique, la variance du bruit est plus faible (95.5 contre 113) et les p-value de la statistique Ljung-Box plus fortes que dans le modèle ajusté manuellement. Les mesures d'erreur intra-échantillon sont à l'avantage du modèle automatique. La p-value, de l'ordre de 10% pour le terme MA saisonnier, est assez élevée et nous interroge sur sa significativité.

Remarques

– L'estimation 1.8925 est notée `drift` dans la sortie. C'est la quantité qui s'ajoute à y_t à chaque période et, tous les 12 mois, c'est bien la quantité $12 \times 1.8925 = 22.7102$ qui s'ajoute à y_t (cf. section 8.6).

– La dérive, positive, s'observe sur le lag plot (fig. 8.3) : les points, particulièrement au décalage 12, sont majoritairement au-dessus de la diagonale.

– Les valeurs ajustées sont données par `fitted(best.av)`. Ce sont les prévisions à l'horizon 1, sur la période ayant servi à l'estimation. On a d'ailleurs :

```
fitted(best.av)+best.av$residuals=trafav
```

– Le paragraphe suivant sera consacré à la mesure de l'impact des attentats du 11 septembre 2001 sur le trafic passager. L'utilisation de `best.av` donnant de mauvais résultats pour cette étude, nous poursuivons le travail d'après `mod4`.

Pour visualiser la qualité de l'ajustement `mod4`, équation 8.1, superposons la série et sa bande de prédiction à 80 % (fig. 8.9).

```
> ec80=mod4$sigma2^.5*qnorm(0.90)
> vajust=fitted(mod4)
> matri=as.ts(cbind(trafav,vajust-ec80,vajust+ec80),start=c(1996,1),
+    frequency=12)
> plot(matri,plot.type='single',lty=c(1,2,2),xlab="temps",ylab='trafic',
+  main="",cex.main=0.8)
> legend(par("usr")[1], par("usr")[4],
+        c("Valeur observée","Bande de prédiction"),lwd=1,lty=c(1,2))
```

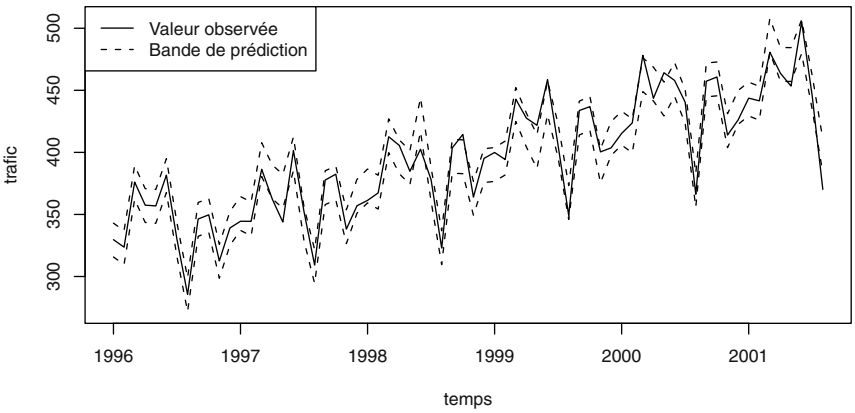

Fig. 8.9 – Trafic avant septembre 2001 - série et bande de prédiction à 80%.

La bande est assez étroite et la proportion de points contenus dans cette bande semble raisonnable. Calculons-la :

```
> indi=(trafav-(vajust-ec80))>0&(vajust+ec80-trafav)>0
> prop=100*sum(indi)/length(indi)
```

La proportion observée vaut 85%, elle est un peu supérieure à la valeur théorique de 80%. On considérera que l'ajustement est satisfaisant. Une proportion trop inférieure à 80% indiquerait un mauvais ajustement et une proportion trop supérieure, un ajustement qui suivrait trop bien les données, par exemple par excès de paramètres.

Effectuons un test ADF dont l'hypothèse nulle est : la série `diff(trafav,12)` est non-stationnaire.

```
> require(urca)
> ur1=ur.df(diff(trafav,12),lags=1,type="drift")
> summary(ur1)

###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + z.diff.lag)

Residuals:
     Min       1Q   Median       3Q      Max
-32.2470  -7.0913   0.4016   8.6523  33.0272

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  16.7059     4.9777   3.356 0.001499 **
z.lag.1      -0.7344     0.1978  -3.714 0.000507 ***
z.diff.lag   -0.2316     0.1408  -1.645 0.106114
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.75 on 51 degrees of freedom
Multiple R-squared: 0.5005,        Adjusted R-squared: 0.4809
F-statistic: 25.55 on 2 and 51 DF,  p-value: 2.056e-08

Value of test-statistic is: -3.7136 6.921

Critical values for test statistics:
      1pct  5pct 10pct
tau2 -3.51 -2.89 -2.58
phi1  6.70  4.71  3.86
```

Les p-values ne dépassent pas 1%, donc on rejette l'hypothèse de non-stationnarité. De façon convergente, un test de KPSS dont l'hypothèse nulle est : « la série est stationnaire », conclut à la stationnarité de la série différenciée saisonnièrement :

```
> kpss.test(diff(trafav,12))

        KPSS Test for Level Stationarity

data:  diff(trafav, 12)
KPSS Level = 0.3097, Truncation lag parameter = 1,
p-value = 0.1
```

Cette p-value de 10% conduit à ne pas rejeter l'hypothèse (nulle) de stationnarité.

Exercice 8.1
Vérifier qu'il n'est pas nécessaire de régresser sur un retard supérieur à 1 dans le test ADF ci-dessus, obtenu par `ur1 = ur.df(diff(trafav,12),lags=1,type=drift")`".

8.4 Impact sur le volume de trafic

Pour la prévision, nous nous appuyons sur le modèle obtenu manuellement.

8.4.1 Prévision ponctuelle

La série s'arrêtant en août 2001, la prévision pour chaque mois de 2002 s'obtient par `forecast()` avec un horizon de 16 mois.

```
> prev2002=forecast(mod4,h=16,level=80)
```

Si on examine la structure de la sortie par `str(prev2002)`, on voit que `forecast()` produit une liste de 10 éléments dont certains sont eux-mêmes des listes. L'élément `mean` de type `Time-Series` est la prévision ponctuelle, c'est-à-dire la moyenne

conditionnelle au passé arrêté à août 2001 et les limites de la bande de prédiction sont données par `lower` et `upper`. Superposons maintenant la réalisation et la prédiction du trafic :

```
> apredire=window(trafmensu,start=c(2001,9),end=c(2002,12))
> matri2=cbind(apredire,prev2002$lower,prev2002$upper,prev2002$mean)
> plot(matri2,plot.type='single',lty=c(1,2,2,3),
+   xlab="temps",xaxt='n',ylab='trafic')
> axis(1,at=c(2001.7,2002,2002.9),
+   labels=c("sept-2001","jan-2002","dec-2002"),cex.axis=.8)
> legend("topleft",c("Valeur obs.","Bande de préd. à 80%","Préd. moyenne"),
+   lwd=1, lty=c(1,2,3),cex=.8 )
```

La prévision ponctuelle de l'année 2002 est donnée par :

```
> pr2002=prev2002$mean[5:16]
```

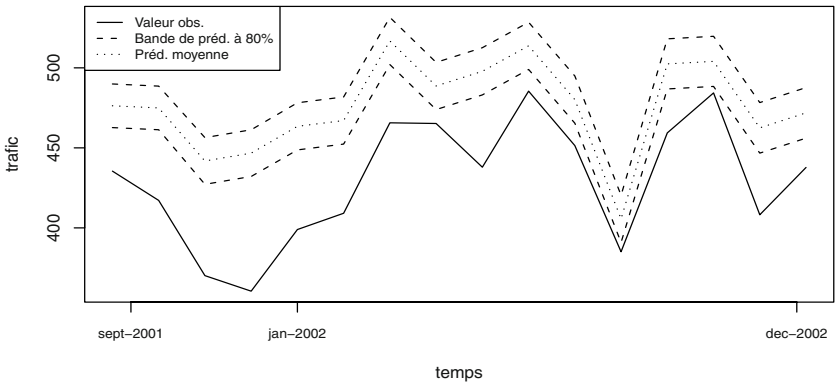

Fig. 8.10 – Trafic de septembre 2001 à fin 2002 - série et bande de prédiction à 80%.

On note (fig. 8.10) que la prévision est constamment supérieure à la réalisation et, surtout, que la bande de prévision à 80% ne contient jamais la série. Il est manifeste que le modèle `mod4` ne convient pas à partir de septembre 2001. Nous allons quantifier cette observation en calculant pour chaque mois, l'ordre quantile de la réalisation pour la loi de la prévision. Un ordre anormalement éloigné de 50% indique que la valeur observée est peu compatible avec le modèle qui a donné la prévision.

Appelons y_0 la réalisation un certain mois. Pour le même mois, nous avons la loi conditionnelle au passé de la prédiction : $\mathcal{N}(\mu, \sigma^2)$. L'ordre quantile de y_0 pour cette loi est $F(\frac{y_0-\mu}{\sigma})$ où F désigne la fonction de répartition d'une v.a. $\mathcal{N}(0,1)$. D'autre part, l'estimation de la moyenne μ est fournie en sortie de `forecast()`. Par contre, l'écart type σ n'est pas directement fourni dans les sorties de `forecast()`. On peut l'obtenir à partir de la borne supérieure de l'intervalle de prédiction de niveau 80% de la façon suivante. Si $q(\alpha)$ désigne le quantile d'ordre α d'une variable $\mathcal{N}(0,1)$, la borne supérieure de l'intervalle de prédiction à 80% est : $\mu +$

$q(0.9)\sigma$ d'où l'on tire l'estimation de σ pour chaque horizon. `prev2002$upper` et `prev2002$mean` contiennent respectivement les moyennes et les bornes supérieures d'où l'on déduit les écarts types. On trouve ensuite l'ordre quantile de toute valeur y_0. Le code nécessaire est :

```
> sigpred=(prev2002$upper-prev2002$mean)/qnorm(.9)
> ordres.q=pnorm((apredire-prev2002$mean)/sigpred)
> round(100*ordres.q,digits=2)
```

```
      Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
2001                                                0.01 0.00 0.00 0.00
2002 0.00 0.00 0.00 2.09 0.00 0.69 0.62 3.72 0.02 5.32 0.00 0.28
```

Aucun ordre n'est supérieur à 5.32%, la série observée en 2002 est très inférieure à sa prédiction. Le modèle ajusté avant septembre 2001 n'est plus valable ensuite. La fonction `forecast()` fournit pour chaque horizon la prévision moyenne, c'est-à-dire la moyenne conditionnelle au passé. Sur le graphique, la distance verticale entre la prédiction de la valeur pour un mois et la réalisation est la perte moyenne pour ce mois. Mais, une fois le modèle estimé, il nous est possible de simuler un grand nombre de trajectoires de la série pour l'année 2002 ; de calculer pour chaque simulation la perte par mois ou pour l'année et enfin d'obtenir une distribution de la perte. C'est ce que nous envisageons maintenant.

8.4.2 Simulation de trajectoires

Nous considérons que la réalisation de la série en 2002 est une donnée et la différence, prévision moins réalisation pour un certain mois, est une mesure ponctuelle de la perte. Si l'on voulait évaluer la perte pour le premier trimestre 2002 et en donner une mesure de précision, il faudrait : prédire la perte totale pour ce trimestre en additionnant les pertes mensuelles et, sous l'hypothèse de normalité de la prévision, estimer la variance de la perte du trimestre. On en déduirait ensuite un intervalle de prédiction de la perte. Mais les prévisions de chaque mois ne sont pas indépendantes et le calcul de cette variance peut être compliqué.
Il est plus simple de recourir à des simulations pour fabriquer un grand nombre de trajectoires, calculer pour chacune la perte par mois, éventuellement la perte par trimestre, par an... On dispose ainsi d'un échantillon de vecteurs de pertes indépendants pour la période qui nous intéresse, échantillon sur lequel nous pourrons estimer la perte moyenne, représenter un histogramme de cette perte...

Technique de simulation

Nous devons simuler des trajectoires de longueur 16 à partir du modèle (8.1) et en utilisant comme valeurs initiales les valeurs de la série observée. Comme la prise en compte des valeurs initiales n'est pas très bien documentée dans l'aide de `simulate()`, nous allons écrire directement le code de l'initialisation (cf. l'expression 7.5 et la discussion qui suit).

D'abord nous devons exprimer y_t en fonction de son passé, sans utiliser l'opérateur retard. Partant de (8.1), nous multiplions des deux côtés par le dénominateur, nous obtenons ainsi

$$(1 - \mathrm{B}^{12})(1 - \phi_2 \mathrm{B}^2)(1 - \phi_{12}\mathrm{B}^{12})y_t = \texttt{cte} + z_t. \tag{8.3}$$

Le polynôme d'autorégression de la série différenciée est :

```
> require(polynom)
> poly.ar=polynomial(coef=c(1,0,-mod4$coef[2]))*
+ polynomial(coef=c(1,rep(0,11),-mod4$coef[3]))
```

Le terme `cte` dans (8.3) est $(1 - \phi_2)(1 - \phi_{12}) \times 12 \times$ `drift` :

```
> cte=predict(poly.ar,1)*12*mod4$coef[4]
```

Observons ici que `predict()`, appliquée à un polynôme, l'évalue en son deuxième argument. Pour écrire l'équation de récurrence qui exprime y en fonction de son passé, nous effectuons le produit de `poly.ar` par le terme qui comporte la racine unité saisonnière, $(1 - \mathrm{B}^{12})$:

```
> fac.rsaiso=polynomial(coef=c(1,rep(0,11),-1))
> coef.yg=fac.rsaiso*poly.ar
```

Le polynôme de la partie MA est réduit au terme de degré 0.

```
> poly.ma=1
```

Le vecteur `coef.yg` allant jusqu'au retard 26, il nous faut 26 valeurs initiales : le trafic de juillet 1999 à août 2001, `traf.ini` ci-dessous. Nous sommes maintenant en mesure d'écrire le code de la simulation.
Nous devons encore préciser le nombre de simulations `nsim` et l'horizon `hori`. Nous définissons le modèle par `ARMA()` et considérons la constante additive `cte` comme le coefficient d'un input constant, égal à 1, `entree`.

```
> require(dse)
> traf.ini=window(trafmensu,start=c(1999,7),end=c(2001,8))
> y0=traf.ini; nsim=10000; hori=16
> ysim=matrix(0,ncol=nsim,nrow=hori)
> set.seed(347) # choix d'une graine
> bb=matrix(rnorm(nsim*hori,sd=mod4$sigma2^.5),nrow=hori,ncol=nsim)
> AR=array(as.vector(coef.yg),c(length(coef.yg),1,1))
> Cte=array(cte,c(1,1,1))
> entree=as.matrix(rep(1,hori))
> BM=array(as.vector(poly.ma),c(length(poly.ma),1,1))
> mod.sim=ARMA(A=AR,B=BM,C=Cte)
```

Nous pouvons maintenant effectuer les simulations.

```
> for (sim in 1:nsim){
+  ysim[,sim]=simulate(mod.sim,y0=rev(y0),input=entree,
+      sampleT=hori,noise=as.matrix(bb[,sim]))$output}
```

A titre de vérification, comparons la moyenne et l'écart type théoriques de la prévision fournies par `forecast()` avec leurs versions empiriques calculées sur les trajectoires simulées.
Le tableau `ysim` contient les trajectoires simulées de septembre 2001 à décembre 2002. L'année 2002 correspond aux lignes 5 à 16. Par `apply(ysim[5:16,],1,mean)` on calcule la moyenne par mois sur l'ensemble des simulations. On calcule également la variance par mois des simulations et on extrait l'écart type. Par `rbind()` on empile les moments empiriques et théoriques pour obtenir la matrice `compar` et on en imprime les colonnes correspondant aux mois pairs dans le tableau 8.1. Nous calculons par mois les moyennes et les variances des simulations et nous groupons dans une matrice, ces moyennes, les prévisions obtenues par la modélisation, les écarts types empiriques des simulations et les écarts types théoriques :

```
> moy.mois=apply(ysim[5:16,],1,mean)
> var.mois=apply(ysim[5:16,],1,var)
> compar=round(rbind(moy.mois,prev2002$mean[5:16],
+              var.mois^.5,sigpred[5:16]),digits=2)
```

Tableau 8.1 – Moyennes et écarts types des prévisions - simulations et calcul théorique.

	févr.	avr.	juin	août	oct.	déc.
moy. emp.	467.16	488.78	513.77	405.66	504.20	472.03
moy. théo.	467.09	488.66	513.76	405.60	504.05	471.88
é. types emp.	11.54	11.55	11.60	11.54	12.33	12.17
é. types théo.	11.49	11.50	11.51	11.51	12.22	12.32

Ce tableau montre que les deux versions théoriques et empiriques, sont très proches. L'avantage des simulations est qu'on peut calculer d'autres statistiques que les moyennes des prédictions mensuelles, comme la distribution de la perte annuelle que nous examinons maintenant.

Distribution de la perte en 2002

Calculons les pertes mensuelles simulées en 2002. Pour cela nous retranchons de chaque série simulée pour 2002 le trafic observé ; nous exprimons également ces pertes en pourcentage du trafic observé. Dans le code ci-dessous, chaque ligne de `perte.sim` correspond à un mois de 2002 et chaque colonne à une simulation. En additionnant par colonne `perte.sim` nous obtenons l'échantillon des pertes simulées annuelles pour 2002, `perte.an`. Nous fabriquons les pertes en pourcentage du trafic observé, `perte.an.pct`. Enfin nous calculons un certain nombre de quantiles de ces distributions, présentés dans le tableau 8.2.

```
> traf.02=window(trafmensu,start=c(2002,1),end=c(2002,12))
> perte.sim=ysim[5:16,]-matrix(traf.02,nrow=12,ncol=nsim)
> perte.an=apply(perte.sim,2,sum)
> perte.an.pct=100*(perte.an/sum(traf.02))
```

```
> perte= rbind(perte.an, perte.an.pct)
> q.perte=t(apply(perte,1,quantile,probs=c(0,.10,.25,.5,.75,.90,1)))
```

Tableau 8.2 – Quantiles de la distribution des pertes en valeur (milliers de passagers) et en pourcentage pour 2002.

	0%	10%	25%	50%	75%	90%	100%
perte.an	255.41	414.01	448.22	486.39	524.58	558.29	695.69
perte.an.pct	4.83	7.83	8.48	9.20	9.92	10.56	13.15

Nous pouvons voir sur le tableau 8.2 que la perte en passagers de l'année 2002 est contenue à 80% dans l'intervalle (414, 558) milliers de passagers ou que la perte médiane est légèrement supérieure à 9%.

Remarques

– Pour la simulation de trajectoire, nous avons fait des tirages indépendants dans la loi de l'erreur estimée (voir dans le code ci-dessus la fabrication de `bb`). L'erreur est supposée distribuée normalement et l'exploration du début de la section 8.3 nous conforte dans ce présupposé.

– Si la normalité n'est pas acceptable, on peut cependant estimer le modèle ARIMA en choisissant l'option `method="CSS"`(*Conditional Sum of Squares*) dans `Arima()`, voir Cryer & Chan (2008) ou section 4.5.

– Ensuite, pour les simulations, les erreurs ne pouvant pas être tirées d'une loi gaussienne, une solution consiste à les tirer par des tirages avec remise dans les résidus obtenus à l'estimation. Il suffit pour cela de remplacer dans la fabrication de `bb`, `rnorm(nsim*hori,sd=mod4$sigma2^.5)` par `sample(mod4$residuals,nsim*hori,replace=TRUE)`

8.5 Etude après le 9/11 - lissage exponentiel

Nous avons pu constater combien la modélisation du trafic par un ARIMA est longue et délicate. Qui plus est, la série après le 9/11 montre une période d'instablilité (fig. 8.11) et n'obéit à un mécanisme assez régulier qu'à partir de 2004.

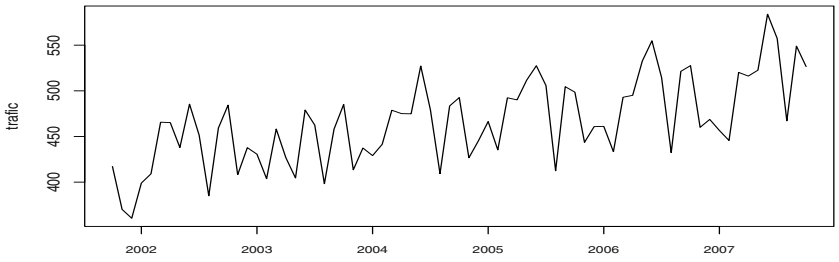

Fig. 8.11 – Trafic mensuel après le 9/11.

Il semble donc prudent, d'une part, de baser des prédictions sur un passé assez court, d'autre part, de choisir une méthode facile à utiliser. Suivant ces prescriptions, nous allons utiliser le lissage exponentiel pour prédire la série pour les 6 premiers mois de 2006 à partir des années 2004 et 2005.
Nous avons vu que la série présente une saisonnalité et que l'amplitude des mouvements saisonniers reste à peu près constante au cours du temps. Un niveau localement constant et une composante saisonnière additive semblent donc convenir. On examine alors le modèle à moyenne localement constante et saisonnalité. C'est une version simplifiée de l'équation (6.16) sans composante d'état pour la pente :

$$y_{t|t-1} = l_{t-1} + s_{t-m} \tag{8.4a}$$
$$\epsilon_t = y_t - y_{t|t-1} \tag{8.4b}$$
$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)l_{t-1} \tag{8.4c}$$
$$s_t = \gamma^*(y_t - l_t) + (1-\gamma^*)s_{t-m}. \tag{8.4d}$$

La série d'apprentissage est la suivante :

```
> traf0405=window(trafmensu,start=c(2004,1),end=c(2005,12))
```

et la série à prédire est :

```
> traf06_1_6=window(trafmensu,start=c(2006,1),end=c(2006,6))
```

Le lissage exponentiel du modèle (8.4) à moyenne localement constante et saisonnalité additive est obtenu grâce à :

```
> es.1=ets(traf0405,model="ANA")
> summary(es.1)

ETS(A,N,A)
Call:
 ets(y = traf0405, model = "ANA")

  Smoothing parameters:
    alpha = 0.3641
    gamma = 1e-04

  Initial states:
    l = 454.2696
    s = -26.4708 -43.4796 21.7431 16.7409 -61.1632 22.6946
           57.4746 24.0734 11.6141 17.9053 -25.821 -15.3115

  sigma:  8.7877

     AIC     AICc      BIC
208.5940 255.2606 225.0867

In-sample error measures:
       ME      RMSE       MAE       MPE      MAPE      MASE
3.4015165 8.7876520 7.9331761 0.7037849 1.6901215 0.2209421
```

On constate que le paramètre `alpha` n'est proche ni de 0 ni de 1. Proche de 0, il indiquerait que la moyenne est pratiquement constante ; proche de 1, il traduirait une mise à jour violente et sans doute la nécessité d'essayer un modèle avec tendance localement linéaire. Le paramètre `gamma` est très faible, mais il faut se souvenir qu'il concerne la correction de saisonnalité apportée par la dernière observation. Cette correction est nécessairement faible. La fonction `ets()` a une option de recherche automatique du meilleur modèle au sens du critère AIC. On obtient simplement ce meilleur modèle par `es.2=ets(traf0405)` ; c'est le même modèle que celui choisi précédemment par l'observation de la série. La prévision se fait normalement par `forecast()` appliquée à l'objet issu de `ets()`.

Simulation d'un modèle de lissage exponentiel

Nous avons vu, à l'occasion de l'estimation de la perte de trafic en 2002, que la simulation de trajectoires peut être très utile. Simulons maintenant des trajectoires du modèle que nous venons d'ajuster par `ets()`. D'abord, il faut écrire la représentation espace-état du modèle (8.4) :

$$\mathbf{x}_t = \begin{bmatrix} l_t \\ s_t \\ s_{t-1} \\ \vdots \\ s_{t-11} \end{bmatrix}, \underset{1\times 13}{\mathbf{w}} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 \end{bmatrix}, \mathbf{F}_{11} = \begin{bmatrix} 1 \end{bmatrix}, \underset{12\times 1}{\mathbf{F}_{21}} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$\mathbf{F}_{12} = \mathbf{F}'_{21}, \underset{12\times 12}{\mathbf{F}_{22}} = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 \\ & \ddots & & \\ 0 & \ddots & 0 & 0 \\ 0 & \cdots & 1 & 0 \end{bmatrix}, \mathbf{F} = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{21} \\ \mathbf{F}_{21} & \mathbf{F}_{22} \end{bmatrix}, \underset{1\times 13}{\mathbf{g}} = \begin{bmatrix} \alpha \\ \gamma \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

la représentation espace-état est alors :

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{g}\epsilon_t \tag{8.5a}$$

$$y_t = \mathbf{w}\mathbf{x}_{t-1} + \epsilon_t. \tag{8.5b}$$

Rappelons que l'équation d'état est une relation de récurrence avec un second membre obtenu à partir de l'innovation. Il faut donc tirer des innovations suivant la loi estimée ou choisie, puis expliciter les récurrences qui font passer de l'état à une date à l'état à la date suivante et à la nouvelle observation. On prend comme état initial le dernier obtenu sur l'intervalle d'observation. Il est commode d'utiliser `simulate()`pour faire directement cette simulation à partir de la représentation espace-état du processus, représentation qu'on peut obtenir par `SS()` de **dse**[1].

1. La fonction `SS()` permet d'écrire des représentations espace-état de modèles représentés soit par le filtre de Kalman soit par le filtre d'innovation. La fonction reconnaît le modèle choisi par l'utilisateur d'après les noms des paramètres. Il est indispensable de consulter l'aide en ligne.

Dans les notations de **dse**, le filtre d'innovation, qui peut représenter le lissage exponentiel, s'écrit :

```
z(t) = F z(t-1) + G u(t) + K w(t-1)
y(t) = H z(t) + w(t)
```

Ici `u(t)` est un input, nul dans notre cas ; `w(t)` est l'innovation et `z(t)` correspond à x_{t-1} dans (8.5). Nous sommes maintenant en mesure d'écrire le code de la simulation. Pour cela nous définissons le modèle d'innovation par `SS()` et d'après l'estimation obtenue dans `es.1` :

```
> saiso=matrix(c(rep(0,11),1),nrow=1)
> saiso=rbind(saiso,cbind(diag(x=1,ncol=11,nrow=11),
+        matrix(rep(0,11),ncol=1)))
> # blocs ligne de la matrice de transition
> f1=matrix(c(1,rep(0,12)),nrow=1)
> f2=cbind(matrix(rep(0,12),ncol=1),saiso)
> fmat=rbind(f1,f2)
> gmat=NULL
> kmat=matrix(c(es.1$fit$par[1:2],rep(0,11)),ncol=1)
> hmat=matrix(c(1,rep(0,11),1),nrow=1)
> mod.es=SS(F=fmat,G=gmat,K=kmat,H=hmat,z0=es.1$states[nrow(es.1$states)-1,])
```

Nous lançons maintenant la simulation de 10 000 trajectoires des 6 premiers mois de 2002. La série des innovations est tirée dans la loi estimée.

```
> nsim=10000; resulsim=matrix(0,ncol= nsim, nrow=6)
> set.seed(975)
> for(i in 1:nsim){
+   resulsim[,i]=simulate(mod.es,
+            noise=list(w=matrix(rnorm(6,sd=es.1$sigma2^.5),ncol=1),
+            w0=rnorm(1,sd=es.1$sigma2^.5)))$output}
```

Enfin, nous représentons simultanément les quantiles d'ordre 5%, 10%, 90% et 95% de la série réalisée (fig. 8.12), c'est-à-dire des bandes de prédiction à 80% et 90%, basées sur les quantiles observés et non sur des quantiles théoriques.

```
> q2006=apply(resulsim,1,quantile,probs=c(0.05,.10,.90,.95))
> mat.rep=t(rbind(q2006,as.numeric(traf06_1_6)))
> matplot(1:6,mat.rep,type='l',lty=c(1,2,2,3,3),ylab='trafic',
+    xlab='temps : premier semestre 2006',col="black",lwd=1.5)
> leg.txt=c("Série",expression(q[0.05]),expression(q[0.10]),
+   expression(q[0.90]),expression(q[0.95]))
> legend(1,540,leg.txt,lty=c(1,2,2,3,3))
```

Nous observons que les réalisations sortent de l'intervalle à 90% dès le 5^{e} mois, sans pour autant diverger de l'allure générale de la série.

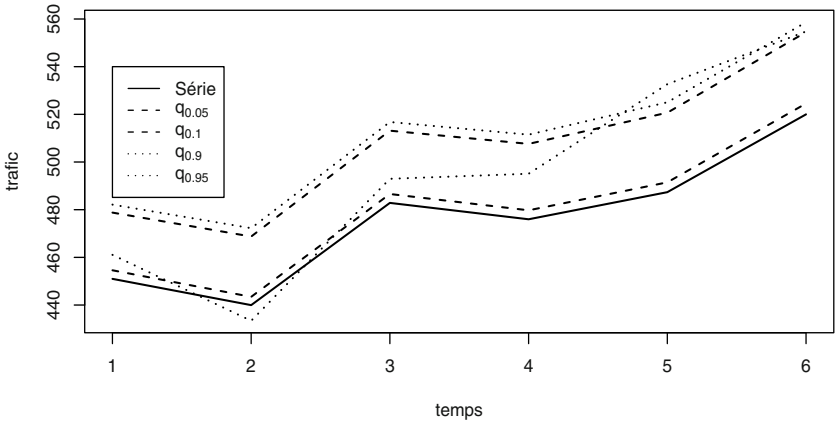

Fig. 8.12 – Prédiction du premier semestre 2006.

8.6 Estimation d'un SARIMA dans R - Vérification

Nous avons indiqué comment R comprend le modèle SARIMA$(p,0,0)(P,1,0)_{12}$ avec dérive (section 5.1). Pour estimer un tel modèle, R considère :

$$Y_t = a + b\,t + e_t$$

où après différenciation à l'ordre 12, $(1-\mathrm{B}^{12})e_t$ est SARMA$(p,0)(P,0)_{12}$ centré. Cette différenciation donne

$$(1-\mathrm{B}^{12})Y_t = 12\,b + (1-\mathrm{B}^{12})e_t$$

et $12b$ est la moyenne de la série différenciée. R estime b et non $12b$. Pour confirmer ces informations, estimons par ce biais le modèle du trafic avant obtenu manuellement. Pour cela, il nous faut introduire la variable explicative t dans le modèle.

```
> xmat=as.matrix(1:length(trafav))
> (mod4x=Arima(trafav,order=c(2,0,0),seasonal=list(order=c(1,1,0),period=12),
+   fixed=c(0,NA,NA,NA),xreg=xmat))

Series: trafav
ARIMA(2,0,0)(1,1,0)[12]
...
Coefficients:
      ar1     ar2     sar1    xmat
        0  0.3819  -0.6168  1.9533
s.e.    0  0.1339   0.1128  0.1279
```

```
sigma^2 estimated as 113.1:  log likelihood = -214.88
AIC = 437.76   AICc = 438.96   BIC = 447.88
```

Comparant cette estimation à celle de `mod4`, résumée dans l'expression (8.1), nous constatons qu'elles sont bien identiques. On voit la trace de cette méthode d'estimation dans `mod4` : si l'on examine sa structure on remarque à la fin de la liste

```
$ xreg      : int [1:68, 1] 1 2 3 4 5 6 7 8 9 10 ...
```

c'est-à-dire le régresseur explicitement introduit au début de la section.

Chapitre 9

Température mensuelle moyenne à Nottingham Castle

9.1 Exploration

Nous avons commencé à examiner la série des températures à Nottingham Castle (fig. 9.1) au chapitre 1, notamment dans le commentaire de la figure 1.9.

```
> data(nottem)
> plot.ts(nottem,xlab='temps',ylab='température')
```

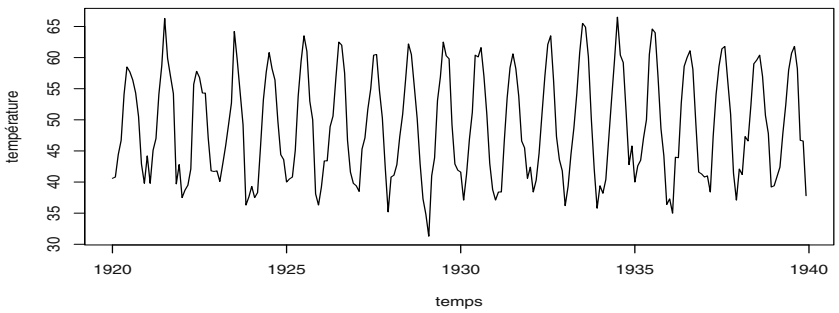

Fig. 9.1 – Températures à Nottingham Castle.

Nous avons conclu qu'elle n'est pas stationnaire et que la non-stationnarité est principalement liée à la saisonnalité de la série. Nous en envisageons ici deux modélisations, l'une où la saisonnalité est associée à un trend stochastique, l'autre où elle est associée à un trend déterministe. Nous construisons ensuite les prévisions correspondantes.

9.2 Modélisation

Partageons d'abord la série. Le début (1920 à 1936) servira à la modélisation, c'est la trajectoire d'apprentissage ; on comparera prévision et réalisation sur les années 1937 à 1939, trajectoire de validation.

```
> nott1=window(nottem,end=c(1936,12))
> nott2=window(nottem,start=c(1937,1))
> require(caschrono)
> plot2acf(nott1,diff(nott1,12),lag.max=40,
+   main=c("nott1",expression(paste("(1-",B^{12},") nott1",sep=""))))
```

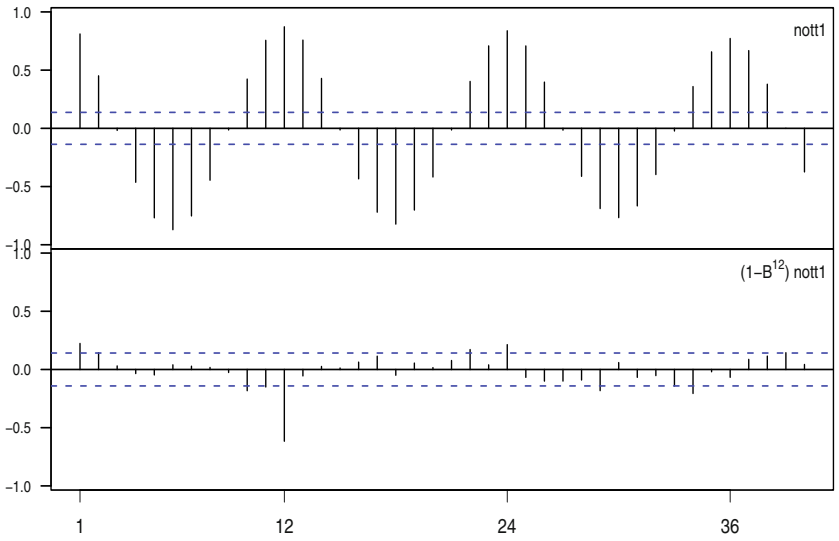

Fig. 9.2 – ACF de la température et de la température différenciée.

On observe que l'ACF présente des pics aux retards multiples de 12 qui ne s'atténuent que très lentement (fig. 9.2 haut). C'est le signe d'une non-stationnarité saisonnière. On dessine l'ACF de la série différenciée à l'ordre 12 (fig. 9.2 bas). Elle montre un pic à l'ordre 12, une valeur significativement différente de 0 en 24 suivie d'une forte atténuation, caractéristiques d'une série stationnaire avec saisonnalité.

9.2.1 Modèle SARIMA

Le modèle classique, Anderson (1976), repris par R pour modéliser cette série, est un SARIMA$(1,0,0)(2,1,0)_{12}$. Estimons-le.

```
> fitm=Arima(nott1,order=c(1,0,0),list(order=c(2,1,0),period=12))
> summary(fitm)

Series: nott1
ARIMA(1,0,0)(2,1,0)[12]
```

```
...
Coefficients:
        ar1     sar1     sar2
      0.324  -0.8848  -0.3042
s.e.  0.069   0.0708   0.0752

sigma^2 estimated as 5.76:  log likelihood = -445.44
AIC = 898.88   AICc = 899.09   BIC = 911.91

In-sample error measures:
         ME        RMSE         MAE         MPE        MAPE
 0.01243165  2.32827752  1.78093646 -0.23455300  3.81681564
       MASE
 0.40772539
```

Examinons la blancheur des résidus de l'ajustement jusqu'au retard 30 :

```
> ret6 = seq(6,30,6)
> t(Box.test.2(residuals(fitm),ret6,type="Ljung-Box",decim=2,fitdf=3))
        [,1]  [,2]  [,3] [,4]  [,5]
Retard  6.00 12.00 18.00 24.0 30.00
p-value 0.71  0.81  0.67  0.5  0.63
```

On conclut à la blancheur du bruit et on voit facilement que les coefficients sont très significatifs.
Le modèle finalement ajusté est avec $z_t \sim \text{BBN}(0, 5.76)$

$$(1 - \mathrm{B}^{12})y_t = \frac{1}{(1 - 0.324\,\mathrm{B})(1 + 0.885\,\mathrm{B}^{12} + 0.304\,\mathrm{B}^{24})} z_t. \tag{9.1}$$

C'est un modèle à composante saisonnière stochastique. La grande régularité de la série suggère de lui ajuster un modèle à composante saisonnière déterministe. C'est l'objet de la prochaine section. Ensuite, nous comparerons les deux approches par les qualités prédictives respectives de ces modèles et par leurs mesures d'erreur intra-échantillon.

9.2.2 Régression sur fonctions trigonométriques

La série des températures montre une très grande régularité. On se propose donc de la régresser sur des fonctions capables de capter cette régularité. Nous utilisons dans ce but les fonctions trigonométriques $\cos(\omega t)$ et $\sin(\omega t)$ de période 12. Ce sont les fonctions $\cos(2\pi f t)$, $\sin(2\pi f t), f = 1/12, 2/12, \cdots, 6/12$, f est une fréquence. Mais $\sin(\pi t) = 0\ \forall t$. Finalement on doit former la matrice des régresseurs, X d'élément (t, j)

$$= \begin{cases} \cos(2\pi \frac{j}{12} t) & \text{pour } j = 1, \cdots, 6, \\ \sin(2\pi \frac{j-6}{12} t) & \text{pour } j = 7, \cdots, 11 \end{cases} \tag{9.2}$$

sur lesquels on effectuera la régression de la température.

Préparation des données et premières régressions. Nous fabriquons le data frame des régresseurs `xmat0` d'après (9.2) :

```
> f=t(as.matrix(1:6))/12
> temps=as.matrix(1:length(nottem))
> xmat0=cbind(cos(2*pi*temps%*%f),sin(2*pi*temps%*%f))[,-12]
> xmat0=as.data.frame(xmat0)
> colnames(xmat0)=c('cos_1','cos_2','cos_3','cos_4','cos_5','cos_6',
+                   'sin_1','sin_2','sin_3','sin_4','sin_5')
```

Ces régresseurs étant calculés, nous séparons les intervalles d'apprentissage et de validation et effectuons une régression linéaire de `nottem` sur les fonctions trigonométriques.

```
> xmat1=xmat0[1:length(nott1),]
> xmat2=xmat0[(length(nott1)+1):length(nottem),]
> attach(xmat1,warn.conflicts=FALSE)
> # on régresse  sur les colonnes de la matrice
> mod1=lm(nott1~cos_1+cos_2+cos_3+cos_4+cos_5+cos_6+sin_1+
+                 sin_2+sin_3+sin_4+sin_5,data=xmat1)
> summary(mod1)

Call:
lm(formula = nott1 ~ cos_1 + cos_2 + cos_3 + cos_4 + cos_5 +
    cos_6 + sin_1 + sin_2 + sin_3 + sin_4 + sin_5, data = xmat1)

Residuals:
    Min      1Q  Median      3Q     Max
-7.5647 -1.5618  0.2529  1.4074  6.0059

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 48.942647   0.165139 296.373  < 2e-16 ***
cos_1       -9.301603   0.233541 -39.829  < 2e-16 ***
cos_2       -0.005392   0.233541  -0.023   0.9816
cos_3       -0.005882   0.233541  -0.025   0.9799
cos_4        0.110294   0.233541   0.472   0.6373
cos_5        0.228073   0.233541   0.977   0.3300
cos_6       -0.174020   0.165139  -1.054   0.2933
sin_1       -6.990084   0.233541 -29.931  < 2e-16 ***
sin_2        1.539318   0.233541   6.591  4.1e-10 ***
sin_3        0.334314   0.233541   1.431   0.1539
sin_4        0.494993   0.233541   2.120   0.0353 *
sin_5        0.197927   0.233541   0.848   0.3978
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.359 on 192 degrees of freedom
Multiple R-squared: 0.9296,        Adjusted R-squared: 0.9256
F-statistic: 230.5 on 11 and 192 DF,  p-value: < 2.2e-16
```

Remarques
- Les propriétés des fonctions trigonométriques sont rappelées en exercice.
- L'intervalle d'estimation comporte un nombre entier d'années et les fréquences retenues sont celles d'événements revenant au moins tous les 12 mois. Il y a donc orthogonalité entre les variables explicatives par année de $2n = 12$ mois. On peut le vérifier en calculant par exemple,
  `> t(as.matrix(xmat1[1:12,]))%*%as.matrix(xmat1[1:12,])`.

L'orthogonalité des colonnes de X nous permet d'attribuer sans ambiguïté une part de la variabilité de la série à chaque variable explicative (il n'y a aucune colinéarité entre les colonnes de X) et donc de supprimer *simultanément* les variables explicatives repérées comme non significatives sur la colonne des p-values, `Pr(>|t|)`. Nous voyons qu'il faut seulement conserver les variables `cos_1`, `sin_1`, `sin_2` et `sin_4`.

```
> mod2=lm(nott1~cos_1+sin_1+sin_2+sin_4,data=xmat1)
> summary(mod2)

Call:
lm(formula = nott1 ~ cos_1 + sin_1 + sin_2 + sin_4, data = xmat1)

Residuals:
    Min      1Q  Median      3Q     Max
-7.8427 -1.5334  0.2123  1.5524  6.1590

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  48.9426     0.1643 297.832  < 2e-16 ***
cos_1        -9.3016     0.2324 -40.025  < 2e-16 ***
sin_1        -6.9901     0.2324 -30.078  < 2e-16 ***
sin_2         1.5393     0.2324   6.624  3.2e-10 ***
sin_4         0.4950     0.2324   2.130   0.0344 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.347 on 199 degrees of freedom
Multiple R-squared: 0.9277,        Adjusted R-squared: 0.9263
F-statistic: 638.8 on 4 and 199 DF,  p-value: < 2.2e-16

> var.mod2=summary(mod2)$sigma^2
```

L'ajustement est très satisfaisant. Le modèle obtenu, dont la courbe est représentée sur SiteST, est :

$$y_t = 48.9426 - 9.3016\ \cos(2\pi\frac{1}{12}t) - 6.9901\ \sin(2\pi\frac{1}{12}t) + 1.5393\ \sin(2\pi\frac{2}{12}t) + 0.495\ \sin(2\pi\frac{4}{12}t) + z_t \quad (9.3)$$

mais les $\widehat{z}_t$ sont sans doute corrélés et nous allons examiner cette question.

Remarque
On aurait pu estimer le modèle MCO à l'aide de la fonction `Arima()` en n'indiquant aucun ordre AR ou MA :

```
> attach(xmat1,warn.conflicts=FALSE)
> xmat1a=cbind(cos_1,sin_1,sin_2,sin_4)
> mod2b=Arima(nott1,order=c(0,0,0),xreg=xmat1a)
```

On trouve effectivement les mêmes estimations que par `lm()` dans `mod2` ; cependant on note des différences dans les écarts types des estimateurs. Ceci peut s'expliquer par le fait que `Arima()` fournit une estimation de la matrice des covariances des estimateurs à partir du Hessien de la log-vraisemblance, estimation peu précise. La prédiction est ensuite obtenue par :

```
> attach(xmat2,warn.conflicts=FALSE)
> xmat2a=cbind(cos_1,sin_1,sin_2,sin_4)
> pred.mco2=forecast(mod2b,xreg=xmat2a)
```

Prise en compte de l'autocorrélation des résidus. L'ACF et la PACF des résidus de l'ajustement ci-dessus (fig. 9.3) montrent une légère autorégression aux ordres 1 et 24 en faveur d'un autorégressif saisonnier SARMA$(1,0)(2,0)_{12}$.

```
> acf2y(residuals(mod2),lag.max=30,numer=FALSE)
```

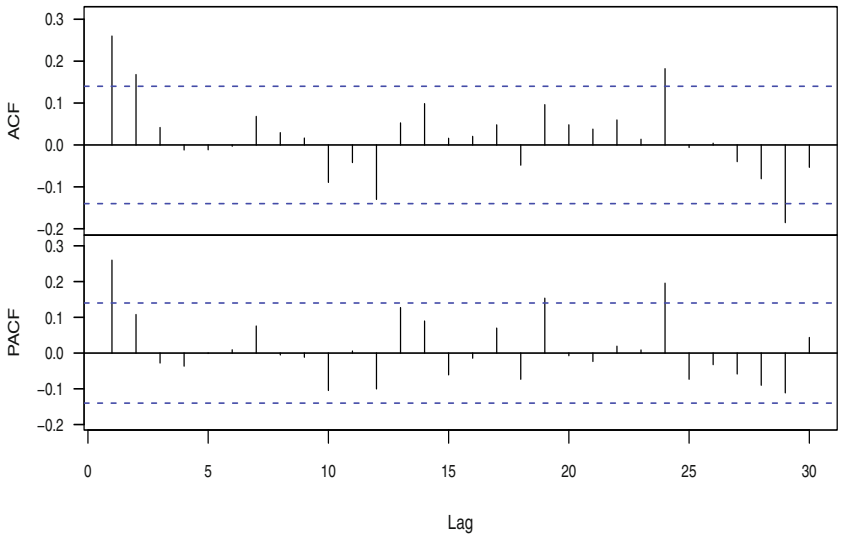

Fig. 9.3 – ACF et PACF des résidus de l'ajustement MCO.

Par ailleurs, l'ajustement MCO ci-dessus contenant une constante, les résidus sont de moyenne nulle, c'est pourquoi on utilise l'option `include.mean=FALSE`

```
> modar1b=Arima(residuals(mod2),order=c(1,0,0),
+  list(order=c(2,0,0),period=12),include.mean=FALSE)
> summary(modar1b)

Series: residuals(mod2)
ARIMA(1,0,0)(2,0,0)[12] with zero mean
...
Coefficients:
         ar1     sar1    sar2
      0.2920  -0.1246  0.2243
s.e.  0.0675   0.0687  0.0739

sigma^2 estimated as 4.644:  log likelihood = -446.91
AIC = 901.81   AICc = 902.01   BIC = 915.09

In-sample error measures:
          ME         RMSE          MAE          MPE
4.491004e-03 2.154925e+00 1.696185e+00 9.401833e+01
        MAPE         MASE
1.619752e+02 7.489265e-01
```

Examinons la blancheur des résidus.

```
> t(Box.test.2(residuals(modar1b),ret6,type="Ljung-Box",decim=2,fitdf=3))

        [,1]  [,2]  [,3]  [,4]  [,5]
Retard   6.0 12.00 18.00 24.00 30.00
p-value  0.9  0.95  0.89  0.93  0.93
```

L'ajustement est satisfaisant. Nous pouvons maintenant modéliser simultanément moyenne et erreur de la série en indiquant à `Arima()` la matrice de régression et les ordres du modèle ARMA de l'erreur.

```
> attach(xmat1,warn.conflicts=FALSE)
> mod3b=Arima(nott1,order=c(1,0,0),list(order=c(2,0,0),period=12),
+               xreg =cbind(cos_1,sin_1,sin_2,sin_4))
> summary(mod3b)

Series: nott1
ARIMA(1,0,0)(2,0,0)[12] with non-zero mean
...
Coefficients:
         ar1     sar1    sar2  intercept    cos_1    sin_1
      0.2919  -0.1247  0.2249    48.9831  -9.2931  -6.9806
s.e.  0.0675   0.0687  0.0740     0.2318   0.3042   0.3052
       sin_2   sin_4
      1.5542  0.5108
s.e.  0.2609  0.1980

sigma^2 estimated as 4.643:  log likelihood = -446.89
AIC = 911.77   AICc = 912.7   BIC = 941.63
```

```
In-sample error measures:
         ME        RMSE         MAE         MPE        MAPE
-0.02181195  2.15466211  1.69599523 -0.27328466  3.60400416
       MASE
 0.38827905
```

Vérifions la blancheur du résidu de cet ajustement :

```
> t(Box.test.2(residuals(mod3b),ret6,type="Ljung-Box",decim=2,fitdf=8))
```

```
        [,1]  [,2]  [,3]  [,4]  [,5]
Retard   6.0 12.00 18.00 24.00 30.00
p-value  0.9  0.95  0.89  0.93  0.93
```

les p-values sont élevées. Enfin, tous les coefficients sont significatifs :

```
> t_stat(mod3b)
```

```
            ar1      sar1     sar2 intercept     cos_1
t.stat 4.326942 -1.814458 3.041061  211.3235 -30.55133
p.val  0.000015  0.069607 0.002357    0.0000   0.00000
           sin_1    sin_2    sin_4
t.stat -22.87154 5.957029 2.580095
p.val    0.00000 0.000000 0.009877
```

Finalement le modèle ARMAX ajusté est avec $z_t \sim \text{BBN}(0, 4.64)$

$$\begin{aligned} y_t = 48.9831 - 9.2931\ \cos(2\pi\frac{1}{12}t) - 6.9806\ \sin(2\pi\frac{1}{12}t) \\ + 1.5542\ \sin(2\pi\frac{2}{12}t) + 0.5108\ \sin(2\pi\frac{4}{12}t) + u_t \\ u_t = \frac{1}{(1 - 0.2919\text{B})(1 + 0.1247\text{B}^{12} - 0.2249\text{B}^{24})} z_t \end{aligned} \quad (9.4)$$

et les estimations des écarts types des estimateurs des paramètres (*standard error of estimates*) sont données aux lignes `s.e.` dans la sortie ci-dessus. La partie déterministe du modèle est

$$\begin{aligned} \widehat{\mathsf{E}}(y_t) = 48.9831 - 9.2931 \cos(2\pi\frac{1}{12}t) - 6.9806 \sin(2\pi\frac{1}{12}t) \\ + 1.5542 \sin(2\pi\frac{2}{12}t) + 0.5108 \sin(2\pi\frac{4}{12}t), \end{aligned} \quad (9.5)$$

c'est l'estimation de la moyenne de y_t, la partie stochastique est u_t, SARMA de moyenne nulle.
La prédiction à l'horizon 1 de y_t dans l'échantillon est

$$y_{t|t-1} \equiv \mathsf{E}(y_t | y_{t-1}, y_{t-2}, \cdots) = \widehat{\mathsf{E}}(y_t) + u_{t|t-1}.$$

On l'obtient par `fitted(mod3b)` et $\widehat{\mathsf{E}}(y_t)$ peut s'obtenir en calculant `fitted(mod3b)-residuals(mod3b)`.

Remarques

– Comparons la variabilité du résidu de l'ajustement MCO `residuals(mod2b)` à celle de l'innovation dans (9.4), à travers les variances de ces séries. Ces variabilités sont respectivement : 5.4003 et 4.665. Le gain relatif apporté par la prise en compte de l'autocorrélation des erreurs vaut : 13.62%.

– On peut voir que le coefficient `sar1` dans `modar1b` n'est pas très significatif (p-value = 0.07). Si l'on veut contraindre ce paramètre à zéro, on utilisera l'instruction

```
(modar1c=Arima(residuals(mod2),order=c(1,0,0),
   list(order=c(2,0,0),period=12),include.mean=FALSE,fixed=c(NA,0,NA)))
```

– L'instruction `attach(xmat1)` permet de décrire `xreg` dans `Arima` par les noms des colonnes des variables, autrement on aurait écrit :
`xreg= xmat1[,c(1,7,8,10)]` et les noms des explicatives auraient été alors moins clairs.

9.3 Prévision

Effectuons la prévision de la série `nott2` des 36 derniers mois (section 9.2) : nous pourrons ainsi comparer prévision et réalisation.

▶ Prévision à partir du modéle SARIMA : le code pour la prévision et le graphique superposant réalisation de 1920 à 1936 et prévision des 3 années suivantes est :

```
> pred.sarima=forecast(fitm,h=36)
> tnupp=ts(pred.sarima$upper[,1],start=c(1937,1),frequency=12)
> tnlow=ts(pred.sarima$lower[,1],start=c(1937,1),frequency=12)
```

En examinant `pred.sarima` par `str(pred.sarima)`, on constate que seule la prédiction moyenne (`pred.sarima$mean`) est de classe `ts` et correctement datée. Or `ts.plot` ne peut dessiner que des séries de classe `ts`, c'est pourquoi on définit les séries `tnupp` et `tnlow`.

▶ Prévision à partir du modèle ARMAX : il faut donner la valeur des explicatives pour l'horizon de prédiction ; elle est contenue dans `xmat2` fabriquée plus haut.

```
> attach(xmat2,warn.conflicts=FALSE)
> pred.armax=forecast(mod3b,h=36,xreg=cbind(cos_1,sin_1,sin_2,sin_4))
```

L'option `warn.conflicts=FALSE` évite l'affichage d'avertissements. Elle a été employée ici pour ne pas alourdir les sorties. Observons qu'au départ on a fabriqué le data frame `xmat` dont les variables s'appellent : `cos_1`, `sin_1` ... On l'a partitionné en `xmat1` pour la période d'estimation et `xmat2` pour la période de prévision, dont les variables ont ces mêmes noms. Selon l'attachement qu'on fait, on dispose des variables explicatives sur la période d'estimation ou sur la période de prévision.

▶ Prévision à partir du modèle MCO : étant donné le peu de variabilité que capte la modélisation prenant en compte l'autocorrélation des erreurs, il est

intéressant de voir comment se comportent les prévisions d'après le modèle MCO par rapport aux prévisions plus élaborées des autres modèles. Nous les stockons donc :

```
> pred.mco=predict(mod2,xmat2,se.fit=TRUE,level=0.8,interval="prediction")
```

Nous avons demandé à conserver les écarts types des prédictions ainsi que les intervalles de prédiction à 80%.

9.4 Comparaison

Nous effectuons la comparaison des prédictions sur la période d'estimation et sur la période de prévision.
Comparons les coefficients des variables explicatives dans le modèle ARMAX (9.4) avec ceux de ces mêmes variables dans l'ajustement MCO (9.3). Nous constatons qu'ils sont très proches. De plus, les variances résiduelles, 5.51 pour l'ajustement MCO, 4.64 pour l'ARMAX et 5.76 pour le SARIMA ne sont pas très différentes. Examinons également les mesures d'erreur intra-échantillon fournies pas `summary()` dans les trois modèles (pour l'ajustement par MCO, nous avons appliqué `summary()` à `mod2b`). Les erreurs relatives n'ont pas ici grand intérêt puisqu'on travaille sur une même série dans les deux modèles.

Tableau 9.1 – Erreurs intra-échantillon dans les trois modélisations.

	MCO	SARIMA	ARMAX
ME	0.00	0.01	-0.02
RMSE	2.32	2.33	2.15
MAE	1.81	1.78	1.70
MPE	-0.26	-0.23	-0.27
MAPE	3.87	3.82	3.60
MASE	0.41	0.41	0.39

On voit sur le tableau 9.1 que les qualités des deux modélisations, ARMAX et SARIMA, mesurées dans l'échantillon, sont très voisines et supérieures à celles de la modélisation MCO (cf. exercice sur le filtrage).
Passons à la période de prévision. La figure 9.4 montre les intervalles à 80% obtenus par les deux méthodes ainsi que la réalisation. On n'observe pas de différences significatives entre les méthodes. Cette figure est obtenue ainsi : `plot.type="single"` donne des chronogrammes superposés ; d'autre part on veut dessiner de jolis axes, notamment une échelle du temps simple et lisible. On supprime donc le dessin standard des axes par `axes=FALSE` et on les définit ensuite par `axis`. L'examen du graphique, effectué avant la ligne `topleft=...` suggère de placer la légende à peu près au milieu horizontalement et en haut verticalement. L'abscisse du début de la légende est donc environ 1937.8 et l'ordonnée est le haut du graphique donné

par `par()$usr[4]`. `ts.plot()` et `plot.ts()` dessinent des séries sur le même axe mais codets.plot() accepte des séries sur des intervalles de temps différents.

```
> mxupp=ts(pred.armax$upper[,1],start=c(1937,1),frequency=12)
> mxlow=ts(pred.armax$lower[,1],start=c(1937,1),frequency=12)
> plot.ts(cbind(nott2,mxupp,mxlow,tnupp,tnlow),plot.type="single",axes=FALSE,
+ col=c(1,1,1,1,1),lty=c(1,3,3,5,5),xlab="temps",ylab='température')
> aa=time(nott2)[36]
> # examiner le résultat de ce qui précède avant d'exécuter la suite
> axis(1,at=c(1937:1939,aa),lab=c(as.character(1937:1939),""))
> axis(2,pretty(nott2),las=1)
> # examiner le résultat de ce qui précède ...
> topleft=par()$usr[c(1,4)] #coordonnées coin supérieur gauche du graphique
> text.leg=c("observé","ARMAX 80%","ARMAX 80%","SARIMA 80%","SARIMA 80%")
> legend(1937.8,topleft[2],text.leg,lty=c(1,3,3,5,5),merge=TRUE, cex=.8)
```

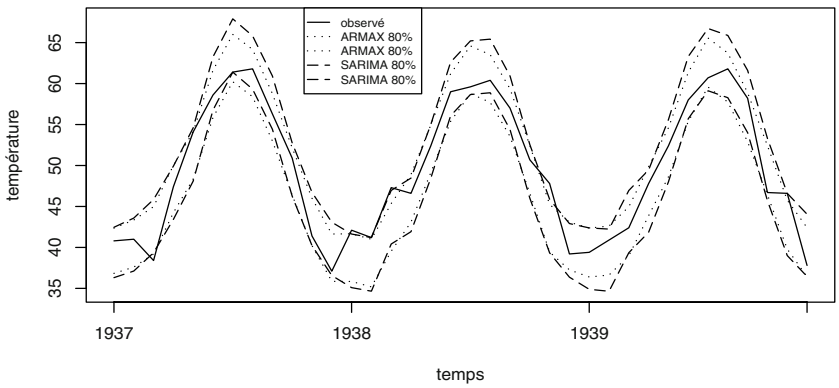

Fig. 9.4 – Années 1937-1939. Intervalles de prévision à 80% et réalisation.

Examinons les EQM (erreurs quadratiques moyennes) de prévision :

$$eqm_h = \sum_{i=1}^{h} (y_{T+i} - p_{m,i})^2/h,\ h = 1, \cdots, 36$$

où $T = 204$ est le numéro de la dernière observation de température utilisée pour l'estimation et $p_{m,i}$ désigne la prévision à l'horizon i par la méthode m, une des trois méthodes examinées. Notons que le moyennage porte ici sur l'horizon de prévision.
Pour calculer les erreurs quadratiques de prévision, nous rassemblons les prévisions moyennes obtenues par chaque méthode ainsi que la série à prédire dans la matrice `aa`, ensuite on calcule les erreurs quadratiques ; chaque ligne de la matrice `eq` correspond à un horizon et chaque colonne à une méthode. Enfin on fait la moyenne des erreurs par horizon de prédiction, ce qui nous donne la matrice `eqm`.

```
> aa=cbind(as.matrix(nott2),pred.mco2$mean,pred.armax$mean,pred.sarima$mean)
> eq=(aa[,1]%*%matrix(1,nrow=1,ncol=3)-aa[,c(2,3,4)])^2
> colnames(eq)=c('mco','armax','sarima')
> eqm=apply(eq,2,'cumsum')/1:36
> colnames(eqm)=c('mco','armax','sarima')
```

Sur le graphique des erreurs quadratiques mensuelles (fig. 9.5 bas), on constate que les trois méthodes sont mauvaises en général aux mêmes horizons, mais que SARIMA est généralement plus mauvaise que MCO ou ARMAX, elles-mêmes très proches. Ce classement est encore plus manifeste sur les erreurs de prédictions moyennées sur l'horizon (fig. 9.5 haut).

```
> matplot(1:36,eqm,type='l',lty=1:3,col='black',xlab="",xaxt="n",lwd=2)
> legend(x=25,y=3,c('mco','armax','sarima'),lty=1:3)
> matplot(1:36,eq,type='l',lty=1:3,col='black',xlab='horizon',lwd=2)
> legend(x=25,y=25,c('mco','armax','sarima'),lty=1:3)
```

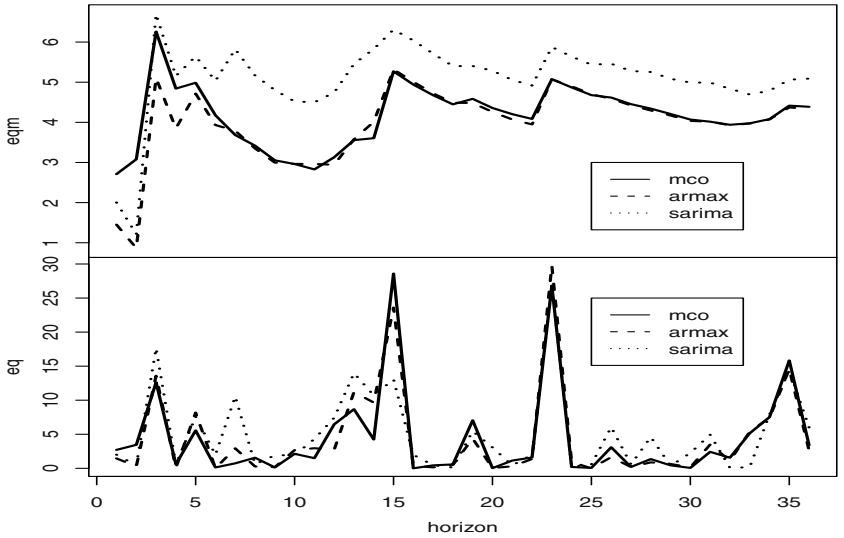

Fig. 9.5 – Années 1937-1939. Erreurs quadratiques de prévision.

Un examen attentif du lag plot (fig. 1.9) nous a convaincu que la série est non stationnaire. La grande régularité de cette série suggère que sa saisonnalité est déterministe. Nous l'avons modélisée à l'aide de fonctions trigonométriques. Par ailleurs, historiquement, cette série a été modélisée par un SARIMA, modèle à saisonnalité stochastique. Nous avons comparé les deux modèles à travers leurs pouvoirs prédictifs. Ils sont très proches. La simplicité du SARIMA plaide en faveur de son utilisation et le modèle retenu contient 3 paramètres. Par contre, les retards font perdre 37 points sur les 240. Dans l'ARMAX retenu, on doit estimer 5 paramètres et on perd 25 observations à cause des retards.

Dans la section suivante, nous considérons à nouveau l'exploration de cette série par un autre outil : l'analyse spectrale.

Exercice 9.1 (Filtrage)
Nous avons vu que les modèles SARIMA et ARMAX ont des variances résiduelles et des qualités prédictives intra-échantillon identiques. Pour comprendre cette proximité répondez aux questions suivantes.

1. Calculer les différences saisonnières de chaque variable explicative (matrice `xmat1a`) et examiner quelques lignes de la matrice résultat.
2. Calculer la moyenne et la variance de chaque série obtenue.
3. Calculer la différence saisonnière de `nott1`, la moyenne et la variance de cette série filtrée. Expliquer.

9.5 Analyse spectrale

L'analyse spectrale s'intéresse à toutes les fréquences envisageables sur une série. Elle peut donc capter des événements de fréquence inférieure à l'année, ce qui n'est pas le cas de la régression sur des fonctions trigonométriques. On ne trouvera pas ici d'exposé sur l'analyse spectrale d'une série temporelle, mais seulement une utilisation très élémentaire de cette méthode sur la série des températures.
Etant donné une série déterministe, y_t, de longueur $n = 2k$, il est possible de la décomposer exactement en une somme de fonctions trigonométriques :

$$y_t = a_0 + \sum_{j=1}^{k} [a_j \cos(2\pi f_j t) + b_j \sin(2\pi f_j t)] \tag{9.6}$$

où $f_j = 1/n, 2/n, \cdots, k/n$. Les f_j sont les *fréquences de Fourier*. On peut calculer les coefficients a_j, b_j par MCO et, vu les propriétés d'orthogonalité des séries trigonométriques, on obtient les expressions suivantes :

$$\begin{aligned} a_0 &= \overline{y}, \\ a_j = \frac{2}{n}\sum_{t=1} y_t \cos(2\pi\frac{j}{n}t), &\qquad b_j = \frac{2}{n}\sum_{t=1} y_t \sin(2\pi\frac{j}{n}t), \\ a_k = \frac{1}{n}\sum_{t=1}(-1)^t y_t, &\qquad b_k = 0. \end{aligned}$$

Par l'orthogonalité des fonctions trigonométriques on peut décomposer d'une unique façon la variabilité de la série, $\sum_{t=1}^{n}(y_t - \overline{y})^2$, par fréquence de Fourier. La variabilité associée à la fréquence j/n est appelée *périodogramme* I à la fréquence $f = j/n$ et vaut

$$I(\frac{j}{n}) = \frac{n}{2}(a_j^2 + b_j^2),\ j = 1, \cdots, k-1, \qquad I(\frac{1}{2}) = n a_k^2.$$

Finalement on a :

$$\sum_{t=1}^{n}(y_t - \overline{y})^2 = \sum_{j=1}^{k} I(\frac{j}{n}).$$

Si y_t est une série aléatoire stationnaire, on peut interpréter cette décomposition comme une ANOVA et tester la significativité de chaque fréquence. Mais ici nous l'utilisons de façon purement descriptive.
Examinons le périodogramme de la série `nottem`,

```
> require(TSA)
> periodogram(nottem)
> aa=periodogram(nottem,plot=FALSE)
```

représenté par un diagramme en bâtons (fig. 9.6).

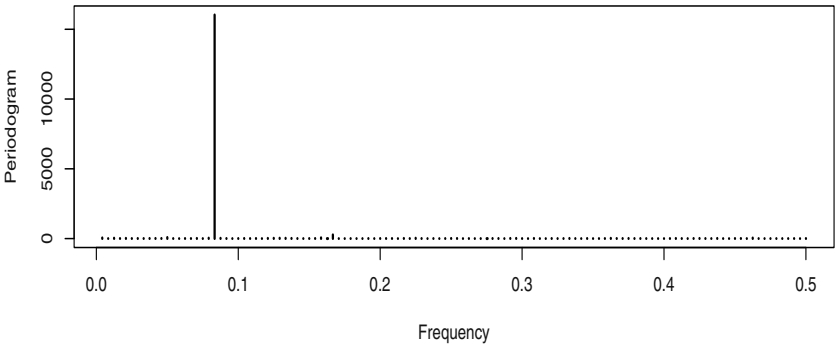

Fig. 9.6 – Périodogramme.

```
> aa=periodogram(nottem,plot=FALSE)
```

La série a 240 points soit 20 ans. On peut vérifier en examinant la structure de `aa` que les fréquences de Fourier, `aa$freq`, sont bien $1/240$, $2/240, \cdots$, $120/240$. La périodicité annuelle correspond à une fréquence de $20/240 = 0.083333$. Un événement qui revient tous les trois mois a une fréquence de $80/240 = 0.33333$. Le périodogramme `aa$spec` montre un pic qui écrase toutes les autres éventuelles contributions (fig. 9.6). Cherchons la fréquence correspondante :

```
> aa$freq[which.max(as.vector(aa$spec))]

[1] 0.08333333
```

qui est bien celle des événements annuels. Cherchons les 5 fréquences de plus grand périodogramme :

```
> ab=order(-aa$spec)[1:5]
> frq.spe=rbind(aa$freq[ab],aa$spec[ab])
> rownames(frq.spe)= c("Fréquence","Périodogramme")
```

Tableau 9.2 – Fréquences et périodogramme de plus grande énergie.

	1	2	3	4	5
Fréquence	0.08	0.17	0.05	0.16	0.00
Périodogramme	16028.50	270.15	82.15	60.71	52.69

Le tableau 9.2 montre une fréquence (12 mois) de périodogramme très élevée, suivie de la fréquence 6 mois. La fréquence suivante 0.05 correspond à un événement qui revient $0.050 \times 240 = 12$ fois dans la période, donc tous les 20 mois. On ne voit pas quel phénomène réel pourrait avoir une telle période.
L'analyse spectrale permet de découvrir les fréquences non directement évidentes, correspondant à une grande variabilité. Elle concerne également les séries stationnaires, mais nous n'aborderons pas ces questions qui dépassent le cadre de ce travail.

Exercice 9.2 (Orthogonalité des fonctions trigonométriques)
A partir de la formule d'Euler :

$$\cos(2\pi f) = \frac{\exp(2\pi i f) + \exp(-2\pi i f)}{2} \quad \sin(2\pi f) = \frac{\exp(2\pi i f) - \exp(-2\pi i f)}{2}$$

et de la somme d'une progression géométrique :

$$q + q^2 + \cdots + q^n = \frac{q(1-q^n)}{1-q}$$

pour tout nombre réel ou complexe $q \neq 1$ et pour une série de longueur $2n$ et $j, k = 0, 1, 2, \cdots, n/2$, vérifier les propriétés d'orthogonalité suivantes :

$$\sum_{t=1}^{n} \cos(2\pi \frac{j}{n} t) = 0 \text{ si } j \neq 0 \tag{9.7}$$

$$\sum_{t=1}^{n} \sin(2\pi \frac{j}{n} t) = 0 \tag{9.8}$$

$$\sum_{t=1}^{n} \cos(2\pi \frac{j}{n} t) \sin(2\pi \frac{k}{n} t) = 0 \tag{9.9}$$

$$\sum_{t=1}^{n} \cos(2\pi \frac{j}{n} t) \cos(2\pi \frac{k}{n} t) = \begin{cases} \frac{n}{2} & \text{si } j = k \ (j \neq 0 \text{ ou } n/2) \\ n & \text{si } j = k = 0 \\ 0 & \text{si } j \neq k \end{cases} \tag{9.10}$$

$$\sum_{t=1}^{n} \sin(2\pi \frac{j}{n} t) \sin(2\pi \frac{k}{n} t) = \begin{cases} \frac{n}{2} & \text{si } j = k \ (j \neq 0 \text{ ou } n/2) \\ 0 & \text{si } j \neq k \end{cases} \tag{9.11}$$

Chapitre 10

Consommation d'électricité

Au chapitre 3, nous avons expliqué la racine carrée de la consommation d'électricité, $\sqrt{\texttt{kwh}}$, par les degrés jours de climatisation, `cldd`, les degrés jours de chauffage `htdd` et deux fonctions du temps. Nous avons noté alors une autocorrélation significative des résidus du modèle ajusté (3.19). Après notre révision des modèles ARIMA (chap. 4 et 5), nous pouvons reprendre cette régression en tenant compte maintenant de cette autocorrélation. Comme au chapitre 3, nous utilisons les 14 premières années (1970-1983) comme période d'apprentissage et vérifions les qualités prédictives des modèles sur l'année 1984.
Identifiant d'abord la dynamique des résidus du modèle (3.19), nous modélisons ensuite simultanément l'erreur et la moyenne de la série de consommation (section 10.2). Nous envisageons également une modélisation alternative. Enfin, nous calculons la prédiction de la consommation pour la dernière année par les différentes méthodes et terminons en comparant les erreurs quadratiques moyennes de prévision.

10.1 Identification de la série des résidus obtenus par MCO

Le modèle que nous avons retenu pour expliquer la consommation $\sqrt{\texttt{kwh}}$ par les variables de température `cldd`, `htdd` et deux fonctions du temps, `temps` et $(\texttt{temps}-1977)^2$ est (3.19) que nous reproduisons ici :

$$\begin{aligned}\sqrt{\texttt{kwh}_t} &= -673.06 + 0.00066\ \texttt{htdd}_t + 0.01\ \texttt{cldd}_t \\ &\quad +0.3456\ \texttt{temps}_t - 0.0059\ (\texttt{temps}-1977)_t^2 + u_t. \qquad (10.1)\end{aligned}$$

Les résidus $\widehat{u}_t$ montraient une autocorrélation significative et nous allons identifier leur modèle, mais d'abord nous les recalculons.
Chargeons et représentons les données de l'étude (fig. 10.1) :

```
> require(caschrono)
> data(khct)
> plot.ts(khct,xlab='temps',main="",cex.lab=.9,cex.axis=.8,
+  oma.multi=c(4.5,3,.2,0),mar.multi=c(0,4,0,.5),las=0)
```

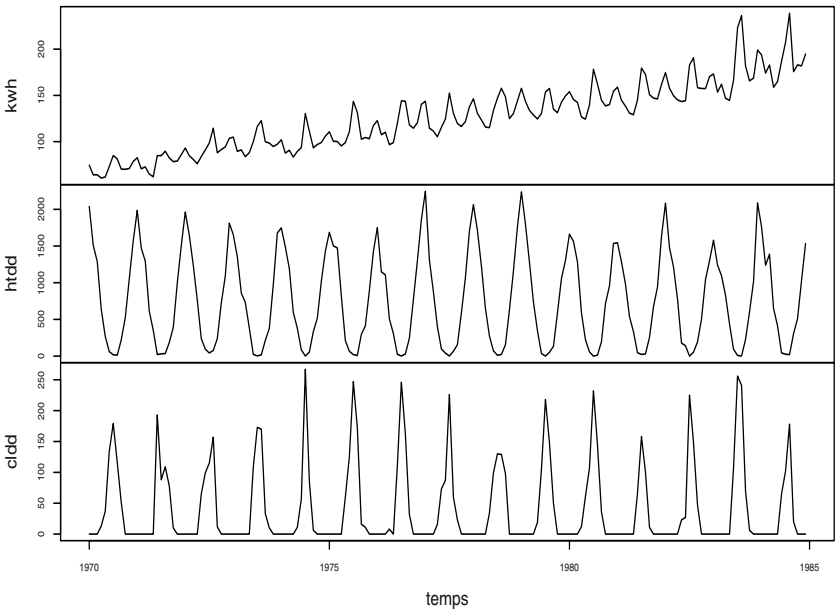

Fig. 10.1 – Chronogrammes de la consommation d'électricité et des variables de température.

Nous pouvons former le data frame des variables de la période d'apprentissage :

```
> khct.df=as.data.frame(window(cbind(khct,time(khct),
+ (time(khct)-1977)^2),end=c(1983,12)))
> colnames(khct.df)=c("kwh","htdd","cldd","t1","t1.2")
```

et réestimer (10.1) :

```
> mod2=lm(sqrt(kwh)~htdd+cldd+t1+t1.2,data=khct.df)
> u=ts(residuals(mod2),start=c(1970,1),frequency=12)
```

Nous avons transformé les résidus en série temporelle de saisonnalité 12, pour ne pas avoir à la préciser au cours de l'emploi de `Arima()` et d'autres fonctions. Examinons l'ACF de ces résidus (fig. 10.2) où on peut noter une persistance de l'aspect saisonnier sur l'ACF, mais à partir d'un niveau assez faible (0.5).

```
> acf2y(u,numer=FALSE)
```

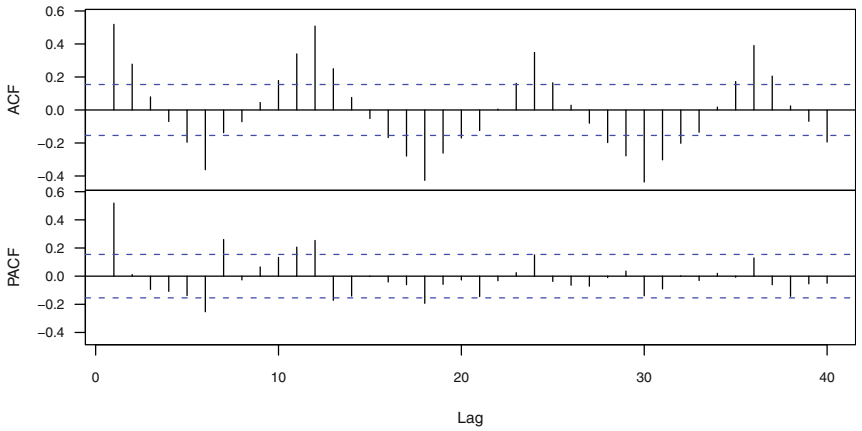

Fig. 10.2 – Fonction d'autocorrélation des résidus du Modèle 2.

Cette persistance suggère un terme AR, et non MA, d'ordre 12. Nous commençons par une modélisation généreuse en paramètres, en contraignant la moyenne à 0, car les résidus sont de moyenne empirique nulle :

```
> (modar1=Arima(u,order=c(3,0,1),seasonal=list(order=c(1,0,1)),
+    include.mean=FALSE))

Series: u
ARIMA(3,0,1)(1,0,1)[12] with zero mean
...
Coefficients:
          ar1     ar2     ar3     ma1    sar1     sma1
      -0.5585  0.5495  0.1480  1.0000  0.9834  -0.8125
s.e.   0.0770  0.0779  0.0778  0.0053  0.0176   0.0958

sigma^2 estimated as 0.0395:  log likelihood = 25.72
AIC = -37.43   AICc = -36.73   BIC = -15.57
```

On observe que le paramètre MA d'ordre 1 prend la valeur 1, ce qui donnerait un modèle non inversible, situation gênante si l'on doit faire des prévisions. Le test de blancheur :

```
> llag = seq(6,30,6)
> t(Box.test.2(residuals(modar1),llag,type ="Ljung-Box",decim=2,fitdf=6))

        [,1]  [,2]  [,3]  [,4] [,5]
Retard     6 12.00 18.00 24.00   30
p-value    1  0.91  0.99  0.99    1
```

donne des p-values élevées. Examinons alors la significativité des paramètres :

```
> t_stat(modar1)
```

```
             ar1      ar2      ar3      ma1     sar1
t.stat -7.252666 7.052729 1.903242 189.6090 55.96974
p.val   0.000000 0.000000 0.057009   0.0000  0.00000
            sma1
t.stat -8.482156
p.val   0.000000
```

On essaie donc de supprimer le terme autorégressif d'ordre 3 peu significatif.

```
> (modar2=Arima(u,order=c(2,0,1),seasonal=list(order= c(1,0,1)),
+                include.mean= FALSE))

Series: u
ARIMA(2,0,1)(1,0,1)[12] with zero mean
...
Coefficients:
         ar1      ar2      ma1    sar1     sma1
      1.1429  -0.2480  -0.7079  0.9847  -0.8225
s.e.  0.2680   0.1696   0.2463  0.0159   0.0890

sigma^2 estimated as 0.03999:  log likelihood = 24.01
AIC = -36.02   AICc = -35.5   BIC = -17.28

> t_stat(modar2)

            ar1       ar2       ma1     sar1      sma1
t.stat 4.264424 -1.461806 -2.874367 62.09656 -9.246156
p.val  0.000020  0.143794  0.004048  0.00000  0.000000
```

Notons le brutal changement de valeur du paramètre MA d'ordre 1 : la modélisation est loin d'être achevée. On constate que le paramètre autorégressif d'ordre 2 n'est pas significatif, nous l'éliminons :

```
> (modar3=Arima(u,order=c(1,0,1),seasonal=list(order=c(1,0,1)),
+               include.mean=FALSE))

Series: u
ARIMA(1,0,1)(1,0,1)[12] with zero mean
...
Coefficients:
         ar1      ma1   sar1     sma1
      0.6868  -0.2576  0.983  -0.8168
s.e.  0.1098   0.1454  0.017   0.0884

sigma^2 estimated as 0.04019:  log likelihood = 24.01
AIC = -38.02   AICc = -37.65   BIC = -22.4
```

Nous retenons donc un modèle SARMA$(1,1)(1,1)_{12}$. Bien que ce modèle soit satisfaisant, comme le coefficient d'autorégression saisonnière est élevé, nous examinons l'identification de la série également à l'aide de la procédure d'identification automatique, `auto.arima()` de **forecast**. Cette fonction peut envisager des différenciations saisonnières.

```
> (mod.auto=auto.arima(u,max.p=4,max.q=4,max.P=1,approximation=FALSE))

Series: u
ARIMA(1,0,1)(1,0,1)[12] with zero mean

Call: auto.arima(x = u, ...

Coefficients:
         ar1      ma1   sar1     sma1
      0.6868  -0.2576  0.983  -0.8168
s.e.  0.1098   0.1454  0.017   0.0884

sigma^2 estimated as 0.04019:  log likelihood = 24.01
AIC = -38.02   AICc = -37.65   BIC = -22.4
```

On observe que la procédure d'estimation ne suggère pas de racine unité simple ou saisonnière et préconise le modèle, inversible, que nous avions retenu. Vérifions enfin que les résidus forment un bruit blanc :

```
> t(Box.test.2(residuals(mod.auto),llag,type="Ljung-Box",decim=2,fitdf=4))

        [,1]  [,2]  [,3]  [,4] [,5]
Retard     6 12.00 18.00 24.00   30
p-value    1  0.88  0.98  0.99    1
```

Nous pouvons passer à l'estimation simultanée des deux parties du modèle : la moyenne et la dynamique de l'erreur.

10.2 Estimation du modèle ARMAX

Nous venons d'identifier un modèle pour l'erreur à travers le résidu de (10.1) et nous pouvons estimer maintenant un modèle linéaire avec u_t obéissant au modèle retenu dans `mod.auto`. Des modifications de modèle pourront encore survenir à cette étape, mais le travail sur les résidus des MCO a permis de dégrossir le problème à peu de frais calculatoires.
On définit la variable à expliquer et on précise les variables explicatives :

```
> kwh1rc=window(sqrt(khct[,"kwh"]),end=c(1983,12))
> xreg1=khct.df[,c("htdd","cldd","t1","t1.2")]
```

Nous estimons donc le modèle linéaire de la régression de `kwh1rc` sur `htdd`, `cldd`, `t1`, `t1.2` en spécifiant que l'erreur est un SARMA$(1,1)(1,1)_{12}$. La tentative d'estimation par

```
mdarx1=Arima(kwh1rc,order=c(1,0,1),seasonal=list(order=c(1,0,1)),xreg=xreg1)
```

donne le message d'erreur :

```
Erreur dans stats:::arima(x = x, order = order, seasonal = seasonal,...
  non-stationary seasonal AR part from CSS
```

Ce message indique (1) la méthode d'estimation en cause (Conditional Sum of Squares) employée au démarrage de l'estimation et (2) qu'au cours de l'estimation la valeur 1 a été obtenue pour le coefficient d'autorégression saisonnière. L'estimation de ce coefficient était 0.983 dans la modélisation du résidu faite par `auto.arima()` à la section précédente. Il est donc vraisemblable qu'au cours d'une itération de l'optimisation, une valeur supérieure ou égale à 1 a été obtenue. Il faut essayer de tourner cette difficulté. Avant de nous orienter vers une modélisation non stationnaire, réexaminons la sortie de l'estimation MCO du chapitre 3. On voit que `t1.2` est beaucoup moins significative que les autres variables explicatives. De plus, les significativités obtenues par MCO sont inexactes, étant donné l'auto-corrélation (forte) des erreurs. Essayons donc une modélisation sans cette variable, la 4e de nos variables explicatives.

```
> xreg2=xreg1[,-4]
> (mdarx2=Arima(kwh1rc,order=c(1,0,1),seasonal=list(order=c(1,0,1)),
+          xreg=xreg2))

Series: kwh1rc
ARIMA(1,0,1)(1,0,1)[12] with non-zero mean
...
Coefficients:
         ar1      ma1    sar1     sma1  intercept   htdd
      0.6990  -0.1137  0.9836  -0.7683  -682.2493  6e-04
s.e.  0.0973   0.1444  0.0132   0.0877    27.1625  1e-04
        cldd      t1
      0.0074  0.3502
s.e.  0.0005  0.0137

sigma^2 estimated as 0.03508:  log likelihood = 33.78
AIC = -49.56   AICc = -48.42   BIC = -21.45

>  t(Box.test.2(residuals(mdarx2),seq(6,30,6),type="Ljung-Box",
+               decim=2,fitdf=8))

        [,1]  [,2]  [,3]  [,4]  [,5]
Retard  6.00 12.00 18.00 24.00 30.00
p-value 0.99  0.52  0.87  0.94  0.97
```

Le résidu est un bruit blanc. On peut examiner les t-statistiques :

```
> t_stat(mdarx2)

            ar1       ma1     sar1      sma1 intercept
t.stat 7.186616 -0.787539 74.40682 -8.755891 -25.11732
p.val  0.000000  0.430966  0.00000  0.000000   0.00000
           htdd   cldd       t1
t.stat 4.873846 14.925 25.49415
p.val  0.000001  0.000  0.00000
```

Le terme MA d'ordre 1 n'étant pas significatif, on le supprime :

```
> (mdarx3c=Arima(kwh1rc,order=c(1,0,0),seasonal=list(order=c(1,0,1)),
+   xreg=xreg2))
```

```
Series: kwh1rc
ARIMA(1,0,0)(1,0,1)[12] with non-zero mean
...
Coefficients:
         ar1    sar1     sma1  intercept   htdd    cldd
      0.6323  0.9840  -0.7766  -680.9117  6e-04  0.0073
s.e.  0.0605  0.0138   0.0912    24.8211  1e-04  0.0005
          t1
      0.3496
s.e.  0.0126

sigma^2 estimated as 0.03524:  log likelihood = 33.47
AIC = -50.94   AICc = -50.03   BIC = -25.95
```

Le test de blancheur demeure satisfaisant :

```
> t(Box.test.2(residuals(mdarx3c),seq(6,30,6),type="Ljung-Box",
+              decim=2,fitdf=7))

        [,1]  [,2]  [,3]  [,4]  [,5]
Retard  6.00 12.00 18.00 24.00 30.00
p-value 0.99  0.41  0.78  0.89  0.93
```

Le modèle obtenu finalement est :

$$\begin{aligned}\sqrt{\texttt{kwh}}_t &= -680.9117 + 0.00057\ \texttt{htdd}_t + 0.0073\ \texttt{cldd}_t \\ &\quad + 0.3496\ \texttt{temps}_t + u_t,\ t = 1, \cdots, 168 \\ u_t &= \frac{1 - 0.7766\,\mathrm{B}^{12}}{(1 - 0.6323\,\mathrm{B})(1 - 0.984\,\mathrm{B}^{12})} z_t, \qquad \widehat{\mathsf{var}}(z_t) = 0.03524.\end{aligned} \tag{10.2}$$

Enfin on peut examiner le graphique des résidus à la moyenne, $\widehat{u}_t$.

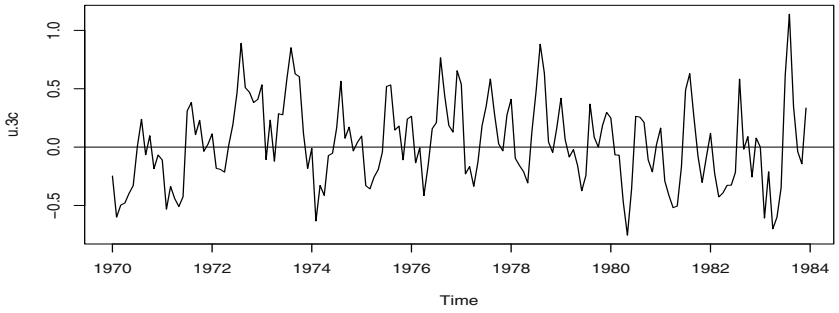

Fig. 10.3 – Chronogramme des résidus du modèle (10.2).

Après examen de la structure de `mdarx3c`, on voit que ces résidus s'obtiennent ainsi :

```
> u.3c=kwh1rc-as.matrix(xreg2)%*%as.matrix(mdarx3c$coef[5:7])-mdarx3c$coef[4]
```

Observons bien que `u.3c` est le résidu à la moyenne et non l'innovation qui est l'erreur de prévision de la série à l'horizon 1, quand on a modélisé également la dynamique de l'erreur. Le chronogramme obtenu par

```
> plot(temps1,u.3c,type='l')
> abline(h=0)
```

montre une forme parabolique de sommet situé vers 1973 (fig. 10.3). Nous avons abandonné le terme quadratique à cause de difficultés numériques. Pour voir s'il faut le réintroduire, nous effectuons la régression du résidu sur une parabole de sommet en janvier 1973 :

```
> tt=(khct.df$t1-1973)^2
> mmo=lm(u.3c~tt)
> summary(mmo)

Call:
lm(formula = u.3c ~ tt)

Residuals:
     Min       1Q   Median       3Q      Max
-0.75685 -0.25367 -0.04412  0.22099  1.21109

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0711727  0.0372012   1.913   0.0574 .
tt          -0.0012790  0.0007823  -1.635   0.1040
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3566 on 166 degrees of freedom
Multiple R-squared: 0.01585,        Adjusted R-squared: 0.009917
F-statistic: 2.673 on 1 and 166 DF,  p-value: 0.1040
```

`tt` n'est pas significative. Nous en restons donc au modèle (10.2).
Examinons la variabilité prise en compte par le modèle. Les variabilités de `kwh1rc`, la série initiale, de son espérance mathématique estimée et de la prédiction à l'horizon 1 dans le modèle (10.2) sont respectivement : 393.97, 361.39, 391.72. Le gain relatif apporté par la prise en compte de l'autocorrélation des erreurs est de $100\,(391.72-361.39)/391.72 = 7.7\%$. Il est bien plus important que dans le modèle (9.4) de la température à Nottingham Castle, où il est de 0.96%.
Il est intéressant de comparer l'estimation de la régression à celle obtenue, chapitre 3, pour le modèle 1 (3.6),

```
> (mod2s=lm(sqrt(kwh)~htdd+cldd+t1,data=khct.df))

Call:
lm(formula = sqrt(kwh) ~ htdd + cldd + t1, data = khct.df)

Coefficients:
```

```
(Intercept)         htdd         cldd           t1
 -6.741e+02    6.507e-04    9.957e-03    3.461e-01
```

Les estimations de la moyenne par cette régression MCO et dans `mdarx3c` sont proches et, dans les deux cas, les coefficients de `htdd` et `cldd` sont sensiblement différents entre eux ; ce qui justifie la transformation de la température en ces deux variables. Le gain tiré de la prise en compte de l'autocorrélation des erreurs est par contre manifeste sur la prévision, comme nous le verrons un peu plus loin.

Exercice 10.1
Superposer les chronogrammes de `u` défini à la section 1 et `u.3c`. Commenter.

Exercice 10.2
Tester que le coefficient de `cldd` est 10 fois plus grand que celui de `htdd` dans (10.2).

10.3 Estimation d'un modèle à erreur non stationnaire

L'estimation menée à la section 10.2 a d'abord donné un modèle non stationnaire. Considérant les résultats antérieurs, notamment la significativité approximative de la régression par MCO, nous avons enlevé la variable `t1.2` et avons obtenu le modèle ARMAX (10.2). Mais on peut considérer que, l'estimation du terme d'autorégression saisonnière du résidu $\widehat{u}$ étant de l'ordre de 0.98, il y a risque de racine unité et qu'il est donc prudent de différencier saisonnièrement, même si le choix automatique du modèle n'a pas retenu de modèle non stationnaire (section 10.1). C'est ce que nous faisons maintenant : nous corrigeons le modèle `armax1` du début de la section 10.2 qui n'a pas pu être estimé. Le modèle était :

```
mdarx1=Arima(kwh1rc,order=c(1,0,1),seasonal=list(order=c(1,0,1)),xreg=xreg1)
```

Nous y remplaçons l'autorégression saisonnière par une différenciation saisonnière à l'ordre 1 :

```
> (modarimax1=Arima(kwh1rc,order=c(1,0,1),seasonal=list(order=c(0,1,1)),
+    xreg=xreg1))

Series: kwh1rc
ARIMA(1,0,1)(0,1,1)[12]
...
Coefficients:
         ar1      ma1     sma1   htdd    cldd      t1
      0.9984  -0.5407  -0.7521  6e-04  0.0074  0.0052
s.e.  0.0026   0.1239   0.0737  0e+00  0.0005     NaN
         t1.2
      -0.0040
s.e.   0.0142

sigma^2 estimated as 0.03855:  log likelihood = 26.67
AIC = -37.34   AICc = -36.36   BIC = -12.94
```

Numériquement, `NaN` (Not a Number), écart type donné pour `t1`, indique que le programme d'optimisation n'a pas réussi à calculer les dérivées secondes de la fonction à optimiser. Ceci indique que statistiquement le modèle est mal spécifié. Il faut remédier à cette situation.
Le modèle `modarimax1` est un modèle intégré, précisément un modèle ARIMAX. C'est un modèle comportant des variables explicatives et une erreur non stationnaire. Pour estimer un modèle intégré, `Arima()` différencie *la série et les variables explicatives*, estime ensuite le modèle stationnaire ARMA correspondant et reconstitue les résidus sur la série initiale. R estime donc le modèle ARMAX :

$$(1-\mathrm{B}^{12})y_t = (1-\mathrm{B}^{12})\mathbf{x}_t'\beta + u_t,\ t=1,\cdots,168 \tag{10.3}$$

$$u_t = \frac{(1+\theta\mathrm{B})(1+\theta_{12}\mathrm{B}^{12})}{(1-\phi\mathrm{B})}z_t \tag{10.4}$$

où y_t désigne `kwh1rc` et $\mathbf{x}_t'$ une ligne des explicatives `xreg1` définies au début de la section 10.2. Or `diff(xreg1, lag = 12)` contient une colonne de constantes que R traite comme une variable explicative quelconque. Si l'on veut que R estime le modèle

$$(1-\mathrm{B}^{12})y_t = c + (1-\mathrm{B}^{12})\mathbf{x}_t'\beta + u_t,\ t=1,\cdots,168 \tag{10.5}$$

$$u_t = \frac{1+\theta_{12}\mathrm{B}^{12}}{(1-\phi\mathrm{B})}z_t, \tag{10.6}$$

il faut préciser qu'on veut estimer un modèle avec dérive (`include.drift=TRUE`) [1].
On obtient :

```
> (modarimax1b=Arima(kwh1rc,order=c(1,0,1),seasonal=list(order=c(0,1,1)),
+                    xreg = xreg1, include.drift=TRUE) )

Series: kwh1rc
ARIMA(1,0,1)(0,1,1)[12] with drift
...
Coefficients:
         ar1      ma1     sma1   htdd    cldd     t1
      0.6556  -0.1144  -0.7775  6e-04  0.0072  0.004
s.e.  0.1171   0.1617   0.0798  0e+00  0.0005    NaN
         t1.2   drift
      -0.0056  0.0287
s.e.   0.0024  0.0010

sigma^2 estimated as 0.03432:  log likelihood = 35.9
AIC = -53.8   AICc = -52.57   BIC = -26.35
```

1. On a discuté le fonctionnement de la fonction `Arima()` dans les modèles intégrés (section 8.6 et section 5.1).

La dérive c est correctement estimée, mais on a toujours une mauvaise spécification (un écart type n'est pas estimé). On se trouve en fait avec deux variables représentant une constante et il faut supprimer $(1 - B^{12})$`t1` qui est une constante et fait double emploi avec la dérive. C'est la troisième colonne de la matrice des variables explicatives.

```
> xreg1b=xreg1[,-3]
> (mx2b=Arima(kwh1rc,order=c(1,0,1),seasonal=list(order=c(0,1,1)),
+                       xreg =xreg1b,include.drift= TRUE))

Series: kwh1rc
ARIMA(1,0,1)(0,1,1)[12] with drift
...
Coefficients:
         ar1      ma1     sma1   htdd    cldd     t1.2
      0.6559  -0.1148  -0.7774  6e-04  0.0072  -0.0056
s.e.  0.1171   0.1620   0.0798  1e-04  0.0005   0.0024
       drift
      0.0291
s.e.  0.0010

sigma^2 estimated as 0.03432:  log likelihood = 35.9
AIC = -55.8   AICc = -54.82   BIC = -31.4

> t(Box.test.2(residuals(mx2b),seq(6,30,6),type="Ljung-Box",decim=2,fitdf=7))

        [,1]  [,2]  [,3]  [,4]  [,5]
Retard     6 12.00 18.00 24.00 30.00
p-value    1  0.68  0.93  0.97  0.99
```

Les p-values de ce test de blancheur sont très satisfaisantes. On examine maintenant la significativité des coefficients.

```
> t_stat(mx2b)

            ar1       ma1      sma1     htdd     cldd
t.stat 5.600948 -0.708908 -9.740972 4.265007 14.45168
p.val  0.000000  0.478382  0.000000 0.000020  0.00000
            t1.2    drift
t.stat -2.293603 28.68058
p.val   0.021813  0.00000
```

On peut simplifier en éliminant le terme MA d'ordre 1 :

```
> (modarimax2c=Arima(kwh1rc,order=c(1,0,0),seasonal=list(order=c(0,1,1)),
+                     xreg=xreg1b,include.drift=TRUE) )

Series: kwh1rc
ARIMA(1,0,0)(0,1,1)[12] with drift
...
Coefficients:
         ar1     sma1   htdd    cldd     t1.2   drift
      0.5814  -0.7896  6e-04  0.0071  -0.0056  0.0290
```

```
s.e.  0.0671   0.0806  1e-04  0.0005   0.0023  0.0009

sigma^2 estimated as 0.0343:  log likelihood = 35.65
AIC = -57.3   AICc = -56.54   BIC = -35.95
```

```
> resmx=residuals(modarimax2c)
> t(Box.test.2(resmx,seq(6,30,6),type="Ljung-Box",decim=2,fitdf=6))
```

```
        [,1]  [,2]  [,3]  [,4]  [,5]
Retard  6.00 12.00 18.00 24.00 30.00
p-value 0.99  0.62  0.89  0.95  0.98
```

Les p-values sont suffisamment élevées (ici encore, on note que la p-value du terme quadratique est très supérieure aux autres). Enfin, tous les paramètres sont significatifs :

```
> t_stat(modarimax2c)
```

```
            ar1      sma1     htdd     cldd      t1.2
t.stat 8.670446 -9.796514 4.219374 14.80818 -2.462667
p.val  0.000000  0.000000 0.000024  0.00000  0.013791
          drift
t.stat 31.67535
p.val   0.00000
```

En résumé, on a estimé le modèle :

$$(1-\mathrm{B}^{12})\sqrt{\texttt{kwh}_t} =$$
$$0.34824 + (1-\mathrm{B}^{12})(0.00059\ \texttt{htdd}_t + 0.00711\ \texttt{cldd}_t - 0.00559\ (\texttt{temps}_t - 1973)^2) + u_t,\ t = 1, \cdots, 168$$
$$u_t = \frac{1 - 0.7896\ \mathrm{B}^{12}}{1 - 0.5814\ \mathrm{B}} z_t, \qquad \widehat{\mathsf{var}}(z_t) = 0.0343. \quad (10.7)$$

Dans l'écriture du modèle, on a bien porté la dérive annuelle alors que c'est la dérive mensuelle qui est fournie par `Arima()`. La normalité a été examinée à la section 3.6 sur les résidus des MCO. Il n'est pas nécessaire d'y revenir. On peut observer que les modèles (10.2) et (10.7) sont très proches numériquement.

10.4 Prévision de l'année 1984

Nous n'avons pas utilisé pour l'estimation la dernière année de données disponibles. Nous disposons ainsi des variables pour l'année à prédire. On pourra ainsi comparer prédiction et réalisation de la consommation en 1984. Nous superposerons les intervalles à 80% des prédictions sans et avec dynamique de l'erreur sur un même graphique et calculerons les EQM de prévision pour chaque mois de 1984, afin de comparer les trois méthodes que nous avons essayées.
La situation est la suivante (cf. chap. 3, section 3.5). F désignant les indices de temps de la trajectoire à prédire, c'est-à-dire les mois de 1984, le modèle est

$$Y_F = X_F\boldsymbol{\beta} + U_F \tag{10.8}$$

mais à la différence de (3.13) où $U_F \sim \mathcal{N}(0, \sigma_u^2 I_m)$, maintenant, U_F est une série temporelle, un SARMA ou un SARIMA dont la matrice des covariances a été estimée sur la période d'apprentissage (1970-1983) et les valeurs initiales sont fournies par les résidus de l'estimation du modèle.

Prévision par le modèle MCO. Elle a été effectuée au chapitre 3 (section 3.5, objet `p2`), mais pour la commodité de l'exposé nous reprenons les calculs. Nous sélectionnons d'abord les variables explicatives pour l'année 1984, rassemblées dans le data frame `an84`, et la variable dépendante pour 1984, `kwh2rc`, puis effectuons la prévision sur le modèle `mod2` :

```
> khct.df.84=as.data.frame(window(cbind(khct,time(khct),
+ (time(khct)-1977)^2),start=c(1984,1)))
> colnames(khct.df.84)=c("kwh","htdd","cldd","t1","t1.2")
> p2=predict(mod2,khct.df.84,interval="prediction",level=0.80,se.fit=TRUE)
```

Prévision par le modèle ARMAX. Nous avons dû éliminer le terme quadratique de temps, nous l'éliminons donc des régresseurs pour cette méthode.

```
> xreg.p=khct.df.84[,2:4]
> prev.3c=forecast(mdarx3c,h=12,level=c(80,95),fan=FALSE,xreg=xreg.p)
```

Examinons quelques lignes de la sortie `prev.3c`

```
> str(prev.3c)
List of 10
 $ method   : chr "ARIMA(1,0,0)(1,0,1)[12] with non-zero mean"
 $ model    :List of 15
  ..$ coef     : Named num [1:7] 6.32e-01 9.84e-01 -7.77e-01 -6.81e+02 5.70e-04 ...
  .. ..- attr(*, "names")= chr [1:7] "ar1" "sar1" "sma1" "intercept" ...
 ...
 $ level    : num [1:2] 80 95
 $ mean     : Time-Series [1:12] from 1984 to 1985: 13.9 13.2 13.3 12.7 12.6 ...
 $ lower    : num [1:12, 1:2] 13.7 12.9 13 12.4 12.3 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "80%" "95%"
 $ upper    : num [1:12, 1:2] 14.2 13.5 13.6 13 12.9 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "80%" "95%"
...
```

C'est une liste de 10 objets dont seulement quelques-uns ont été reproduits. Les premiers reprennent le modèle sur lequel est basée la prévision, la composante `level` donne les niveaux de confiance choisis pour les intervalles de prédiction, `mean` est la prédiction de la moyenne ; c'est bien une série de 12 valeurs et `upper`

et `lower` donnent les bornes inférieures et supérieures de ces intervalles pour les deux niveaux retenus. On peut extraire les écarts types de prévision à l'aide de la prévision moyenne et d'une borne inférieure de niveau connu. Ainsi,

```
> etyp.pred=(prev.3c$upper[,1]-prev.3c$mean)/qnorm(0.9)
> etyp.pred2=(prev.3c$upper[,2]-prev.3c$mean)/qnorm(0.975)
```

sont identiques.

Prévision par le modèle ARIMAX. Nous effectuons la prévision sur les 12 derniers mois d'après le modèle (10.7). Pour cela nous avons besoin du terme quadratique et nous devons éliminer le terme linéaire de temps.

```
> xreg.2c=khct.df.84[,c(2,3,5)]
> prev.2c=forecast(modarimax2c,h=12,level=c(80,95),fan=FALSE,xreg=xreg.2c)
```

On compare maintenant réalisation, `kwh2rc`, et prévision en superposant les intervalles de prévision et la valeur observée, seulement pour MCO et ARMAX afin de ne pas surcharger le graphique. Nous comparerons ensuite les trois méthodes par leurs EQM. Dans le code ci-dessous, nous utilisons les abréviations de noms de mois de la fonction `months()`, extraits d'une série de 12 dates.

```
> kwh2rc=window(sqrt(khct[,"kwh"]),start=c(1984,1))
> aa=seq(as.Date("2000/1/1"),by="month",length.out=12)
> id.mois=months(aa,abbreviate=TRUE)
> az=cbind(kwh2rc,prev.3c$lower[,1],prev.3c$upper[,1],p2$fit[,2:3])
> plot(ts(az,frequency=12,start=c(1984,1)),plot.type="single",
+  lty=c(1,2,2,3,3),ylab=expression(sqrt(kwh)),cex.main=.8,
+  xlab='1984',xaxt ="n")
> axis(1,at=seq(1984, 1984.917, length.out=12),labels=id.mois)
> legend(par("usr")[1], par("usr")[4],c("Valeur observée",
+ "Prédiction ARMAX","Prédiction MCO"),lwd=1,lty=c(1,2,3))
```

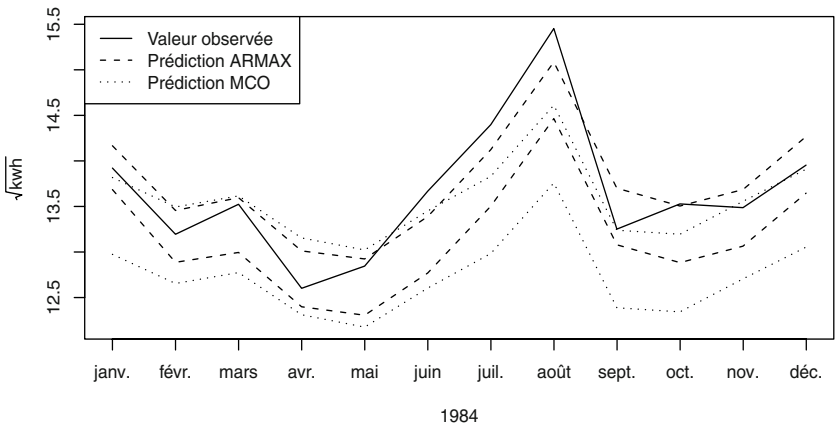

Fig. 10.4 – Consommation en 1984, bandes de prédiction par MCO et ARMAX.

On a porté (fig. 10.4) les bandes de prédiction à 80% obtenues par les MCO et par la modélisation ARMAX.

```
> # ARMAX
> un80=sum((kwh2rc<prev.3c$lower[,1])|(kwh2rc>prev.3c$upper[,1]))
> un95=sum((kwh2rc<prev.3c$lower[,2])|(kwh2rc>prev.3c$upper[,2]))
> cat('taux de non-appartenance 95 (ARMAX)= ',sum(un95)/12,'\n')

taux de non-appartenance 95 (ARMAX)=  0.25

> cat('taux de non-appartenance 80 (ARMAX)= ',sum(un80)/12,'\n')

taux de non-appartenance 80 (ARMAX)=  0.3333333

> pp=c(sum(un80),sum(un95))/12
> # ARIMAX
> un80i=sum((kwh2rc<prev.2c$lower[,1])|(kwh2rc>prev.2c$upper[,1]))
> un95i=sum((kwh2rc<prev.2c$lower[,2])|(kwh2rc>prev.2c$upper[,2]))
> ppi= c(sum(un80i),sum(un95i))/12
> cat("taux de non-appartenance 80 (ARIMAX)= ",sum(un80i)/12,'\n')

taux de non-appartenance 80 (ARIMAX)=  0.5833333

> cat('taux de non-appartenance 95 (ARIMAX)= ',sum(un95i)/12,'\n')

taux de non-appartenance 95 (ARIMAX)=  0.3333333
```

Les taux de non-appartenance (des réalisations aux intervalles de prédiction à 80% et 95%) sont de (0.33, 0.25) pour le modèle ARMAX et (0.58, 0.33) pour le modèle ARIMAX. Ils sont sensiblement plus élevés que les taux attendus (0.20 et 0.05), ce qui suggère un changement de la dynamique au cours de l'année prédite. On voit sur le graphique qu'à niveau donné (80%), quand on tient compte de la dynamique de l'erreur (1) les bandes de prédiction sont plus étroites et (2) le taux de couverture (théoriquement de 80%) est plus proche de cette valeur que quand on n'en tient pas compte.
Notons que dans la pratique les variables explicatives à un certain horizon h ne sont pas disponibles et qu'on doit les prédire. Or les séries `htdd` et `cldd` sont des transformations non linéaires de la série des températures moyennes quotidiennes. La logique est donc, à partir des séries de température quotidienne, de faire leur prédiction à l'horizon voulu en jours, puis d'en déduire les prévisions de `htdd` et `cldd`.

Comparaison des prédictions. Nous comparons les trois modélisations (MCO vue au chapitre 3, ARMAX et ARIMAX ci-dessus) par les erreurs quadratiques moyennes de prévision pour l'année 1984 que nous définissons par

$$eqm_h = \sum_{i=1}^{h} (\sqrt{\texttt{kwh}}_{T+i} - p_{m,i})^2/h,\ h = 1, \cdots, 12$$

où $T = 168$ est le numéro de la dernière observation utilisée pour l'estimation et $p_{m,i}$ désigne la prévision à l'horizon i par la méthode m, une des trois méthodes examinées. Le tableau ci-dessous regroupe ces eqm. On y observe une nette supériorité de la modélisation ARMAX, sur MCO évidemment, mais également sur la modélisation ARIMAX.

```
> p0=predict(mod2,khct.df.84,se.fit=TRUE) #MCO
> # ARMAX
> prev.3c=forecast(mdarx3c,h=12,level=c(80,95),fan=FALSE,xreg=xreg.p)
> # ARIMAX
> prev.2c=forecast(modarimax2c,h=12,level=c(80,95),fan=FALSE,xreg=xreg.2c)
> b.arimax=cumsum((kwh2rc-prev.2c$mean)^2)/1:12 #EQM
> b.armax=cumsum((kwh2rc-prev.3c$mean)^2)/1:12
> b.mco=cumsum((kwh2rc-p0$fit)^2)/1:12
> aaa=t(cbind(b.mco,b.armax,b.arimax))
> rownames(aaa)=c("MCO","ARMAX","ARIMAX"); colnames(aaa)=id.mois
```

Tableau 10.1 – Erreurs quadratiques de prévision pour 1984.

	févr.	avr.	juin	août	oct.	déc.
MCO	0.15	0.10	0.15	0.43	0.42	0.38
ARMAX	0.00	0.02	0.08	0.16	0.14	0.12
ARIMAX	0.02	0.06	0.17	0.29	0.26	0.23

10.5 Prédiction sur la série non transformée

On a modélisé la racine carrée de la consommation, on a donc prédit cette racine carrée. A cause de la symétrie de la loi normale, les intervalles de prédiction sont symétriques autour de la prédiction moyenne. Leurs bornes sont les quantiles d'ordre $\alpha/2$ et $1-\alpha/2$ de la prédiction, avec par exemple $\alpha = 10\%$. La transformation $x \hookrightarrow x^2$ est monotone croissante pour $x > 0$ et transforme donc les quantiles en quantiles de mêmes ordres, mais les bornes transformées ne sont évidemment pas symétriques autour des moyennes transformées. De plus, si X est une v.a. de moyenne μ, alors μ^2 n'est que l'approximation à l'ordre 1 de la moyenne de la v.a. X^2. Aussi, pour obtenir des prédictions de la série initiale, sans approximations difficiles à calculer, nous allons simuler un certain nombre de trajectoires de $\sqrt{\texttt{kwh}_t}$ pour l'année 1984, élever au carré et calculer sur ces données transformées la médiane et différents quantiles. Nous aurons ainsi des intervalles de prédiction sur la série originale.

Nous utilisons `simulate()` de **dse** suivant la même démarche qu'à la section 7.3 du chapitre 7.

Nous devons simuler des trajectoires de 12 observations d'une série obéissant à `mdarx3c`, mis en forme dans l'expression (10.2). Nous allons estimer la moyenne

de `kwh2rc` pour 1984 et nous ajouterons à cette estimation des trajectoires de l'erreur u_t simulées. Dans le vocabulaire de `simulate()`, nous nous intéressons à

$$A(\mathrm{B})u_t = B(\mathrm{B})z_t. \tag{10.9}$$

Nous voyons sur (10.2) que l'autorégression va jusqu'au retard 13 et la moyenne mobile jusqu'au retard 12. Il faut donc extraire du résultat de l'estimation ces valeurs. L'aide en ligne de `simulate()` nous précise que ces valeurs doivent être fournies en temps retourné : la première valeur doit correspondre à t0.

```
> require(dse)
> ret.u=rev(u.3c)[1:13]; ret.z=rev(residuals(mdarx3c))[1:12]
> coef0=mdarx3c$coef
```

Le résidu $\widehat{u}_t$ a été calculé plus haut, il est stocké dans `u.3c`, il est retourné à l'aide de la fonction `rev()`. Ensuite il faut écrire les arrays, `A` et `B` correspondant à (10.9) à partir des coefficients estimés dans `mdarx3c`. Comme l'autorégression est saisonnière, il faudra effectuer le produit des polynômes autorégressifs.

```
> require(polynom)
> A.u=polynomial(c(1,-coef0[1]))*polynomial(c(1,rep(0,11),-coef0[2]))
> A.arr=array(A.u,c(length(A.u),1,1))
> B.arr=array(c(1,coef0[3]),c(2,1,1))
> mod.u=ARMA(A=A.arr, B=B.arr)
```

La prédiction `pred.moy` de la moyenne est identique pour toutes les simulations ; on lui ajoute la simulation du bruit pour obtenir une trajectoire de $\sqrt{\texttt{kwh}}$ et enfin on revient à `kwh` :

```
> pred.moy=mdarx3c$coef[4]+as.matrix(xreg.p)%*%as.matrix(mdarx3c$coef[5:7])
> nsim=10000; pred.y=matrix(NA,ncol=nsim,nrow=12)
> set.seed(539)
> wsim=matrix(rnorm(12*nsim, sd=mdarx3c$sigma2^.5),ncol=nsim,nrow=12)
> for(i in 1:nsim){
+     pred.y[,i]=pred.moy +simulate(mod.u,y0=ret.u,
+       noise=list(w=as.matrix(wsim[,i]),w0=ret.z),sampleT=12)$output}
> pred.kwh=pred.y^2 # retour aux données initiales
> quant=apply(pred.kwh,1,quantile,probs=c(.05,.10,.90,.95,.5)) # quantiles
```

On a calculé les quantiles des simulations pour chaque horizon de 1 à 12 et on les représente ainsi que la série réalisée (fig. 10.5).

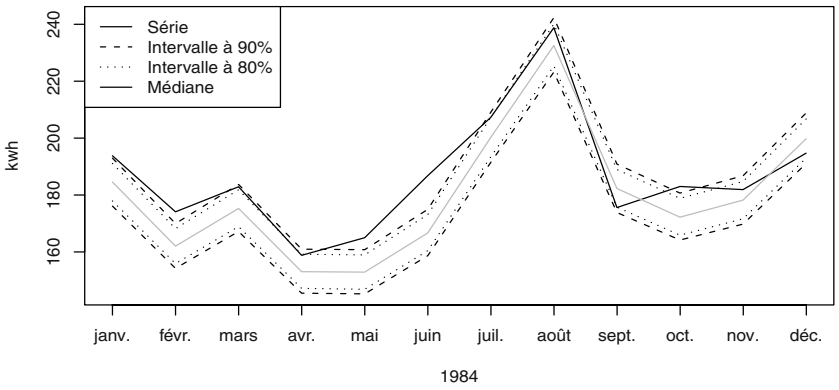

Fig. 10.5 – Prédiction de la consommation en 1984 par simulation.

On retiendra du tableau 10.1 que le modèle (10.2) est le plus satisfaisant des trois.

Chapitre 11

Production de lait

Nous étudions la collecte mensuelle de lait en France de janvier 1980 à janvier 2010. Cette série a été affectée par l'introduction de quota laitiers en janvier 1984.

11.1 Analyse exploratoire

Nous chargeons la série

```
> require(caschrono)
> lait2=read.table(system.file("/import/collecteLait.txt",
+ package="caschrono"),header=FALSE,sep=";",
+ colClasses=c('character',rep('numeric',3)),dec=".",
+ col.names=c("mois","an","evol","coll.v","cum.v","coll.m","cum.m"))
> lait=ts(lait2$coll.v/1000,start=c(1979,1),frequency = 12)
> head(lait,3)
```

```
[1] 1597.318 1594.601 1959.054
```

`lait` est la série de la collecte mensuelle de lait, exprimée en millions de litres, de janvier 1979 à janvier 1990, et nous avons imprimé les collectes de janvier à mars 1979.
Examinons la série par une décomposition élémentaire en tendance, saisonnalité et erreur (fig. 11.1) où le trait vertical marque janvier 1984, date de l'introduction des quota laitiers en France.

```
> decomp=decompose(lait)
> plot(decomp)
> abline(v=1984)
```

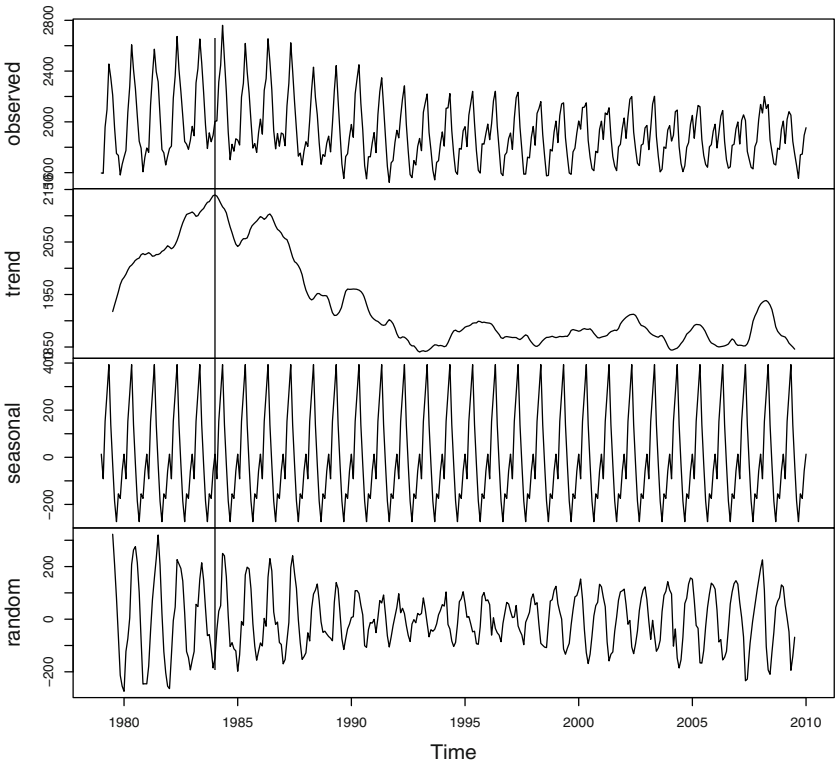

Fig. 11.1 – Collecte mensuelle de lait en France, décomposition additive de la série.

Tendance. On observe l'évolution de la série à moyen terme sur le chronogramme du trend : elle semble croître jusqu'à la mise en place des quota, sa croissance ralentit ensuite, puis la série décroît jusque vers la fin de 1994, pour se stabiliser ensuite. Mais cette évolution est accompagnée de beaucoup de bruit. On peut au moins noter que la série décroît jusqu'au milieu de 1991 (voir le calcul de la date ci-dessous) puis se stabilise à un niveau moyen de l'ordre de 1900. On trouve la date du minimum de la série par :

```
> num=which.min(lait)
> t.lait=time(lait)
> cat('temps de la collecte minimale : ',t.lait[num],'\n')
temps de la collecte minimale :  1991.667
```

exprimée en année et fraction d'année qui correspond à août 1991.

Erreur. On observe enfin que l'erreur résiduelle est d'autant plus variable que le niveau de production est élevé. On n'essaiera pas de prendre en compte cette

hétéroscédasticité, qui s'explique sans doute en grande partie de la façon suivante : la collecte est la somme des collectes par vache. Le progrès technique n'a augmenté cette collecte que marginalement sur la période considérée ; la collecte est donc d'autant plus élevée qu'elle concerne davantage de vaches. Et l'aléa observé est d'autant plus variable que le troupeau est nombreux.

Saisonnalité. On voit clairement la saisonnalité, d'origine agro-biologique, sur les chronogrammes de la série et de la composante saisonnière, mais examinons-la de plus près à l'aide d'autres graphiques.

Month plot. Nous dessinons le month plot par :

```
> monthplot(lait,xlab='mois')
```

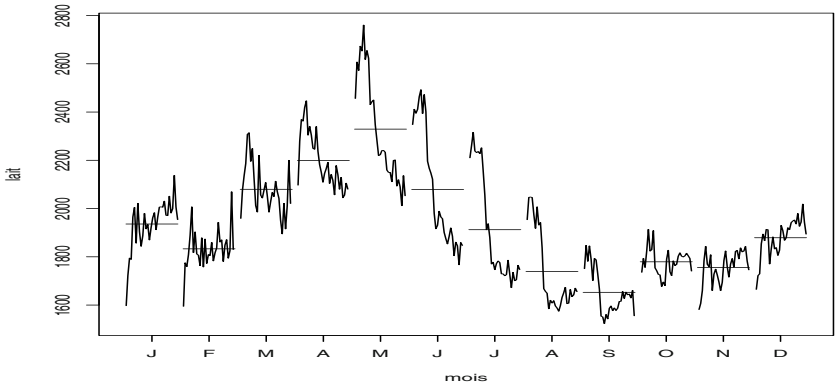

Fig. 11.2 – Month plot de la collecte de lait : 1980-2010.

La courbe (fig. 11.2) dessinée par les moyennes des 12 mois sur le month plot représente le cycle de la vache, un aspect déterministe de la saisonnalité. Le month plot de la série est plus proche de celui de la température à Nottingham Castle que de celui d'un SARMA (fig. 1.11). On observe aussi que la baisse due aux quota est très sensible sur les mois d'avril à septembre, mais pour les autres mois, on ne constate pas de baisse et même, de décembre à février, une hausse. Un graphique des séries par année distinguant les séries avant ou après quota est susceptible de nous éclairer. On a appelé year plot un tel graphique (cf. section 1.2).

Year plot. Dans notre cas, le nombre élevé d'années rendrait le year plot peu lisible. Aussi remplaçons-nous les séries annuelles après quota par les séries 1er et 3^{e} quartile. Dans le code ci-dessous, nous représentons la série en une matrice, à raison d'une colonne par année. Nous fabriquons les séries quartiles des années après quota. Nous complétons l'année par une 13^{e} valeur, manquante, qui donne la place pour l'écriture des légendes. Enfin, par `matplot()`, nous représentons les colonnes en choisissant deux types de trait, selon qu'on est avant ou après

l'introduction des quota. Les années avant quota figurent en trait plein et les quartiles des années après quota, en pointillé.

```
> ans = 1979:2009
> freq=12
> y.m=as.matrix(window(lait,start=c(ans[1],1),end=c(ans[1],freq)))
> for(i in ans[-1]){
+ y.m=cbind(y.m,as.matrix(window(lait,start=c(i,1),end=c(i,freq))))}
> q2=t(apply(y.m[,-(1:5)],1,quantile,c(.25,.75)))
> ypl=cbind(y.m[,1:5],q2)
> colnames(ypl)=c(unlist(lapply(ans[1:5],toString)),'q.25','q.75')
```

Dans le code ci-dessus, `y.m` est la matrice de la collecte avec une colonne par année et une ligne par mois. Pour calculer les quartiles par mois de la collecte des années 1984 et suivantes, on applique `quantile()` à chaque ligne de la sous-matrice de `y.m` obtenue en éliminant les 5 premières colonnes. Ensuite, par le code ci-dessous, on superpose sur un même graphique les chronogrammes des cinq premières années et les quartiles des années suivantes. Pour écrire une légende par chronogramme, on prévoit un 13e mois d'ordonnées manquantes, où nous positionnerons un texte par année. Ce texte devrait être positionné à côté de la donnée du mois de décembre. Pour éviter des chevauchements, nous modifions par tâtonnement les positions verticales d'un facteur proche de 1 qui ne modifie pas l'ordre des valeurs des 12es mois.

```
> matplot(1:(freq+1),rbind(ypl,NA),xlab='mois',ylab='séries',
+       lty= c(rep(1,5),rep(2,2)), type='o', lwd=2,
+       col=c(gray(0:4 / 8),rep('black',2)),pch=c(21:25,NA,NA))
> x.t=freq; y.t=ypl[freq,]
> y.t=ypl[freq,]*c(.97,.98,1.011,.99,1.011,1.009,1.014)
> text(x.t,y.t,colnames(ypl),pos=4,cex=.8,col=c(gray(0:4/8),rep('black',2)))
```

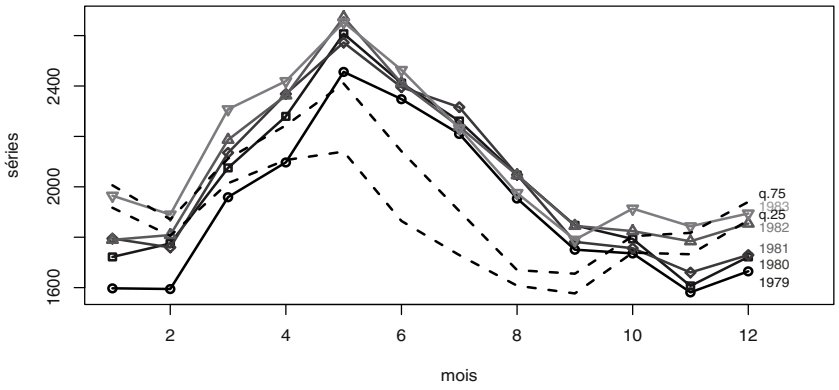

Fig. 11.3 – Year plot de la collecte de lait. Années avant quota, séries annuelles en trait plein ; années après quota, séries des quartiles en tiret.

On observe (fig. 11.3) que d'octobre à mars, les quantités collectées avant et après quota se ressemblent. La baisse de collecte n'est sensible que d'avril à septembre. Le month plot laissait prévoir un tel constat.

Lag plot. Enfin, examinons le lag plot de la série (fig. 11.4) :

```
> lag.plot(rev(lait),set=c(1:12),pch="+",col="black")
```

Sa forme, très allongée au retard 12, est un indice de non-stationnarité de la série. Cependant, il n'est pas aussi simple à interpréter que celui de `nottem` (fig. 1.9). On note cependant des similitudes aux retards 6 et 8 notamment. Si l'on sépare les lag plots de la série finissant en décembre 1985 de celle commençant en janvier 2004, on obtient des formes ressemblant au lag plot de `nottem` (SiteST).

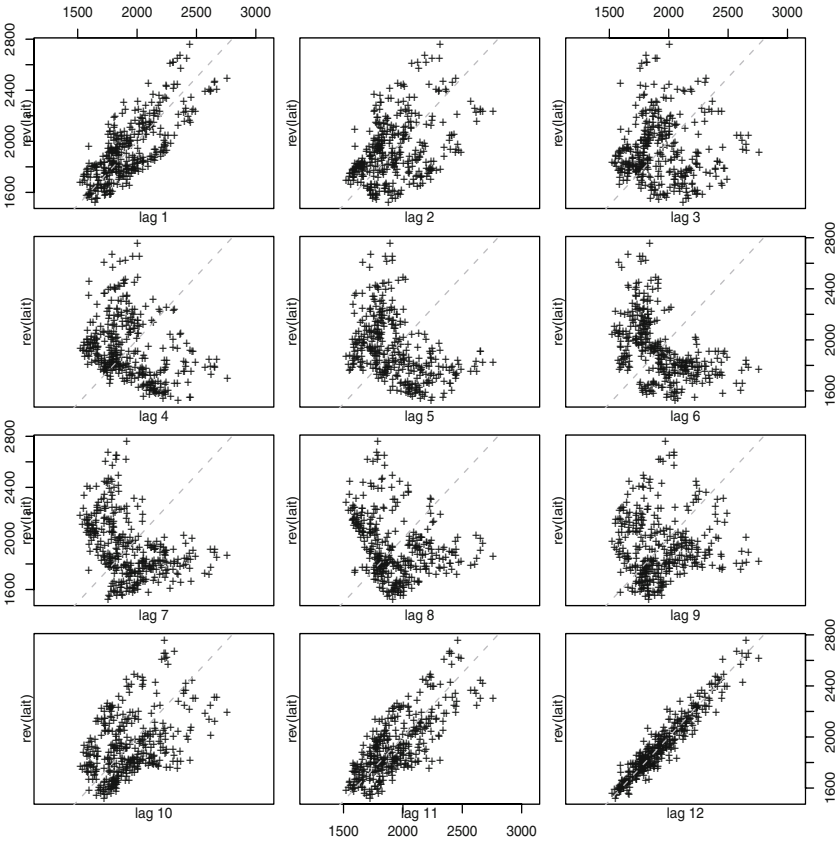

Fig. 11.4 – Lag plot de lait.

Normalité. En vue de la modélisation de la série par un modèle ARIMA, nous examinons sa normalité par un test de D'Agostino dans **fBasics** dans sa version omnibus.

```
> require(fBasics)
> aa=dagoTest(lait)
```

La structure de la sortie nous indique que, si nous nous intéressons uniquement à la p-value du test omnibus, il suffit de récupérer la composante `aa@test$p.value[1]`. Comme elle est très faible sur la série brute, nous essayons les transformations log(.) et $\sqrt{.}$.

```
> a1=aa@test$p.value[1]
> a2=dagoTest(log(lait))@test$p.value[1]
> a3=dagoTest(lait^.5)@test$p.value[1]
> aa=as.matrix(c(a1,a2,a3))
> rownames(aa)=c('Série brute','log','racine')
> colnames(aa)='p-value'
```

Les p-values sont stockées dans `aa`, reproduite dans le tableau 11.1, ci-dessous.

Tableau 11.1 – Lait - Test de normalité de D'Agostino.

	p-value
Série brute	0.000000
log	0.000293
racine	0.000005

Aucune p-value n'est raisonnablement élevée, la transformation log donnant la moins faible p-value. Nous envisageons alors une transformation normalisante de Box-Cox à l'aide de `powerTransform()` de **car** et calculons la série transformée

```
> require(car)
> (ptr=powerTransform(lait)$lambda)

     lait
-1.811208

> lait.tr =(lait^ptr - 1)/ptr
> dagoTest(lait.tr)@test$p.value[1]

Omnibus  Test
  0.007810533
```

Nous constatons que la p-value est certes un peu plus élevée que dans la transformation log, mais la série transformée a une très faible variabilité et sa modélisation se révèle difficile. Aussi, nous retenons finalement la série transformée en log.
Dans la suite du chapitre nous essaierons de modéliser la log-série avant quota. Un modèle SARMA, donc stationnaire, semble convenir. L'étude de la collecte après la mise en place des quota, sur la partie de série dont la fluctuation semble stabilisée, nous donne un modèle non stationnaire (section 11.3). Le changement radical de

modèle qui s'impose ainsi entre les deux périodes incite à revoir la modélisation du début. Le month plot de la série ressemble à celui observé pour la température à Nottingham Castle, série typiquement non stationnaire (fig. 1.11). On a vu également qu'une fonction périodique s'ajuste bien à cette température. Nous reprendrons donc la modélisation par des fonctions trigonométriques en y ajoutant (section 11.5), au moins de façon schématique, l'intervention que représente l'installation des quota.

11.2 Modélisation avant 1984

Commençons par examiner les fonctions d'autocorrélation de la série avant 1984 (fig. 11.5). La décroissance de l'ACF de 12 en 12 n'est pas très rapide et la valeur au retard 1 est de l'ordre de 0.7, ce qui ne suggère pas une racine unité. La PACF en 12 est faiblement significative, ce qui est inattendu en présence de saisonnalité marquée, comme on l'a noté précédemment et comme on l'observe sur l'ACF.

```
> log.lait=log(lait)
> lait.avant=window(log.lait,end=c(1983,12))
> xy.acfb(lait.avant,lag.max=26,numer=FALSE)
```

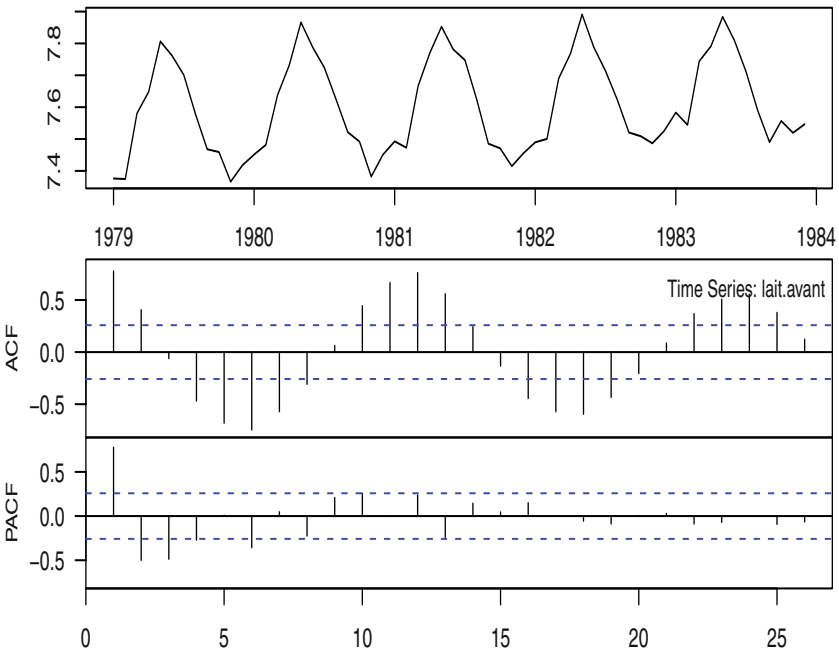

Fig. 11.5 – ACF et PACF de la collecte : 1979-1984.

Nous allons nous aider de `auto.arima()` de **forecast** pour modéliser la série en

gardant les valeurs par défaut des paramètres. La fonction choisit alors les ordres de différenciation simple et saisonnière, et autorise une autorégression et une moyenne mobile d'ordre 5 au plus et 2 dans les termes saisonniers :

```
> (mod0=auto.arima(lait.avant))

...
Coefficients:
         ar1   sar1    sar2  intercept
      0.8064  0.642  0.3229     7.5933
s.e.  0.0799  0.153  0.1575     0.1292

sigma^2 estimated as 0.0007318:  log likelihood = 115.81
AIC = -221.62   AICc = -220.51   BIC = -211.15

> ret=c(6,12,18,24,30)
> t(Box.test.2(residuals(mod0),ret,type ="Ljung-Box",decim=2,fitdf=3))

        [,1] [,2]  [,3] [,4] [,5]
Retard  6.00 12.0 18.00 24.0 30.0
p-value 0.07  0.1  0.21  0.5  0.7

> t_stat(mod0)

            ar1      sar1     sar2 intercept
t.stat 10.08703 4.194998 2.049746  58.75386
p.val   0.00000 0.000027 0.040389   0.00000
```

La p-value du test de blancheur est faible à l'ordre 6 et l'examen de l'ACF des résidus révèle qu'ils sont significativement autocorrélés négativement au décalage 5, ce qui entraîne la faible p-value pour le test de la nullité des autocorrélations aux ordres 1 à 6. La série étant courte, nous augmentons parcimonieusement le nombre de paramètres, en permettant un coefficient autorégressif puis moyenne mobile, d'ordre 5. L'algorithme d'optimisation ne converge dans aucun de ces deux cas. Tenant compte de la faible normalité de la série, nous abandonnons la méthode du maximum de vraisemblance au profit de la méthode CSS avec une autorégression saisonnière à l'ordre 1. Après quelques tentatives, nous retenons un coefficient moyenne mobile au retard 5 :

```
> (mod1b=Arima(lait.avant,order=c(1,0,5),seasonal=list(order=c(1,0,0)),
+   fixed=c(NA,rep(0,4),rep(NA,3)),method="CSS"))

...
Coefficients:
         ar1  ma1  ma2  ma3  ma4      ma5    sar1
      0.5880    0    0    0    0  -0.3043  0.9437
s.e.  0.1031    0    0    0    0   0.1397  0.0394
      intercept
         8.0009
s.e.     0.2969

sigma^2 estimated as 0.0006355:  part log likelihood = 135.7
```

```
> t(Box.test.2(residuals(mod1b),ret,type ="Ljung-Box",decim=2,fitdf=4))

        [,1]  [,2]  [,3]  [,4]  [,5]
Retard  6.00 12.00 18.00 24.00 30.00
p-value 0.56  0.38  0.25  0.54  0.64

> t_stat(mod1b)

            ar1       ma5     sar1 intercept
t.stat 5.703244 -2.178515 23.95409  26.94445
p.val  0.000000  0.029368  0.00000   0.00000
```

Nous obtenons maintenant des résultats satisfaisants en terme de blancheur des résidus et de significativité des coefficients. Ainsi, la normalité de la série à modéliser n'étant pas assurée, l'algorithme d'estimation par la méthode du maximum de vraisemblance qui suppose cette normalité ne pouvait converger. Le test omnibus de D'Agostino :

```
> aa= dagoTest(residuals(mod1b))
```

donne une p-value de 0.9 ; on peut considérer que les résidus sont normalement distribués. Maintenant que nous avons obtenu un modèle satisfaisant, dont les résidus sont normalement distribués, essayons d'estimer ce modèle par maximum de vraisemblance.

```
> (mod1bm=Arima(lait.avant,order=c(1,0,5),seasonal=list(order=c(1,0,0)),
+    fixed=c(NA,rep(0,4),rep(NA,3))))

...
Coefficients:
         ar1  ma1  ma2  ma3  ma4      ma5    sar1
      0.8038    0    0    0    0  -0.2819  0.9492
s.e.  0.0796    0    0    0    0   0.1296  0.0214
      intercept
         7.5930
s.e.     0.0869

sigma^2 estimated as 0.000744:  log likelihood = 116.39
AIC = -222.78   AICc = -219.18   BIC = -203.93

> t(Box.test.2(residuals(mod1bm),ret,type="Ljung-Box",decim=2,fitdf=4))

        [,1]  [,2]  [,3]  [,4]  [,5]
Retard  6.00 12.00 18.00 24.00 30.00
p-value 0.41  0.31  0.31  0.62  0.77

> t_stat(mod1bm)

            ar1       ma5     sar1 intercept
t.stat 10.10262 -2.175119 44.39425    87.372
p.val   0.00000  0.029621  0.00000     0.000
```

La variance du bruit est 0.0007 ; elle montre une diminution importante par rapport à celle de la série : 0.0217. Finalement, le modèle ajusté par maximum de vraisemblance avant la mise en place des quota est

$$y_t = 7.593 + \frac{1 - 0.2819\mathrm{B}^5}{(1 - 0.8038\mathrm{B})(1 - 0.9492\mathrm{B}^{12})} z_t \quad (11.1a)$$

$$z_t \sim \mathrm{BBN}(0, 0.000744). \quad (11.1b)$$

Superposons sur un même graphique la série et la bande de confiance à 80% (fig. 11.6) :

```
> demi.b=qnorm(.9)*mod1bm$sigma2^.5
> b.inf=fitted(mod1bm)-demi.b
> b.sup=fitted(mod1bm)+demi.b
> dans=(b.inf<lait.avant)&(lait.avant<b.sup)
> plot.ts(cbind(lait.avant,b.inf,b.sup),plot.type='single',xlab='temps',
+ ylab="Log collecte",lty=c(1,2,2))
```

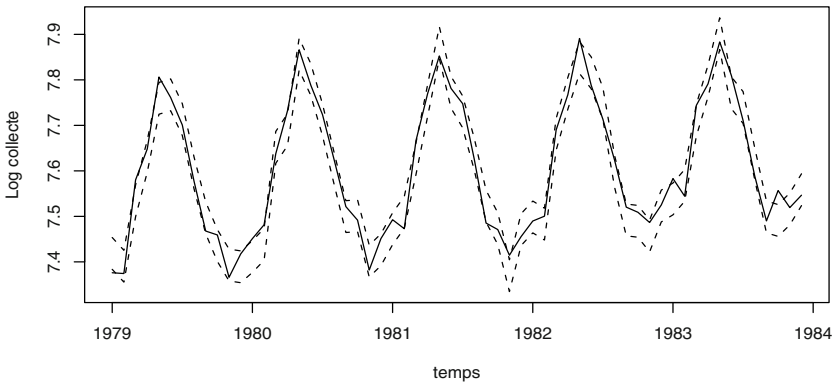

Fig. 11.6 – Collecte et bande de confiance à 80% : 1979-1984.

La bande de confiance n'est pas trop large et le pourcentage de points dans la bande 76.67% reste très proche des 80% théoriques. Nous pourrions nous servir de ce modèle pour prédire la collecte de l'année suivante, en l'absence de quota.

```
> pred56=forecast(mod1bm,h=12,level=80)
> et.pred=(pred56$upper-pred56$mean)/qnorm(.9)
```

Il est utile de se souvenir que `et.pred` contient les écarts types de la prévision conditionnellement aux observations, $1, 2, \cdots, 12$ instants avants et donc que `et.pred[1]` coïncide avec `mod1$sigma2^.5`. Un calcul semblable est effectué pour le trafic passager à Blagnac à la section 8.4.

En résumé, l'exploration de la série a révélé sa non-stationnarité, mais un modèle stationnaire s'ajuste bien à son début, essentiellement parce que la série en question est courte : 4 ans. On peut s'attendre que pour une série plus longue, la non-stationnarité doive être prise en compte. C'est ce que nous allons voir maintenant.

11.3 Essai de modélisation après l'introduction des quota

Nous considérons la partie de la série qui, à l'examen du graphique, est stabilisée : c'est la sous-série commençant en 1995 (SiteST). Nous essayons d'y ajuster un modèle de la même famille que pour la série avant.

```
> apres95=window(log.lait,start=c(1995,1))
> (mod2=Arima(apres95,order=c(1,0,0),seasonal=list(order=c(1,0,0))))

...
Coefficients:
         ar1    sar1  intercept
      0.6613  0.9440     7.5279
s.e.  0.0551  0.0178     0.0496

sigma^2 estimated as 0.000522:  log likelihood = 413.55
AIC = -819.1   AICc = -818.87   BIC = -806.31

> t(Box.test.2(residuals(mod2),ret,type="Ljung-Box",decim=2,fitdf=2))

        [,1] [,2]  [,3]  [,4]  [,5]
Retard  6.00   12 18.00 24.00 30.00
p-value 0.91    0  0.01  0.01  0.05
```

Le résidu montre une autocorrélation très significative au retard 12, ce qui était attendu vu le lag plot de la série complète. Il faut donc différencier saisonnièrement la série. Un premier essai montre que la dérive n'est pas significative et nous ne l'incluerons donc pas dans le modèle :

```
> (mod2=Arima(apres95,order=c(1,0,0),seasonal=list(order=c(1,1,0)),
+                include.drift=FALSE))

Series: apres95
ARIMA(1,0,0)(1,1,0)[12]
...
Coefficients:
         ar1     sar1
      0.6721  -0.4174
s.e.  0.0574   0.0755

sigma^2 estimated as 0.0004506:  log likelihood = 409.82
AIC = -813.64   AICc = -813.5   BIC = -804.25

> t(Box.test.2(residuals(mod2),ret,type="Ljung-Box",decim=2,fitdf=2))

        [,1]  [,2]  [,3]  [,4] [,5]
Retard  6.00 12.00 18.00 24.00 30.0
p-value 0.98  0.25  0.41  0.12  0.3

> t_stat(mod2,decim=2)
```

```
         ar1  sar1
t.stat 11.71 -5.53
p.val   0.00  0.00
```

L'ajustement est satisfaisant, mais il n'y a aucune unité entre les modélisations des deux périodes. Essayons un modèle SARIMA pour l'ensemble de la série.

11.4 Modélisation SARIMA de toute la série

Nous essayons le modèle qui convenait à partir de 1995.

```
> mod2.tot=Arima(log.lait,order=c(1,0,0),seasonal=list(order=c(1,1,0)),
+   include.drift=FALSE)
> summary(mod2.tot)

...
Coefficients:
         ar1     sar1
      0.7457  -0.3777
s.e.  0.0361   0.0506

sigma^2 estimated as 0.0005793:  log likelihood = 831.83
AIC = -1657.65   AICc = -1657.58   BIC = -1645.98

In-sample error measures:
           ME          RMSE           MAE           MPE
-0.0001038617  0.0237176454  0.0184312129 -0.0012078560
         MAPE          MASE
 0.2435722789  0.2609638975

> t(Box.test.2(residuals(mod2.tot),ret,type="Ljung-Box",decim=2,fitdf=2))

        [,1]  [,2] [,3]  [,4]  [,5]
Retard   6.0 12.00 18.0 24.00 30.00
p-value  0.9  0.09  0.2  0.01  0.01

> t_stat(mod2.tot,decim=2)

         ar1  sar1
t.stat 20.66 -7.46
p.val   0.00  0.00
```

L'ajustement est moyennement satisfaisant. Examinons les ACF et PACF des résidus.

```
> acf2y(residuals(mod2.tot),numer=FALSE)
```

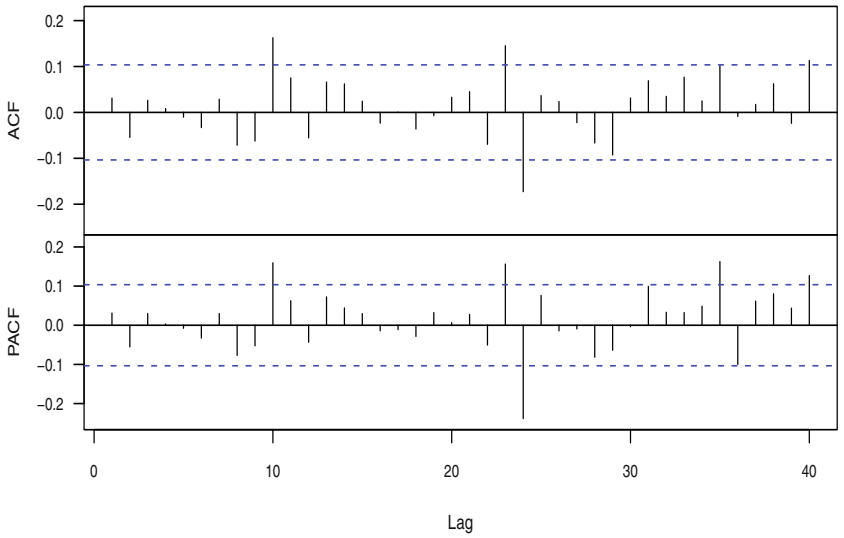

Fig. 11.7 – SARIMA(1,0,0)(1,1,0), ACF et PACF des résidus.

Nous constatons (fig. 11.7) qu'il reste de la PACF significative en 24 et ajoutons donc un terme autorégressif saisonnier d'ordre 2 pour la prendre en compte.

```
> mod3.tot=Arima(log.lait,order=c(1,0,0),
+   seasonal=list(order=c(2,1,0)),include.drift=FALSE)
> summary(mod3.tot)

...
Coefficients:
         ar1     sar1     sar2
      0.7549  -0.4423  -0.1786
s.e.  0.0357   0.0540   0.0549

sigma^2 estimated as 0.0005617:  log likelihood = 837
AIC = -1666.01   AICc = -1665.9   BIC = -1650.45

In-sample error measures:
           ME          RMSE           MAE           MPE
-0.0001013276  0.0233558789  0.0181487242 -0.0010670996
         MAPE          MASE
 0.2398023863  0.2569641955

> t(Box.test.2(residuals(mod3.tot),ret,type ="Ljung-Box",decim=2,fitdf=3))

        [,1]  [,2]  [,3]  [,4]  [,5]
Retard   6.0 12.00 18.00 24.00 30.00
p-value  0.7  0.03  0.04  0.01  0.02

> t_stat(mod3.tot,decim=2)
```

```
         ar1  sar1  sar2
t.stat 21.13 -8.19 -3.25
p.val   0.00  0.00  0.00
```

Le test de blancheur des résidus n'est guère satisfaisant.

```
> acf2y(residuals(mod3.tot),numer=FALSE)
```

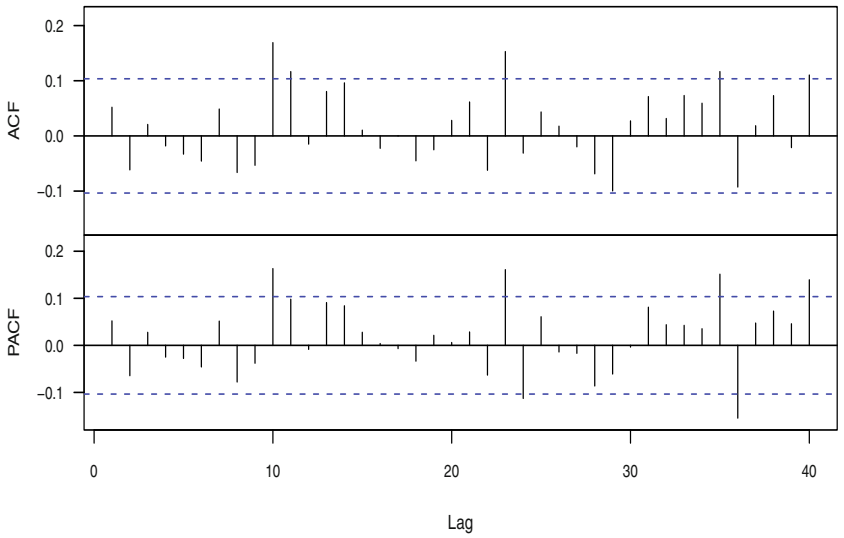

Fig. 11.8 – SARIMA(1,0,0)(2,1,0), ACF et PACF des résidus.

L'ACF des résidus (fig. 11.8) montre encore des autocorrélations significatives aux retards 10 et 23 notamment.
Faisons le point de la situation. Nous avons du mal à trouver un modèle SARIMA pour l'ensemble de la série log-lait. Pour `nottem`, nous avons pu ajuster d'une part un modèle avec saisonnalité stochastique, d'autre part un modèle avec saisonnalité déterministe. Nous essayons maintenant un tel modèle pour notre série.

11.5 Modélisation ARMAX de la collecte

Nous allons procéder en deux étapes : faire par MCO une première estimation de la moyenne de la série, puis identifier la dynamique des résidus. Ensuite, on estimera simultanément la moyenne et la dynamique des résidus.

11.5.1 Modélisation MCO

Nous commençons par régresser par MCO la production sur les fonctions trigonométriques $\cos(\omega t)$ et $\sin(\omega t)$ de période 12, $\omega = 2\pi\frac{k}{12},\ k = 1, 2, \cdots, 6$, comme

on les a introduites (chap. 9, section 9.2.2) et, pour exprimer l'intervention survenant en janvier 1984, sur l'indicatrice du temps avant `ind.av` et sur `ind.apr = 1-ind.av`. Nous devons évidemment supprimer la constante de la liste des régresseurs.

```
> f=t(as.matrix(1:6))/12
> num.temps=as.matrix(1:length(lait))
> temps=time(log.lait)
> ind.av=ifelse(as.numeric(temps)<1984,1,0)
> ind.apr=1-ind.av
> xmat0=cbind(cos(2*pi*num.temps%*%f),sin(2*pi*temps%*%f))[,-12]
> xmat0=as.data.frame(xmat0)
> xmat0b=cbind(ind.av,ind.apr,xmat0)
> colnames(xmat0b)=c('ind.av','ind.apr','cos1','cos2','cos3','cos4','cos5',
+                          'cos6','sin1','sin2','sin3','sin4','sin5')
```

On voit que `ind.av` prend la valeur 1 avant l'intervention des quota et 0 ensuite. Comme $\sin(2\pi t) = 0$, on ne retient pas la colonne correspondante dans la fabrication de la matrice des régresseurs, `xmat0`. On effectue la régression MCO annoncée

```
> mod1=lm(log.lait~ind.av+ind.apr+cos1+cos2+cos3+cos4+cos5+cos6+sin1+
+                    sin2+sin3+sin4+sin5-1,data=xmat0b)
> summary(mod1)

Call:
lm(formula = log.lait ~ ind.av + ind.apr + cos1 + cos2 + cos3 +
    cos4 + cos5 + cos6 + sin1 + sin2 + sin3 + sin4 + sin5 - 1,
    data = xmat0b)

Residuals:
      Min        1Q    Median        3Q       Max
-0.219615 -0.074141  0.007864  0.073968  0.273001

Coefficients:
          Estimate Std. Error  t value Pr(>|t|)
ind.av   7.6044230  0.0146654  518.527  < 2e-16 ***
ind.apr  7.5500605  0.0061159 1234.503  < 2e-16 ***
cos1    -0.0664815  0.0078649   -8.453 7.15e-16 ***
cos2     0.0364069  0.0078751    4.623 5.28e-06 ***
cos3     0.0098468  0.0078803    1.250    0.212
cos4    -0.0039344  0.0078751   -0.500    0.618
cos5     0.0087163  0.0078646    1.108    0.268
cos6    -0.0055169  0.0055648   -0.991    0.322
sin1    -0.0006275  0.0083743   -0.075    0.940
sin2    -0.0066383  0.0080082   -0.829    0.408
sin3    -0.0032882  0.0079299   -0.415    0.679
sin4     0.0093093  0.0078668    1.183    0.237
sin5    -0.0018149  0.0078717   -0.231    0.818
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.1075 on 360 degrees of freedom
Multiple R-squared: 0.9998,        Adjusted R-squared: 0.9998
F-statistic: 1.419e+05 on 13 and 360 DF,  p-value: < 2.2e-16
```

```
> resid.mod1=ts(residuals(mod1),start=c(1979,1),frequency=12)
```

Le R2 ajusté est supérieur à 99%. La régression sur ces fonctions trigonométriques semble être une bonne voie. Nous voyons qu'un grand nombre des coefficients des fonctions trigonométriques ne sont vraisemblablement pas significatifs, mais comme les résidus sont sans doute autocorrélés, nous ne simplifions pas immédiatement le modèle ajusté par MCO.

11.5.2 Identification des résidus de l'ajustement MCO

Nous examinons l'autocorrélation des résidus (fig. 11.9). La PACF des résidus a une valeur très significative en 12 et 24, ce qui incite à les ajuster par un SARMA avec autorégression saisonnière jusqu'à l'ordre 2.

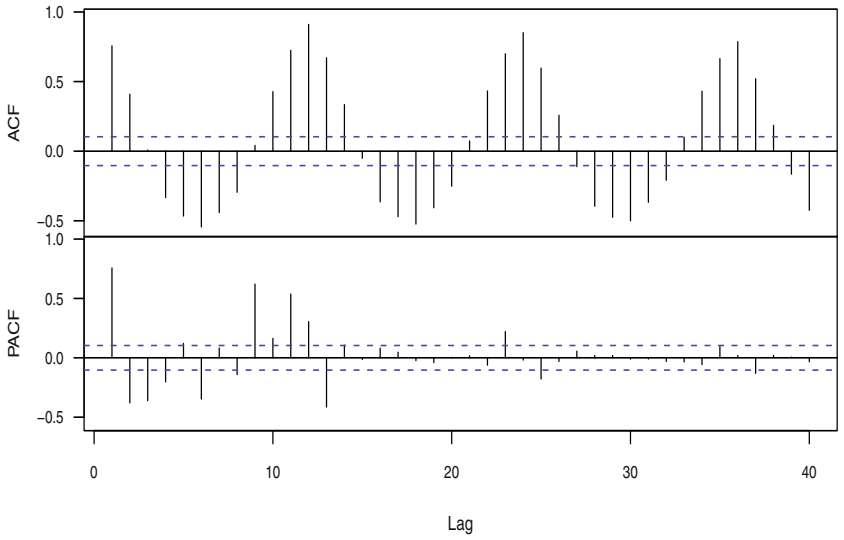

Fig. 11.9 – Lait - ACF et PACF du résidu après ajustement MCO.

```
> (modar1=Arima(resid.mod1,order=c(1,0,0),seasonal=list(order=c(2,0,0))))
```

```
Series: resid.mod1
ARIMA(1,0,0)(2,0,0)[12] with non-zero mean
...
Coefficients:
         ar1    sar1    sar2  intercept
```

```
      0.7364  0.5892  0.3631    -0.0254
s.e.  0.0360  0.0499  0.0507     0.0596

sigma^2 estimated as 0.0005838:  log likelihood = 845.68
AIC = -1681.37   AICc = -1681.21   BIC = -1661.76
```

```
> t(Box.test.2(residuals(modar1),ret,fitdf=3,decim=4))
```

```
          [,1]  [,2]   [,3]    [,4]    [,5]
Retard  6.0000 12.00 18.000 24.0000 30.0000
p-value 0.5346  0.02  0.048  0.0029  0.0058
```

```
> acf2y(residuals(modar1),numer=FALSE)
```

`modar1` n'est pas une modélisation satisfaisante des résidus MCO. On ajoute des termes autorégressifs jusqu'à l'ordre 9 :

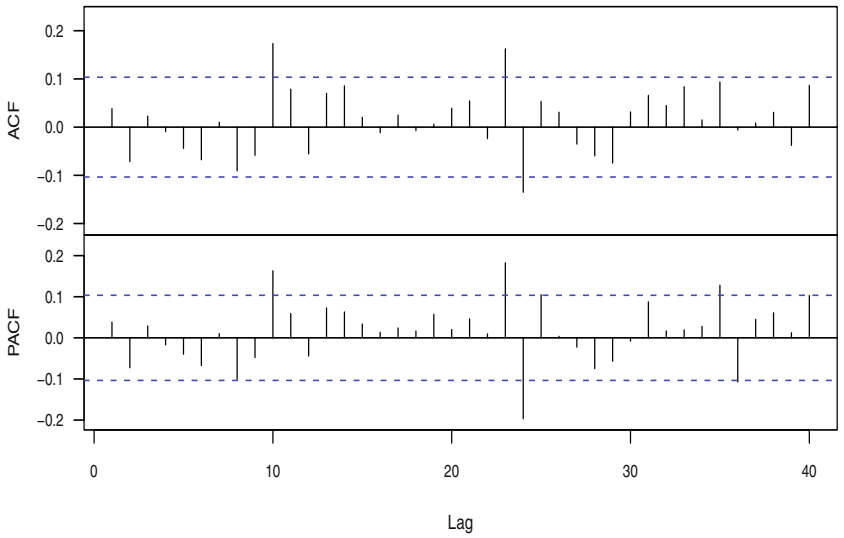

Fig. 11.10 – Lait - ACF et PACF du résidu après ajustement SARMA.

```
> modar1b=Arima(resid.mod1,order=c(9,0,0),seasonal=list(order =c(2,0,0)))
> t(Box.test.2(residuals(modar1b),ret,fitdf=11,decim=4))
```

```
          [,1]    [,2]    [,3]    [,4]    [,5]
Retard  6.0000 12.0000 18.0000 24.0000 30.0000
p-value 0.9919  0.4949  0.5898  0.0219  0.0361
```

```
> t_stat(modar1b)
```

```
            ar1       ar2      ar3       ar4      ar5
t.stat 15.40724 -1.417109 1.204186 -0.806183 0.019901
p.val   0.00000  0.156451 0.228518  0.420138 0.984123
```

```
            ar6      ar7       ar8      ar9     sar1
t.stat -1.244364 1.326131 -2.056366 3.042599 10.81331
p.val   0.213366 0.184796  0.039747 0.002345  0.00000
           sar2 intercept
t.stat 7.145609 -0.522880
p.val  0.000000  0.601057
```

La blancheur du résidu est satisfaisante, mais nous voyons, comme on pouvait s'y attendre, que les paramètres autorégressifs d'ordres 2 à 7 ne sont pas significatifs. Nous les contraignons à 0. De même, nous contraignons la moyenne à 0 puisqu'on analyse le résidu d'une régression MCO, certes sans constante, mais dont les régresseurs contiennent deux variables qui somment à une constante :

```
> (modar1c=Arima(resid.mod1,order=c(9,0,0),seasonal=list(order=c(2,0,0)),
+  fixed =c(NA,rep(0,6),rep(NA,4)),include.mean=FALSE))

...
Coefficients:
         ar1  ar2  ar3  ar4  ar5  ar6  ar7      ar8     ar9
      0.7394    0    0    0    0    0    0  -0.1121  0.1565
s.e.  0.0351    0    0    0    0    0    0   0.0529  0.0539
        sar1    sar2
      0.5627  0.3818
s.e.  0.0506  0.0509

sigma^2 estimated as 0.0005727:  log likelihood = 849.74
AIC = -1687.49   AICc = -1686.62   BIC = -1640.43

> t(Box.test.2(residuals(modar1c),ret,fitdf=5,decim=4))

          [,1]    [,2]   [,3]    [,4]    [,5]
Retard  6.0000 12.0000 18.000 24.0000 30.0000
p-value 0.5902  0.1303  0.242  0.0134  0.0212
```

Le résultat est assez satisfaisant, bien que la PACF à l'ordre 24 demeure significative ; elle ne pourrait être modélisée que par un terme MA saisonnier d'ordre 2. Mais nous allons affiner la modélisation directement par la modélisation ARMAX de la série `log.lait` car c'est bien ce qui nous intéresse.

11.5.3 Modélisation simultanée de la moyenne et de l'erreur

Nous partons évidemment du modèle retenu pour les résidus et nous estimons simultanément la moyenne de la collecte et la dynamique de l'erreur.

```
> (modar1X=Arima(log.lait,order=c(9,0,0),seasonal=list(order=c(2,0,0)),
+ include.mean=FALSE,xreg=xmat0b,fixed=c(NA,rep(0,6),rep(NA,4),rep(NA,13))))

...
Coefficients:
         ar1  ar2  ar3  ar4  ar5  ar6  ar7      ar8     ar9
      0.7150    0    0    0    0    0    0  -0.1041  0.1340
```

```
s.e.  0.0366    0    0    0    0    0    0   0.0532  0.0543
        sar1    sar2  ind.av  ind.apr     cos1    cos2
      0.5810  0.3660  7.5342   7.5451  -0.0931  0.0296
s.e.  0.0511  0.0512  0.0564   0.0558   0.0441  0.0224
        cos3     cos4    cos5     cos6    sin1     sin2
      0.0092  -0.0019  0.0103  -0.0048  0.0141  -0.0044
s.e.  0.0144   0.0146  0.0127   0.0073  0.0097   0.0050
         sin3    sin4     sin5
      -0.0042  0.0087  -0.0001
s.e.   0.0038  0.0034   0.0035

sigma^2 estimated as 0.0005465:  log likelihood = 858.38
AIC = -1678.76   AICc = -1675.02   BIC = -1580.72
```

```
> t(Box.test.2(residuals(modar1X),ret,fitdf=18,decim=4))
```

```
          [,1]    [,2]    [,3]    [,4]    [,5]
Retard  6.0000 12.0000 18.0000 24.0000 30.0000
p-value 0.7877  0.1896  0.2922  0.0238  0.0325
```

Le test de blancheur est satisfaisant. Examinons maintenant les t-statistiques :

```
> t_stat(modar1X)
```

```
            ar1       ar8      ar9     sar1    sar2
t.stat 19.52154 -1.956248 2.467557 11.37746 7.14284
p.val   0.00000  0.050436 0.013604  0.00000 0.00000
         ind.av  ind.apr      cos1     cos2     cos3
t.stat 133.4955 135.1274 -2.109545 1.324809 0.639705
p.val    0.0000   0.0000  0.034898 0.185234 0.522365
            cos4     cos5      cos6     sin1      sin2
t.stat -0.133048 0.809184 -0.656068 1.446351 -0.877125
p.val   0.894155 0.418409  0.511780 0.148079  0.380419
            sin3     sin4      sin5
t.stat -1.111005 2.573797 -0.018105
p.val   0.266566 0.010059  0.985555
```

On peut supprimer les régresseurs : `cos2`, `cos3`, `cos4`, `cos5`, `cos6`, `sin1`, `sin2`, `sin3` et `sin5`. Nous formons la matrice des régresseurs retenus plutôt que de corriger le vecteur des contraintes `fixed=` de façon un peu compliquée.

```
> xmat0c=xmat0b[,c(1:3,12) ]
> (modar2X=Arima(log.lait,order=c(9,0,0),seasonal=list(order=c(2,0,0)),
+  include.mean=FALSE,xreg=xmat0c,fixed=c(NA,rep(0,6),rep(NA,8))))
```

```
...
Coefficients:
        ar1  ar2  ar3  ar4  ar5  ar6  ar7      ar8     ar9
      0.7275    0    0    0    0    0    0  -0.0957  0.1267
s.e.  0.0360    0    0    0    0    0    0   0.0531  0.0537
        sar1    sar2  ind.av  ind.apr     cos1    sin4
      0.5863  0.3683  7.5365   7.5440  -0.0954  0.0086
```

```
s.e.  0.0509  0.0513  0.0663   0.0658   0.0493  0.0035

sigma^2 estimated as 0.0005539:  log likelihood = 854.96
AIC = -1689.93   AICc = -1688.4   BIC = -1627.18

> t(Box.test.2(residuals(modar2X),ret,fitdf=9,decim=4))

          [,1]    [,2]    [,3]    [,4]    [,5]
Retard  6.0000 12.0000 18.0000 24.0000 30.0000
p-value 0.8676  0.2894  0.4315  0.0342  0.0415

> t_stat(modar2X)

            ar1       ar8      ar9     sar1     sar2
t.stat 20.21156 -1.800473 2.360520 11.50941 7.180332
p.val   0.00000  0.071786 0.018249  0.00000 0.000000
         ind.av  ind.apr      cos1     sin4
t.stat 113.7514 114.7326 -1.933705 2.488495
p.val    0.0000   0.0000  0.053149 0.012828
```

Comparons maintenant les indicatrices d'avant/après quota ; leurs estimations semblent très proches. Par `str(modar2X)`, nous situons la matrice des covariances des estimateurs et l'extrayons des résultats, matrice `aa` ci-dessous, puis la sous-matrice des covariances des paramètres qui nous intéressent, en positions 5 et 6. Enfin nous calculons la variance de la différence des estimateurs `vardif`, puis la t-statistique pour tester sa nullité.

```
> aa=modar2X$var.coef
> (aa0=aa[6:7,6:7])

             ind.av     ind.apr
ind.av  0.004389649 0.004220320
ind.apr 0.004220320 0.004323493

> difa = as.matrix(c(1,-1))
> (vardif = t(difa)%*%aa0%*%difa )

             [,1]
[1,] 0.0002725028

> (t.stat=(t(difa)%*%modar2X$coef[6:7])/vardif^.5)

     [,1]
[1,]    0
```

La p-value est manifestement de l'ordre de 0.5 ; les niveaux ne sont pas significativement différents et nous pouvons supprimer les indicatrices de périodes pour les remplacer par une constante dans la régression

```
> (modar3X=Arima(log.lait,order=c(9,0,0),seasonal=list(order=c(2,0,0)),
+  xreg=xmat0c[,-(1:2)],fixed=c(NA,rep(0,6),rep(NA,7))))
```

```
...
Coefficients:
         ar1  ar2  ar3  ar4  ar5  ar6  ar7      ar8     ar9
      0.7267    0    0    0    0    0    0  -0.0986  0.1299
s.e.  0.0359    0    0    0    0    0    0   0.0528  0.0534
        sar1    sar2  intercept     cos1    sin4
      0.5845  0.3700     7.5410  -0.0950  0.0085
s.e.  0.0508  0.0512     0.0652   0.0493  0.0035

sigma^2 estimated as 0.0005543:  log likelihood = 854.86
AIC = -1691.72   AICc = -1690.37   BIC = -1632.89
```

```
> t(Box.test.2(residuals(modar3X),ret,fitdf=8,decim=4))
```

```
          [,1]    [,2]    [,3]   [,4]    [,5]
Retard  6.0000 12.0000 18.0000 24.000 30.0000
p-value 0.8455  0.2831  0.4261  0.033  0.0401
```

```
> round(t_stat(modar3X),digits=2)
```

```
         ar1   ar8  ar9 sar1 sar2 intercept  cos1 sin4
t.stat 20.22 -1.86 2.43 11.5 7.23    115.69 -1.93 2.47
p.val   0.00  0.06 0.01  0.0 0.00      0.00  0.05 0.01
```

```
> resid.x = residuals(modar3X)
```

Examinons la normalité des résidus :

```
> aa=dagoTest(resid.x)
```

La p-value pour le test omnibus de D'Agostino est 0.33, valeur très satisfaisante. Le modèle retenu est :

$$\texttt{log.lait}_t = 7.54 - 0.09\,\texttt{cos1}_t + 0.01\,\texttt{sin4}_t + u_t \tag{11.2}$$

$$u_t = \frac{1}{(1 - 0.73\,\mathrm{B} + 0.1\,\mathrm{B}^8 - 0.13\,\mathrm{B}^9)(1 - 0.58\,\mathrm{B}^{12} - 0.37\,\mathrm{B}^{24})} z_t,$$

$$z_t \sim \mathrm{BBN}(0, 0.000554).$$

En résumé, la série `log.lait` montre une forte saisonnalité. On y observe un changement de comportement à partir de l'introduction des quota. Nous avons d'abord modélisé la série de 60 observations avant quota et avons obtenu un SARMA, modèle stationnaire. Ensuite nous avons essayé de modéliser la fin de la série et avons obtenu un SARIMA, modèle non stationnaire. Nous avons alors changé de stratégie. Nous avons ajusté : (1) un SARIMA à l'ensemble de la série, sans obtenir de modèle très satisfaisant en termes de blancheur des résidus, puis (2) un ARMAX à la même série. Dans ce dernier modèle, une indicatrice distinguait les situations avant et après quota. Après quelques tâtonnements, simplifications et élimination de variables explicatives, nous avons obtenu un modèle satisfaisant pour toute la série, modèle où les quota n'ont pas d'effet quantifiable sur le niveau moyen de la collecte. La comparaison des AIC du SARIMA et de l'ARMAX nous a conduit à

retenir l'ARMAX pour modéliser la série. L'introduction des quota semble avoir eu comme effet de diminuer très progressivement le niveau de la collecte, sans que ceci puisse être capté par un de nos modèles, en diminuant le niveau du pic de collecte mais pas celui de son « étiage » (fig. 11.3).

Chapitre 12

Hétéroscédasticité conditionnelle

L'examen du chronogramme du cours de l'action Danone et celui de son rendement au chapitre 1 (fig. 1.4) nous ont indiqué que (1) le cours n'est pas stationnaire et (2) le rendement, nul en moyenne, prend sur des périodes assez longues des valeurs absolues élevées, puis sur d'autres périodes des valeurs absolues faibles. Cette observation suggère une autocorrélation de la valeur absolue, ou du carré, du rendement ; elle est confirmée au chapitre 4, exercice 1, où nous avons conclu à la blancheur du rendement mais pas à celle de son carré. On est en présence d'*hétéroscédasticité conditionnelle*.

Ce chapitre présente quelques modèles utiles pour traiter les séries financières. Nous précisons d'abord ce qu'est le rendement du cours d'une action. Nous introduisons (section 12.2) les ARCH et les GARCH, modèles susceptibles de prendre en compte l'hétéroscédasticité conditionnelle, ainsi que des tests de présence d'une telle hétéroscédasticité. Après avoir étudié le plus simple de ces modèles, l'ARCH(1), nous montrons sur des données simulées comment estimer ces modèles dans R.

Munis de ces outils, nous explorons les rendements du CAC40 de Danone, de la Société générale et de L'Oréal autour de la crise de 2007. Nous nous intéressons ensuite plus particulièrement à Danone et à L'Oréal dont nous modélisons les rendements. Une série de rendements étant le plus souvent un bruit blanc, sa prédiction est 0, mais ce bruit blanc présentant une variance conditionnelle à son passé non constante, la prévision de cette variance permet de préciser les intervalles de prévision ; nous les calculons pour ces sociétés.

12.1 Notions de base

Rendement. Soit x_t le cours d'un titre. Le rendement simple est

$$r_t^* = \frac{x_t - x_{t-1}}{x_{t-1}}, \qquad (*)$$

donc $x_t = (1+r_t^*)\,x_{t-1}$ et $\log(x_t) = \log(1+r_t^*)+\log(x_{t-1})$. Si $x_t \simeq x_{t-1}$ on obtient

$$r_t^* \simeq \Delta \log(x_t). \qquad (**)$$

On appelle $r_t = \Delta \log(x_t)$, *rendement composé*. Les deux rendements prennent des valeurs très proches. Nous travaillerons avec le rendement composé. On appelle *volatilité* d'un titre l'écart type de son rendement. Le rendement composé sur k périodes est la somme des rendements composés de chaque période. Une année contenant environ $A = 260$ jours de cotation, le rendement annualisé est alors $r_{A,t} = \log(x_t) - \log(x_{t-A})$. Si le rendement quotidien est un $\mathrm{BB}(0, \sigma^2)$, la volatilité annualisée est $\sqrt{A}\sigma$.

Vocabulaire. On dit qu'un marché est *efficient* si le prix des titres sur ce marché reflète complètement toute l'information disponible. Dans un tel marché, il est impossible de prévoir les rentabilités futures. En termes de séries temporelles, le rendement sur ce marché est donc un bruit blanc. Le bruit blanc est ainsi un modèle de référence pour le rendement d'un titre. Mais, nous le verrons, on rencontre assez couramment des rendements qui ne sont pas des bruits blancs.

Dans les modèles de séries temporelles classiques (ARIMA notamment), la variance de la série conditionnellement à son passé est constante. Or le graphe de beaucoup de séries financières, telles que l'action Danone, suggère que la variance du rendement n'est pas constante. Précisément, les séries de rendement montrent souvent des fluctuations de même ampleur, faible ou forte, pendant plusieurs dates consécutives ; ce qui indique que la variance conditionnelle au passé est parfois faible, parfois élevée. Les modèles de rendement d'action doivent donc pouvoir modéliser la variabilité de la série conditionnellement au passé.

Si le rendement est un bruit blanc mais que son carré montre une autocorrélation, c'est que le rendement est une suite de v.a. non corrélées mais *non indépendantes*. Il ne peut donc pas être un bruit blanc *gaussien*. Voyons ce qu'il en est pour le rendement de Danone.

Normalité du rendement de Danone. Les rendements sont calculés à l'aide de `returns()` de **timeSeries** puis un test de D'Agostino (cf. vignette `Anx3`) examine la normalité :

```
> require(caschrono)
> require(fBasics)
> data(csdl)
> aa=returns(csdl,percentage=TRUE)
> aab=aa[complete.cases(aa)==TRUE,]
> r.csdl=its(aab,as.POSIXct(row.names(aab)))
```

```
> aa=dagoTest(r.csdl[,"Danone"],title ="Danone",description = NULL)
> res.aa=cbind(aa@test$statistic,aa@test$p.value)
```

Les statistiques de test et les p-values sont présentées dans le tableau 12.1.

Tableau 12.1 – Action Danone -Test de normalité du rendement.

	Stat. de test	p-value
Chi2 | Omnibus	66.3926	0.0000
Z3 | Skewness	-2.4423	0.0146
Z4 | Kurtosis	7.7735	0.0000

Les alternatives sont : pour la ligne *Omnibus*, « la distribution n'est pas normale par son aplatissement ou par son asymétrie », pour la ligne *Skewness*, « la distribution n'est pas normale par son asymétrie » et pour la ligne *Kurtosis*, « la distribution n'est pas normale par son aplatissement ». Pour les trois alternatives, la p-value est très faible ; on rejette donc l'hypothèse de normalité du rendement, comme le raisonnement précédent le laissait prévoir, et ce rejet est davantage dû à un excès d'aplatissement qu'à une asymétrie ; notons que le signe de l'asymétrie indique que la distribution des rendements a une queue gauche chargée : il y a davantage de rendements très négatifs que de rendements positifs. Complétons notre compréhension de la non-normalité en examinant graphiquement la distribution du rendement. Superposons sur un même graphique la densité de probabilité du rendement estimée non paramétriquement et la densité d'une distribution normale de mêmes moyenne et variance que le rendement (fig. 12.1). Nous créons pour cela la fonction `density.plot()` :

```
> density.plot=function(x,legende=FALSE,...){
+ H<-hist(x,sub=NULL,ylab="densité",freq=FALSE, ...)
+ abline(v=0,lwd=2)
+ rug(x,ticksize=0.01)
+ xmin=par()$usr[1];xmax=par()$usr[2]
+ tab<-seq(xmin,xmax,0.1)
+ lines(tab,dnorm(tab,mean(x),sd(x)),col="red",lty=2,lwd=2)
+ lines(density(x),lwd=2,col="orange")
+ if(legende)
+ lg0=c("estimation n.p. de la densité","estimation d'une gaussienne")
+ legend("topright",legend=,lg0,lty=c(1,2),lwd=2,
+ col=c("orange","red"),cex=0.9)
+ }
> density.plot(r.csdl[,3],xlab="rendement",ylim=c(0,0.3),
+ nclass=12,xlim=c(-10,10),legende=TRUE)
```

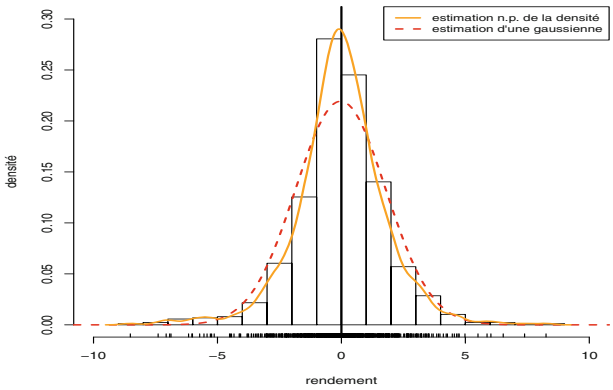

Fig. 12.1 – Densité de probabilité du rendement de l'action Danone.

Le graphe ne permet pas de détecter nettement l'asymétrie de la distribution du rendement, mais montre bien que cette distribution est plus concentrée autour de la moyenne qu'une distribution normale, ce qui vient appuyer notre lecture du tableau des p-values.
La bibliographie sur le sujet est très abondante. Citons notamment Ruppert (2004) et Alexander (2008). Le premier donne une présentation explicite, tant des méthodes statistiques que de la problématique financière, bien adaptée à un public de culture scientifique débutant en finance, alors que la seconde entre dans le détail des questions empiriques dans les séries financières, pour un public averti en finance.

12.2 Modèles d'hétéroscédasticité conditionnelle

Cette section est consacrée à une présentation sommaire des modèles ARCH et GARCH. D'abord, pour pouvoir traiter des séries de moyenne non nulle, situation de certains rendements, imaginons qu'on observe une série y_t qui se comporte comme un rendement, noté ϵ_t, à une constante près

$$y_t = c + \epsilon_t$$

où $\mathsf{E}(\epsilon_t) = 0$. Notons σ_t^2, la variance du rendement conditionnelle au passé.

$$\sigma_t^2 = \mathsf{var}(\epsilon_t | F_{t-1})$$

où F_{t-1} désigne, sans autres précisions, le passé $\epsilon_{t-1}, \epsilon_{t-2}, \cdots$ ou $y_{t-1}, y_{t-2}, \cdots$. Nos observations nous conduisent précisément à penser que σ_t^2 est une fonction de $\epsilon_{t-1}^2, \epsilon_{t-2}^2, \cdots$. Les modèles ARCH (Autoregressive Conditionaly Heteroscedastic) explicitent cette dépendance. Le plus simple, l'ARCH(1), est défini ainsi : ϵ_t suit

un modèle ARCH(1) s'il obéit à

$$\begin{aligned} \epsilon_t &= \sigma_t z_t \\ z_t \sim i.i.d.\mathcal{N}(0,1) \quad &\text{indépendant de } F_{t-1}\ \forall t \\ \sigma_t^2 &= \omega + \alpha_1 \epsilon_{t-1}^2 \end{aligned}$$

où $\omega > 0$ et $0 \leq \alpha_1 < 1$. De même qu'on considère des ARMA de moyenne non nulle, on considère des ARCH(1) de moyenne non nulle : $y_t = c + \epsilon_t$, ϵ_t défini ci-dessus et il est clair que y_t et ϵ_t ont la même dynamique [1].
La suite de la section est un peu technique et sa lecture peut être différée après l'étude de l'ARCH(1).

Définition 12.1 (ARCH(p))
y_t suit un ARCH(p) (processus autorégressif conditionnellement hétéroscédastique d'ordre p) s'il obéit à :

$$\epsilon_t = \sigma_t z_t \tag{12.1}$$

$$z_t \sim i.i.d.\mathcal{N}(0,1) \quad \textit{indépendant de } F_{t-1}\ \forall t$$

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_p \epsilon_{t-p}^2, \tag{12.2}$$

où $\omega > 0$ et $0 \leq \alpha_i,\ i = 1, \cdots, p$ et

$$\alpha_1 + \cdots + \alpha_p < 1.$$

Cette dernière condition assure la stationnarité de ϵ_t. La *variance marginale* (ou inconditionnelle) pour un tel processus est :

$$\mathsf{var}(\epsilon_t) = \frac{\omega}{1 - \sum_{i=1}^{p} \alpha_i}. \tag{12.3}$$

Mais observons que les *variances conditionnelles* $\sigma_{t-1}^2, \sigma_{t-2}^2, \cdots$, sont elles-mêmes des résumés de la variabilité passée, d'où l'idée de modèles parcimonieux où σ_t^2 est fonction de $\epsilon_{t-1}^2, \cdots$ passés et également de variances conditionnelles passées : $\sigma_{t-1}^2, \cdots$. C'est le point de vue des modèles GARCH.

Définition 12.2 (GARCH(p, q))
y_t suit un GARCH(p, q) (processus autorégressif conditionnellement hétéroscédastique généralisé d'ordres p et q) s'il obéit à :

$$\epsilon_t = \sigma_t z_t \tag{12.4}$$

$$z_t \sim i.i.d.\mathcal{N}(0,1) \quad \textit{indépendant de } F_{t-1}\ \forall t$$

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_p \epsilon_{t-p}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_q \sigma_{t-q}^2, \tag{12.5}$$

1. En réalité, si z_t est gaussien comme indiqué, nécessairement, $\alpha_1 < 1/3$, voir par exemple Tsay (2005).

où $\omega > 0$, $0 \leq \alpha_i,\ i = 1, \cdots, p,\ 0 \leq \beta_j,\ j = 1, \cdots, q$ *et*

$$\sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j < 1.$$

Cette dernière condition assure la stationnarité de ϵ_t. La *variance marginale* pour un tel processus est :

$$\mathsf{var}(\epsilon_t) = \frac{\omega}{1 - (\sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j)}. \tag{12.6}$$

Le tableau 12.2 ci-dessous, très inspiré de Ruppert (2004), permet de situer entre eux les ARMA et les GARCH pour ce qui est des moyennes et variances, marginales et conditionnelles.

Tableau 12.2 – Caractéristiques des ARMA et GARCH.

	Modèle			
Caractéristique	BB gau.	ARMA	GARCH	ARMA/GARCH
Moy. condi.	const.	non const.	0	non const.
Var. condi.	const.	const.	non const.	non const.
Dist. condi.	normale	normale	normale	normale
Moy. et var. margi.	const.	const.	const.	const.
Dist. margi.	normale	normale	non normale	non normale

Remarques

- Dans le tableau : ARMA/GARCH désigne un modèle tel que (12.12), « non normale » (constat théorique) pourrait être remplacé par « à queues chargées », constat fait sur la plupart des distributions empiriques de rendements (Danone entre autres).
- Il existe une représentation ARMA d'un GARCH (12.7) mais elle a un bruit non gaussien et il demeure plus difficile d'estimer un modèle de variance, comme un GARCH, qui modélise une donnée non observée, qu'un modèle de moyenne comme un ARMA, qui travaille directement sur des observations.
- Dans la pratique on essaie des valeurs p et q qui ne dépassent pas 1 ou 2.
- Les termes en α_i mesurent la réaction de la volatilité conditionnelle au marché, alors que les termes en β_j expriment la persistance de la réaction aux chocs du marché.
- Dans un modèle à hétéroscédasticité conditionnelle, on distingue la volatilité conditionnelle σ_t et la volatilité marginale ou de long terme $\sqrt{\mathsf{var}(\epsilon_t)}$.
- Dans un modèle GARCH il faut nécessairement $p \geq 1$.
- Une fois estimé un GARCH, en remplaçant dans (12.6) les paramètres par leurs estimations, on obtient une estimation par substitution de la variance inconditionnelle. Il peut arriver que l'estimation ainsi obtenue soit négative, situation qui indique que le modèle n'est pas satisfaisant.

– Le bruit blanc gaussien peut être remplacé par un bruit blanc non gaussien, par exemple, z_t BB distribué suivant une loi de Student centrée, dont il faut alors estimer le paramètre « degrés de liberté ».

Représentations ARMA associées à un ARCH et à un GARCH

Nous admettrons que si ϵ_t suit un ARCH(p) (12.1, 12.2), alors, ϵ_t^2 suit un AR(p) :

$$\epsilon_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_p \epsilon_{t-p}^2 + u_t \qquad u_t \sim \text{BB}\,. \tag{12.7}$$

Ce résultat suggère un test d'hétéroscédasticité conditionnelle que nous examinerons dans la prochaine section. Nous admettrons de même que si ϵ_t suit un GARCH(p, q), alors, ϵ_t^2 suit un ARMA($max(p, q), q$).

12.3 ARCH(1) et test d'hétéroscédasticité conditionnelle

12.3.1 Simulation d'un ARCH(1)

La simulation des ARCH/GARCH se fait à l'aide de `garchSim()` de **fGarch**. Ce package fait partie de `Rmetrics`, cf. Wuertz & Rmetrics Foundation (2010), environnement parallèle à `S+FinMetrics` dont Zivot & Wang (2006) constitue un guide.

Exemple 12.1 Simulons une trajectoire d'un ARCH(1) suivant :

$$\begin{aligned} y_t &= 5 + \epsilon_t \\ \epsilon_t = \sigma_t z_t,\ z_t &\sim \text{BBN}(0, \sigma_z^2) \\ \sigma_t^2 &= 0.1 + 0.9\, \epsilon_{t-1}^2. \end{aligned} \tag{12.8}$$

Le modèle à simuler est défini par `garchSpec()` ; il doit être donné sous forme de liste, les noms des composantes de la liste indiquant le type de modèle. Par `extended=TRUE` on conserve en plus de la série simulée, z_t et σ_t.

```
> require(fGarch)
> spec.1=garchSpec(model=list(mu=5,omega=0.1,alpha=0.9,beta=0),rseed=397)
> archsim.1=garchSim(extended=TRUE,spec.1,n = 300,n.start=10)
> head(archsim.1,2)
```

```
GMT
              garch     sigma        eps
2010-04-06 5.172777 0.3175684  0.5440636
2010-04-07 4.668813 0.3561837 -0.9298203
```

Etant donné que dans la définition du modèle z_t est un bruit blanc gaussien, il est pertinent après l'estimation d'un ARCH ou d'un GARCH de vérifier si le résidu $\widehat{z}_t$

en est aussi un. Examinons la série simulée `archsim.1[,1]` et l'écart type conditionnel `archsim.1[,2]` (fig. 12.2 haut) ainsi que l'ACF de la série `archsim.1[,1]` et l'ACF de son carré centré (`archsim.1[,1]`-5)^2 (fig. 12.2 bas) (SiteST). On note que le carré centré de la série n'est pas un bruit blanc.

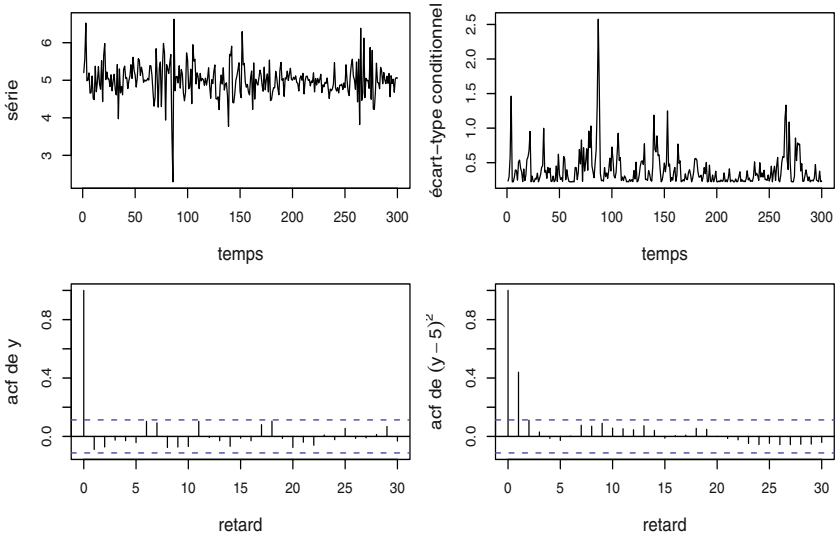

Fig. 12.2 – Simulation d'un ARCH(1).

Exercice 12.1
- Simuler 300 observations de y_t obéissant à (12.8). (On simulera 10 valeurs avant de commencer à stocker la série et on initialisera à 0 la première variance conditionnelle.) On reproduira explicitement les équations. Visualiser la série et recommencer les simulations avec différentes graines.
- Tester la blancheur du carré centré de la série. Commenter.

12.3.2 Moments d'un ARCH(1)

Examinons quelques éléments théoriques sur le modèle le plus simple. Leur compréhension peut aider à se débarrasser de quelques idées fausses qu'on adopte assez facilement à propos des modèles d'hétéroscédasticité conditionnelle.
Considérons ϵ_t obéissant à un ARCH(1) :

$$\epsilon_t = \sigma_t z_t \tag{12.9}$$

$$z_t \sim i.i.d.\mathcal{N}(0,1) \quad \text{indépendant de } F_{t-1}$$

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 \tag{12.10}$$

où $\omega > 0$ et $0 \leq \alpha_1 < 1$.

Calculons la moyenne conditionnelle :

$$\mathsf{E}(\epsilon_t|F_{t-1}) = \mathsf{E}(\sigma_t z_t|F_{t-1}) = \sigma_t \mathsf{E}(z_t|F_{t-1}) = 0$$

et la moyenne marginale (ou inconditionnelle) de ϵ_t est donc nulle également. La variance conditionnelle est $\mathsf{var}(\epsilon_t|F_{t-1}) = \mathsf{E}(\epsilon_t^2|F_{t-1}) = \sigma_t^2$. Calculons maintenant la variance marginale de ϵ_t à partir de la décomposition classique de la variance :

$$\begin{aligned} \mathsf{var}(\epsilon_t) &= \texttt{moy. des var. condi.} + \texttt{var. des moy. condi.} \\ &= \mathsf{E}(\sigma_t^2) + 0. \end{aligned}$$

Mais $\mathsf{E}(\sigma_t^2) = \omega + \alpha \mathsf{E}(\epsilon_{t-1}^2) = \omega + \alpha \mathsf{var}(\epsilon_{t-1})$ et comme ϵ_t est stationnaire :

$$\mathsf{var}(\epsilon_t) = \frac{\omega}{1-\alpha_1}.$$

La condition $\alpha_1 < 1$ est bien nécessaire. Examinons l'autocorrélation d'ordre 1 de ϵ_t. $\mathsf{cov}(\epsilon_t, \epsilon_{t_1}) = \mathsf{E}(\epsilon_t \epsilon_{t-1})$ car ϵ_t est de moyenne nulle.

$$\mathsf{E}(\epsilon_t \epsilon_{t-1}) = \mathsf{E}(\mathsf{E}(\epsilon_t \epsilon_{t-1}|F_{t-1})) = \mathsf{E}(\epsilon_{t-1}\mathsf{E}(\epsilon_t|F_{t-1})) = 0$$

car $\mathsf{E}(\epsilon_t|F_{t-1}) = 0$. De même $\mathsf{cov}(\epsilon_t, \epsilon_{t-k}) = 0 \ \forall k \neq 0$. Ainsi ϵ_t est bien un BB mais à variance conditionnelle non constante. ϵ_t est donc un BB non gaussien dans la définition duquel entre z_t, bruit blanc gaussien (voir le tableau 12.2).
Explicitons maintenant l'aspect autorégressif de l'ARCH(1). Posons $u_t = \epsilon_t^2 - \mathsf{E}(\epsilon_t^2|F_{t-1})$. On peut montrer que u_t est un BB de moyenne nulle. De (12.1), on a

$$\epsilon_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + u_t,$$

donc, si ϵ_t suit un ARCH(1), ϵ_t^2 suit un AR(1).

12.3.3 Tests d'hétéroscédasticité conditionnelle

Test basé sur le multiplicateur de Lagrange. Dans ce cadre, on teste la significativité de la régression (12.7), effectuée sur le carré de la série centrée dont on veut tester l'hétéroscédasticité ; l'hypothèse nulle est : la régression n'est pas significative, c'est-à-dire qu'il n'y a pas d'hétéroscédasticité conditionnelle. Comme l'ordre p est inconnu, on régresse le carré de la série centrée sur une constante et un certain nombre de valeurs passées. S'il n'y a pas d'hétéroscédasticité conditionnelle, les coefficients de la régression ne sont pas significatifs [2]. Si l'on régresse sur k valeurs passées, la statistique de test suit approximativement, en l'absence d'hétéroscédasticité conditionnelle, une loi $\chi^2(k)$. Comme tout test basé sur la distance du χ^2, on rejette l'hypothèse nulle pour une distance élevée et la p-value donne la probabilité de dépasser la distance observée si l'hypothèse nulle est vérifiée.

2. Le test du multiplicateur de Lagrange calcule la distance du χ^2, du vecteur des valeurs estimées des coefficients, sauf la constante, au vecteur nul.

Illustrons ce test sur la série simulée à l'aide de la fonction `ArchTest()` de **FinTS**. On teste la non-significativité de la régression du carré sur son passé. On choisit de régresser le carré de la série sur $k = 12$ retards ; on teste également l'hétéroscédasticité conditionnelle du bruit blanc qui a servi à la simulation de l'ARCH :

```
> require(FinTS)
> ArchTest(archsim.1[,1],lag=12)

        ARCH LM-test; Null hypothesis: no ARCH effects

data:  archsim.1[, 1]
Chi-squared = 21.0413, df = 12, p-value = 0.04978

> ArchTest(archsim.1[,3],lag=12)

        ARCH LM-test; Null hypothesis: no ARCH effects

data:  archsim.1[, 3]
Chi-squared = 6.7548, df = 12, p-value = 0.8734
```

Sous l'hypothèse nulle d'homoscédasticité, la statistique doit être distribuée approximativement suivant un $\chi^2(12)$. Evidemment, pour la série simulée suivant un ARCH(1), on rejette l'hypothèse d'homoscédasticité et on l'accepte pour le bruit blanc gaussien `archsim.1[,3]` qui a servi à la simuler.

Exercice 12.2 (Test d'hétéroscédasticité conditionnelle)
Tester l'hétéroscédasticité conditionnelle de ces deux séries par un test de blancheur de Box-Pierce.

12.4 Estimation et diagnostic d'ajustement d'un GARCH

L'estimation d'un ARCH ou d'un GARCH se fait par la méthode du maximum de vraisemblance. Cette méthode nécessite de préciser la loi du bruit blanc z_t. Nous avons supposé $z_t \sim \text{BBN}(0, 1)$. Nous illustrons la méthode en estimant le modèle d'une série simulée.

Exemple 12.1 (Simulation et estimation d'un GARCH(2,1)) D'abord, nous simulons 420 observations suivant le modèle

$$\begin{aligned} y_t &= 2 + \sigma_t z_t \\ \sigma_t^2 &= 0.09 + 0.15\epsilon_{t-1}^2 + 0.3\epsilon_{t-2}^2 + 0.4\sigma_{t-1}^2 \end{aligned} \tag{12.11}$$

à l'aide de `garchSim()` et abandonnons les 20 premières valeurs :

```
> spec=garchSpec(model=list(mu=2,omega=.09,alph=c(.15,.3),beta=.4),rseed=9647)
> var.margi=0.09/(1-0.15-0.3-0.4)
> y=garchSim(spec,n=420,extended=TRUE)
> y1=y[21:420,1]
```

La série à étudier est formée des 400 dernières observations contenues dans la première colonne de `y`. Avant d'ajuster un modèle à cette série, examinons son chronogramme (fig. 12.3).

```
> plot.ts(y1,xlab='temps')
```

Il y a plusieurs valeurs extrêmes. Ne pas en tenir compte risquerait de fausser considérablement l'estimation, aussi décidons-nous de les remplacer par des valeurs plus raisonnables. Il faut commencer par repérer ces valeurs extrêmes et les situer par rapport à des quantiles extrêmes de la distribution des écarts à la moyenne, en valeur absolue. Nous retenons 5 ordres quantiles élevés et calculons les quantiles correspondants, `q5`.

```
> m1=mean(y1)
> (q5=quantile(abs(y1-m1),probs=c(.975,.98,.985,.99,.995)))

   97.5%      98%    98.5%      99%    99.5%
2.965956 3.095066 3.696268 5.500659 6.443857

> extrem985=which(abs(y1-m1)>q5[3])
> cat('nombre de points : ',length(extrem985),'\n')

nombre de points :  6

> y1b=y1
> y1b[extrem985]=q5[3]*sign(y1[extrem985]-m1)
```

Après examen de ces quantiles, nous décidons de ramener les points de valeur absolue supérieure au quantile d'ordre 98.5% à ce percentile, en conservant le signe de la valeur modifiée. Pour ce faire, nous repérons à l'aide de `which()` les indices des points dépassant ce seuil en valeur absolue, puis nous affectons les nouvelles valeurs à ces points. L'effet de cette modification de valeur s'observe en comparant les chronogrammes (fig. 12.3 et 12.4).

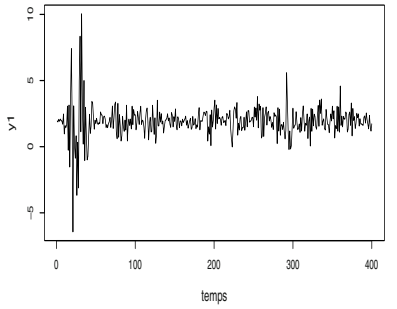

Fig. 12.3 – Série originale.

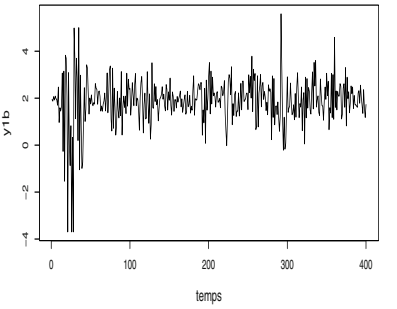

Fig. 12.4 – Série corrigée.

Nous pouvons maintenant estimer le modèle à l'aide de `garchFit()` de **fGarch**.

```
> mod1=garchFit(~garch(2,1),data=y1b,trace=FALSE,include.mean=TRUE)
> summary(mod1)
```

```
...
Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu       1.94822     0.02889   67.427  < 2e-16 ***
omega    0.08188     0.03075    2.663 0.007752 **
alpha1   0.18705     0.08573    2.182 0.029134 *
alpha2   0.37238     0.11902    3.129 0.001756 **
beta1    0.40945     0.11995    3.413 0.000642 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 -464.2879    normalized:  -1.160720

Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  3.133772  0.2086941
 Shapiro-Wilk Test  R    W      0.9953712 0.2820558
 Ljung-Box Test     R    Q(10)  10.23328  0.4202707
 Ljung-Box Test     R    Q(15)  17.5312   0.2881126
 Ljung-Box Test     R    Q(20)  25.39945  0.186574
 Ljung-Box Test     R^2  Q(10)  7.318916  0.6950332
 Ljung-Box Test     R^2  Q(15)  15.10882  0.4436072
 Ljung-Box Test     R^2  Q(20)  18.87781  0.5297806
 LM Arch Test       R    TR^2   7.555168  0.8188556

Information Criterion Statistics:
     AIC      BIC      SIC     HQIC
2.346439 2.396333 2.346132 2.366198
```

Commençons par examiner la fin du résumé de l'estimation.
A partir de `Standardised Residuals Tests:`, on trouve un ensemble de tests portant sur les résidus $\widehat{z}_t$ (`R`) ou sur leur carré (`R^2`) (cf. 12.4) :

- deux tests de normalité des résidus ;
- un test de blancheur des résidus jusqu'aux retards 10, 15 et 20 ;
- des tests d'homoscédasticité conditionnelle, tests de blancheur du carré des résidus et test du multiplicateur de Lagrange.

Les tests de normalité, de blancheur et d'homoscédasticité montrent des p-values élevées. Ce qui est logique, vu qu'on a ajusté le modèle même qui a servi à simuler la série. Les paramètres sont significatifs. Pour entrer dans l'examen détaillé de l'estimation, il nous faut examiner la structure de `mod1` (`str(mod1)`). C'est un objet avec 11 slots (marqués par des `@`), il est donc de classe S4. Certains slots sont des vecteurs et d'autres des listes. Nous voyons que :

– les estimations des paramètres se trouvent dans la composante `par` du slot `@fit` ;
– les écarts types d'estimation sont stockés dans `mod1@fit$se.coef` et la matrice des covariances des estimateurs des paramètres dans `mod1@fit$cvar` ;
– les résidus, valeurs ajustées, variances et écart types conditionnels se trouvent dans les slots `@residuals`, `@fitted`, `@h.t`, `@sigma.t`.

Estimons la variance marginale par substitution (voir l'expression 12.6) :

```
> var.marg.est<-function(mod){
+ param.estim=mod@fit$par
+  std.estim=mod@fit$se.coef
+  k<-which(names(param.estim)=="omega")
+  value=param.estim[k]/(1-sum(param.estim[(k+1):length(param.estim)]))
+ cat("variance marginale : ",value,"\n")
+ }
> var.marg.est(mod1)

variance marginale :  2.631328
```

L'estimation est positive, ce qu'on préfère obtenir. Le contraire, signe que le modèle ajusté ne convient pas, serait étonnant après avoir ajusté sur des données simulées le modèle ayant servi à la simulation elle-même.
Examinons graphiquement la qualité de l'ajustement. L'aide de `garchFit()` nous indique que `plot()`, appliqué à un objet en sortie de `garchFit()`, offre 13 graphiques ; `which=` permet d'en choisir un. Le graphique 3 donne le chronogramme de la série, auquel sont superposés ± deux écarts types conditionnels estimés. Nous recalculons cependant ce graphique pour pouvoir choisir les couleurs.

```
> mu=coef(mod1)["mu"]; sigma.t=mod1@sigma.t
> ymat=cbind(y1b,mu-2*sigma.t,mu+2*sigma.t)
> matplot(1:nrow(ymat),ymat,type='l',col="black",ylab="",xlab="temps",
+             lwd=1.5,lty=c(1,2,2))
```

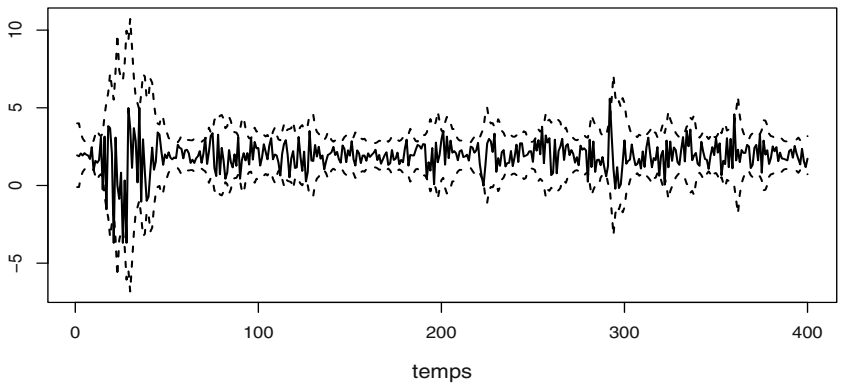

Fig. 12.5 – Trajectoire estimée et bande de deux écarts types conditionnels.

Le seul examen du graphique ne donnant pas une idée précise de la proportion de points dans la bande de confiance, nous la calculons :

```
> prop.obs=mean((y1b >mu-2*sigma.t)&(y1b < mu+2*sigma.t))
> prop.att=1-2*(1-pnorm(2))
```

Les pourcentages observé et attendu valent respectivement 94% et 95%. Le modèle ajusté est (heureusement) satisfaisant. Il est très révélateur d'ajuster sur des données simulées un modèle autre que celui qui a servi à la simulation, par exemple ajuster un ARCH(1) à `y1b[,1]` (SiteST). Des p-values faibles pour les `Standardised Residuals Tests` doivent nous alerter sur la mauvaise qualité de l'ajustement.
Bien que le modèle ajusté soit satisfaisant, nous constatons que certaines estimations sont assez éloignées des paramètres utilisés pour la simulation. On peut examiner cette question en traitant le même exemple avec une autre graine.

Exercice 12.3 (Tests sur `mod1`)
L'estimation de (12.11) a été obtenue dans `mod1`. Faisons l'hypothèse que les estimateurs des paramètres sont approximativement normalement distribués.
- Extraire de la sortie de l'estimation la matrice des variances-covariances estimées des paramètres.
- Tester l'hypothèse que $\alpha_2 = 0.3$.
- Tester l'hypothèse que $(\alpha_2, \beta_1) = (0.3, 0.4)$.
- Conclure.

12.5 Prévision

Examinons le mécanisme de la prévision d'un modèle d'hétéroscédasticité conditionnelle au moyen de l'exemple de la série simulée (12.11). C'est un bruit blanc translaté de $\mu = 2$; sa prévision est donc `mu`, mais en prédisant sa variance conditionnelle on peut donner une bande de prévision autour de μ.

```
> p.cond=predict(mod1,mse="cond",n.ahead=20,plot=FALSE)
> p.cond[c(1,2,19,20),]
```

```
   meanForecast meanError standardDeviation
1      1.948216 0.6822926         0.6822926
2      1.948216 0.6144984         0.6144984
19     1.948216 1.0618770         1.0618770
20     1.948216 1.0777145         1.0777145
```

La prévision `meanForecast` de la moyenne est évidemment constante. Il n'y a pas de modèle sur la moyenne, donc la prévision `meanError` de l'écart type de l'erreur de prévision de la moyenne se confond avec la prédiction de l'écart type conditionnel : `standardDeviation`. L'option `plot=TRUE` dans `predict` fournirait le chronogramme de la série et la bande de prédiction (à 95% par défaut) autour de la prédiction de la moyenne. Si nous voulons superposer la bande de prédiction et les 20 valeurs simulées non utilisées pour l'estimation, `y[401:420,1]`, nous devons calculer les limites de cette bande :

```
> b.inf=p.cond$meanForecast-1.96*p.cond$standardDeviation
> b.sup=p.cond$meanForecast+1.96*p.cond$standardDeviation
> matpr=cbind(y[401:420,1],b.inf,b.sup)
> matplot(1:20,matpr,type='l',lty=c(1,2,2))
```

On peut également simuler, par exemple, 10 000 trajectoires et comparer la bande de confiance empirique qu'on en déduit à la bande calculée (SiteST).

12.6 Modèles à erreur conditionnellement hétéroscédastique

Nous avons présenté les modèles ARCH et GARCH comme modèles de bruit blanc conditionnellement hétéroscédastique. Un tel bruit blanc peut lui-même être l'erreur dans une série x_t obéissant par exemple à un ARMA :

$$x_t = c + \sum_{i=1}^{m} a_i x_{t-i} + \sum_{j=1}^{n} b_j \epsilon_{t-j} + \epsilon_t \tag{12.12}$$

où ϵ_t suit un GARCH(p, q) décrit par (12.4, 12.5). On appelle ARMA/GARCH ce type de modèles (voir le tableau 12.2).

12.7 Etude de séries du marché parisien autour de la crise de 2007-2008

Nous étudions à présent l'impact de la crise commencée mi-2007 sur un indice boursier et sur des titres de volatilités variées, de janvier 2006 à mars 2009. Un graphique permet d'observer l'évolution de la Société Générale, Danone et L'Oréal, et du CAC40 [3]. Nous commençons par repérer la date où le CAC40 commence à chuter : elle nous servira à dater le début de la crise.

12.7.1 Exploration

Représentons simultanément les quatre séries. Les cours des actions ont des ordres de grandeur semblables entre eux et très différents du CAC40. Nous choisissons donc deux échelles, à gauche pour le CAC40, à droite pour les actions. Il est commode de convertir les séries en type `zoo` pour cette représentation. L'objet `datelor` contient la date POSIX du jour où le cours de L'Oréal est maximum et `datemax` la date où le CAC40 est maximum.

3. La Société Générale est un établissement financier touché directement par la crise, alors que les deux autres sociétés, valeurs « défensives », sont moins directement impactées (l'impact se fera peut-être sur un plus long terme).

```
> max.csdl=apply(csdl,2,which.max)
> datelor=csdl@dates[max.csdl["L_Oreal"]]
> datemax=csdl@dates[max.csdl["Cac40"]]
> zz<-zoo(as.data.frame(csdl),csdl@dates)
> plot(zz[,"Cac40"],type="l",xlab="année",ylab="indice Cac 40")
> plot(zz[,2:4],type="l",ann=FALSE,yaxt="n",lty=2,
+ col=rainbow(3),plot.type="single")
> abline(v=c(datemax,datelor))
> axis(side=4)
> mtext(side=4,"indices Société générale, Danone, L'Oréal",line=2.5)
> legend(x="topright",bty="n",lty=c(1,2,2,2),col=c("black",rainbow(3)),
+ legend=paste(colnames(zz),
+     c("(échelle de gauche)",rep("(échelle de droite)",3))))
```

On observe (fig. 12.6) que L'Oréal présente une évolution parallèle au CAC40, avec un impact de la crise retardé à décembre 2007, alors que le cours de la Société Générale chute plus fortement que celui du CAC40, et ceci dès juin 2007.

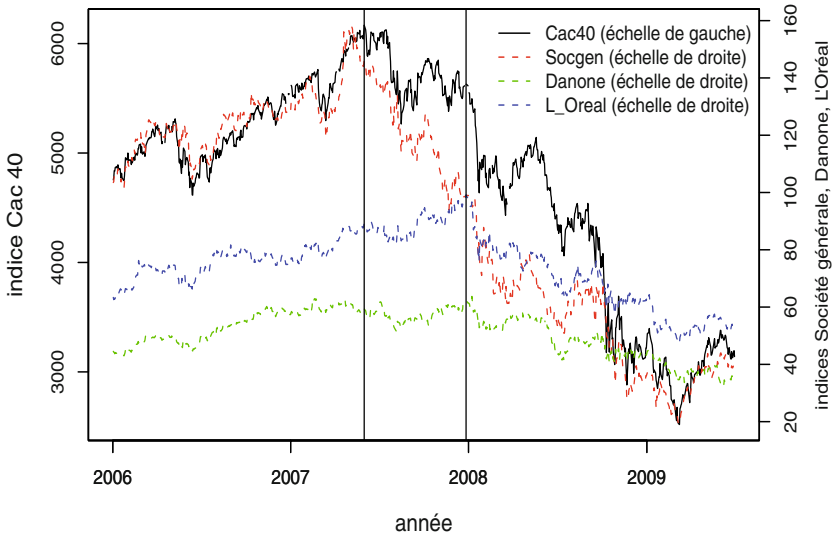

Fig. 12.6 – CAC40 et 3 sociétés - de janvier 2006 à mars 2009.

Nous nous intéressons maintenant aux rendements de ces trois sociétés avant et pendant la crise.

12.7.2 Etude des rendements

Nous calculons les rendements avant (c'est-à-dire avant juillet 2007) et pendant la crise (c'est-à-dire après juillet 2007 pour la Société Générale et Danone, et après décembre 2007 pour L'Oréal), ainsi que les aplatissements et coefficients d'asymétrie des séries correspondantes.

```
> rendav.06=rangeIts(r.csdl,end= "2007-06-01")
> rendapr.06=rangeIts(r.csdl,start="2007-06-02")
> sk.av06=apply(rendav.06,2,skewness,na.rm=TRUE)
> kurt.av06=apply(rendav.06,2,kurtosis,na.rm=TRUE)
> sk.apr06=apply(rendapr.06,2,skewness,na.rm=TRUE)
> kurt.apr06=apply(rendapr.06,2,kurtosis,na.rm=TRUE)
> sk06=rbind(sk.av06,sk.apr06,kurt.av06,kurt.apr06)[,2:3]
> colnames(sk06)=c("Socgen","Danone")
> rownames(sk06)=c("asym.av","asym.apr","aplat.av","aplat.apr")
```

Les séries de rendements contenant au moins la première valeur comme manquant, nous devons utiliser l'option `na.rm=TRUE` dans `skewness()` et `kurtosis()`. Nous calculons sur le même principe le rendement de L'Oréal avant et après le 27 décembre (SiteST). Ces mesures sont rassemblées dans le tableau 12.3.

Tableau 12.3 – Asymétrie et aplatissement des rendements.

	Socgen	Danone	L'Oréal
asym.av	0.26	0.44	0.14
asym.apr	- 0.08	- 0.18	0.47
aplat.av	2.82	1.41	0.57
aplat.apr	2.47	1.53	4.43

L'asymétrie reste positive pour L'Oréal, avant ou pendant la crise, ce qui veut dire que L'Oréal a plus de rendements positifs que négatifs même dans la crise, alors que Danone et la Société Générale ont plus de rendements négatifs que positifs quand on entre dans la crise.

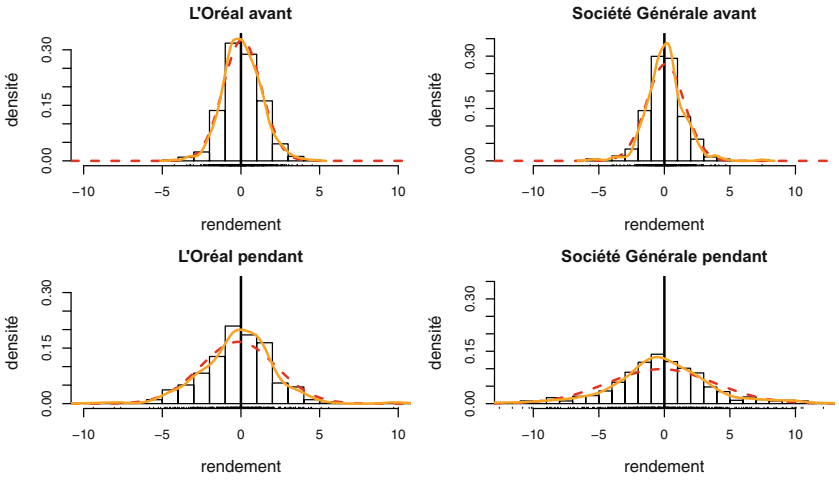

Fig. 12.7 – Rendements avant et pendant la crise.

12.7.3 Hétéroscédasticité conditionnelle des rendements

Nous étudions maintenant l'hétéroscédasticité conditionnelle des rendements à l'aide du test du multiplicateur de Lagrange. Nous nous intéressons aux rendements : (1) du CAC40 et des trois sociétés avant juin 2007, (2) du CAC40, de la Société Générale et de Danone après cette date, (3) de L'Oréal après décembre 2007. Pour les deux premiers cas, on doit appliquer `ArchTest()` à chaque colonne d'une matrice. On utilise `apply()`. Ainsi par :

```
aa.av=apply(rendav.06,2,ArchTest,lag=20)
```

on stocke dans `aa.av` le résultat de l'application de `ArchTest()` à chaque colonne de `rendav.06` et `str(aa.av)` nous montre comment les résultats sont stockés. L'ensemble des p-values de ces tests figurent dans le tableau 12.4.

Tableau 12.4 – Test d'hétéroscédasticité conditionnelle avant et pendant la crise : p-value.

	CAC40	Socgen	Danone	L'Oréal
Avant	0.0000	0.0698	0.1218	0.5281
Pendant	0.0000	0.0000	0.0000	0.0020

Le CAC40 a une forte hétéroscédasticité avant comme pendant la crise, alors que la Société Générale, Danone et L'Oréal, qui n'en montraient que peu ou pas du tout avant la crise, en montrent beaucoup par la suite.

12.8 Etude du rendement de L'Oréal

Nous modélisons maintenant le rendement de L'Oréal après décembre 2007 en vue de faire sa prédiction. Précisément, nous utiliserons comme trajectoire d'apprentissage la série du rendement moins les 50 dernières valeurs. Nous prédirons la moyenne de la série (si elle se révèle non nulle) et l'écart type conditionnel pour ces 50 dernières dates. Isolons les trajectoires d'apprentissage, `r.lor.0`, et de test, `r.lor.1` :

```
> r.lor=rangeIts(r.csdl[,"L_Oreal"],start="2007-12-28")
> r.lor.0=r.lor[1:(length(r.lor)-51)]
> r.lor.1=r.lor[(length(r.lor)-50):length(r.lor)]
```

12.8.1 Estimation

Modélisation du rendement. Examinons l'ACF et la PACF de la série d'apprentissage :

```
> xy.acfb(r.lor.0,numer=FALSE)
```

Nous remarquons (fig. 12.8) que le rendement n'est pas un bruit blanc. Nous commençons donc par en chercher un modèle. L'ACF suggère un MA(4). Après

une estimation sans restriction de ce modèle, nous constatons que la moyenne et le terme de retard 3 ne sont pas significatifs. D'où le code pour estimer un modèle avec la moyenne et le 3e paramètre nuls :

```
> mod.r.lor=Arima(r.lor.0@.Data,order=c(0,0,4),include.mean=FALSE,
+                 fixed=c(NA,NA,0,NA))
> summary(mod.r.lor)

...
Coefficients:
          ma1      ma2  ma3     ma4
      -0.2043  -0.1488    0  0.2619
s.e.   0.0537   0.0537    0  0.0533

sigma^2 estimated as 5.537:  log likelihood = -741.73
AIC = 1491.46   AICc = 1491.65   BIC = 1510.4

In-sample error measures:
         ME        RMSE         MAE         MPE        MAPE
 -0.2159882   2.3531066   1.7458763  64.9929890 183.4554058
       MASE
  0.6171533

> rr.lor=residuals(mod.r.lor)
> ret=c(6,12,18,24)
> t(Box.test.2(rr.lor,ret,type="Box-Pierce",fitdf=3,decim=4))

          [,1]    [,2]    [,3]    [,4]
Retard  6.0000 12.0000 18.0000 24.0000
p-value 0.8235  0.5527  0.6112  0.2233

> t_stat(mod.r.lor, decim=4)

           ma1     ma2    ma4
t.stat -3.8065 -2.7689 4.9146
p.val   0.0001  0.0056 0.0000
```

Le test de Box-Pierce donne des résultats satisfaisants. Testons maintenant l'absence d'hétéroscédasticité conditionnelle dans les résidus de la modélisation.

```
> ArchTest(rr.lor, lag=20)

        ARCH LM-test; Null hypothesis: no ARCH effects

data:  rr.lor
Chi-squared = 32.6343, df = 20, p-value = 0.03699
```

Il reste de l'hétéroscédasticité conditionnelle, mais de façon modérée, dans les résidus de l'ajustement par un MA(4). Avant de prendre en compte cette hétéroscédasticité, examinons la forme de la distribution des résidus où le plot de la densité (fig. 12.9) montre sa déformation par rapport à une densité normale.

```
> density.plot(rr.lor,main="",xlab ="",ylim=c(0,.2),nclass=12,xlim=c(-12,12))
```

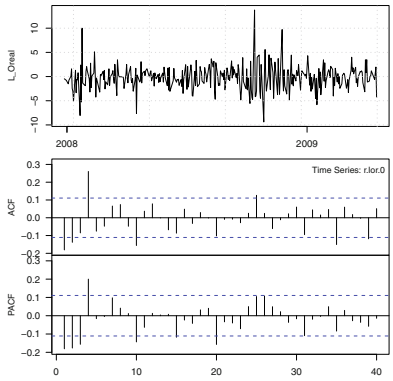

Fig. 12.8 – L'Oréal, ACF et PACF du rendement.

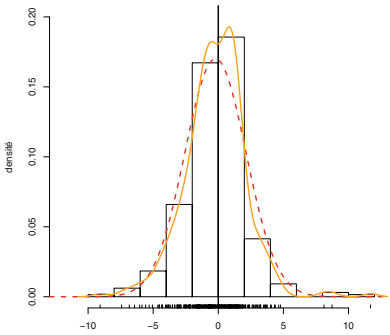

Fig. 12.9 – L'Oréal, densité de probabilité des résidus du modèle MA(4).

Modélisation de l'hétéroscédasticité du rendement. Etudions l'hétéroscédasticité du résidu de la modélisation MA(4) ci-dessus. Après essai d'un ARCH(1) qui donne une estimation négative de la variance inconditionnelle, nous ajustons un GARCH(1, 1).

```
> moda=garchFit(~garch(1,1),data=rr.lor,trace=FALSE,
+    include.mean=TRUE,na.action=na.pass)
> summary(moda)

...
Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu      -0.12585     0.11923   -1.055   0.2912
omega    0.52788     0.28716    1.838   0.0660 .
alpha1   0.14760     0.06644    2.222   0.0263 *
beta1    0.76797     0.08645    8.884   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 -731.213    normalized:  -2.242985

Standardised Residuals Tests:
                               Statistic p-Value
 Jarque-Bera Test   R    Chi^2  23.88071  6.521826e-06
 Shapiro-Wilk Test  R    W      0.9832543 0.0007585491
```

```
 Ljung-Box Test     R    Q(10)  7.897065  0.6388909
 Ljung-Box Test     R    Q(15)  13.07487  0.5965148
 Ljung-Box Test     R    Q(20)  24.05773  0.2398793
 Ljung-Box Test     R^2  Q(10)  13.04006  0.2214405
 Ljung-Box Test     R^2  Q(15)  16.89202  0.3253602
 Ljung-Box Test     R^2  Q(20)  19.55866  0.4858238
 LM Arch Test       R    TR^2   13.71432  0.3193222

Information Criterion Statistics:
     AIC      BIC      SIC     HQIC
4.510509 4.556974 4.510213 4.529052
```

```
> var.marg.est(moda)

variance marginale :  6.252412
```

L'estimation de la variance marginale obtenue par substitution dans (12.6) est positive. La non-normalité, visible sur les p-values des tests de Jarque-Bera Test et Shapiro-Wilk, demeure évidemment et doit nous rendre prudents sur les estimations `Std. Error` fournies. En revanche, le modèle a bien capté l'hétéroscédasticité.

Modélisation simultanée du rendement et de son hétéroscédasticité. Nous continuons l'estimation en combinant les deux modèles, le MA(4) pour l'évolution du rendement et le GARCH(1, 1) pour l'évolution de la variance conditionnelle de l'erreur du modèle MA(4). Au contraire de `Arima()`, `garchFit()` ne permet pas de contraindre certains paramètres ; dans la partie MA, le terme d'ordre 3 est donc libre.

```
> mod.lor=garchFit(formula=~arma(0,4)+garch(1,1),data=r.lor.0@.Data,
+                  trace=FALSE,include.mean=FALSE)
> summary(mod.lor)

...
Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
ma1     -0.21020     0.06297   -3.338 0.000844 ***
ma2     -0.07231     0.06293   -1.149 0.250531
ma3     -0.07947     0.05815   -1.367 0.171755
ma4      0.19608     0.06479    3.027 0.002474 **
omega    0.51245     0.24703    2.074 0.038041 *
alpha1   0.18341     0.07116    2.577 0.009953 **
beta1    0.74208     0.07668    9.677  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 -729.2392    normalized:  -2.23693
```

```
Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  22.7261   1.161690e-05
 Shapiro-Wilk Test  R    W      0.9842646 0.001244203
 Ljung-Box Test     R    Q(10)  6.704402  0.7530253
 Ljung-Box Test     R    Q(15)  13.44294  0.5681263
 Ljung-Box Test     R    Q(20)  22.69119  0.3042109
 Ljung-Box Test     R^2  Q(10)  12.54697  0.2501153
 Ljung-Box Test     R^2  Q(15)  16.20667  0.3684509
 Ljung-Box Test     R^2  Q(20)  19.91751  0.4631005
 LM Arch Test       R    TR^2   13.47358  0.3355782

Information Criterion Statistics:
     AIC      BIC      SIC     HQIC
4.516805 4.598119 4.515909 4.549254
```

Les estimations sont proches de celles obtenues dans les deux précédentes modélisations, d'une part de la moyenne et d'autre part de la variance. Puis nous estimons la variance marginale :

```
> var.marg.est(mod.lor)

variance marginale :  6.878408
```

L'estimation est positive. Finalement le modèle ARMA/GARCH ajusté au rendement de L'Oréal est

$$y_t = \epsilon_t - 0.2102\,\epsilon_{t-1} - 0.0723\,\epsilon_{t-2} - 0.0795\,\epsilon_{t-3} + 0.1961\,\epsilon_{t-4} \tag{12.13}$$

$$\epsilon_t = \sigma_t z_t \tag{12.14}$$

$$\sigma_t^2 = 0.5125 + 0.1834\,\epsilon_{t-1} + 0.7421\,\sigma_{t-1}^2, \tag{12.15}$$

où les paramètres MA 2 et MA 3 ne sont pas significatifs. Représentons maintenant le rendement observé sur les 100 dernières dates de la trajectoire d'apprentissage et un intervalle à 80% basé sur l'estimation de l'écart type conditionnel donné par `mod.lor@sigma.t` (fig. 12.10).

```
> n1=length(r.lor.0)-99;n2=length(r.lor.0);n1.n2=n1:n2;
> mat.est=cbind(r.lor.0[n1.n2],qnorm(.9)*mod.lor@sigma.t[n1.n2],
+               -qnorm(.9)*mod.lor@sigma.t[n1.n2])
> matplot(n1:n2,mat.est,type='l',col='black',lty =c(1,2,2),
+   xlab="100 dernières observations",ylab="rendement",xaxt="n")
> axis(1,at=axTicks(1),labels=r.lor.0@dates[axTicks(1)])
```

La proportion de points dans l'intervalle est de 80.98%, soit la proportion attendue.

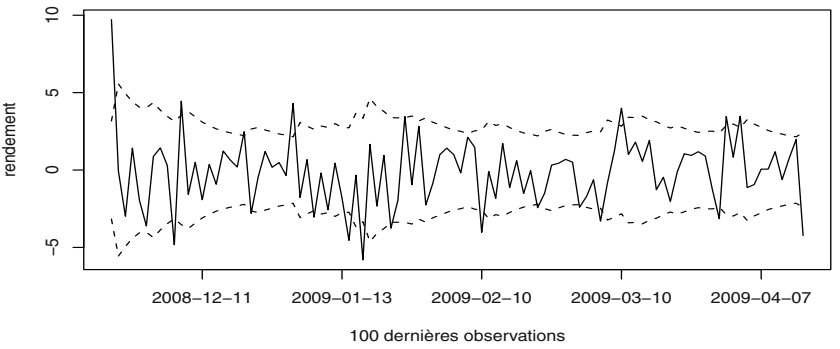

Fig. 12.10 – L'Oréal : rendement observé et intervalle à 80%.

12.8.2 Prédiction du rendement

Nous avons modélisé la série sur un intervalle d'apprentissage. Considérons maintenant la prédiction de la série à l'horizon 50 en nous basant sur le modèle combiné de la moyenne et de l'écart type. Nous calculons des intervalles de prévision à 80% [4].

```
> npred=50
> pred.lor=predict(mod.lor,n.ahead=npred,plot=FALSE,nx=0)
> dem.garch=qnorm(0.9)*pred.lor$standardDeviation
> binf.garch= pred.lor$meanForecast-dem.garch
> bsup.garch=pred.lor$meanForecast+dem.garch
```

En vue de comparer les prévisions, nous calculons également les intervalles de prévision basés sur la modélisation MA(4) initiale qui ignore l'hétéroscédasticité :

```
> pred.arima=predict(mod.r.lor,n.ahead=npred)
> str(pred.arima)

List of 2
 $ pred: Time-Series [1:50] from 327 to 376: 0.32 0.779 0.62 -1.004 0 ...
 $ se  : Time-Series [1:50] from 327 to 376: 2.35 2.4 2.43 2.43 2.5 ...

> demi=qnorm(0.9)*pred.arima$se
> binf.arima=pred.arima$pred-demi
> bsup.arima=pred.arima$pred+demi
> mat.p=cbind(bsup.garch,binf.garch,binf.arima,bsup.arima,r.lor.1[1:npred])
```

Enfin, superposons la série réalisée et les intervalles de prévision obtenus par les deux méthodes :

4. Habituellement on calcule des intervalles à 95%. Or, pour se représenter la qualité d'un ajustement ou d'une prédiction, on compare la proportion de points observés dans un intervalle à la proportion théorique. Si l'on fait ce calcul sur un nombre limité d'observations, moins de 50 par exemple, les nombres sont très petits ; c'est pourquoi nous examinons des intervalles à 80%.

```
> matplot(1:npred,mat.p,type='l',col='black',lty=c(1,1,2,2,3),
+   lwd=2,xlab="prévision à l'horizon 50",ylab="rendement")
> leg.txt=c("ARMA-GARCH","ARMA","réalisation")
> legend(14,3,leg.txt,lty=c(1,2,3))
```

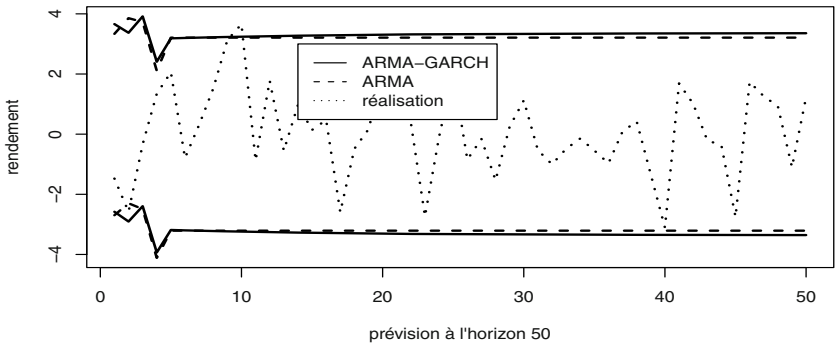

Fig. 12.11 – L'Oréal : intervalle de prévision à 80% et réalisation du rendement.

Nous observons (fig. 12.11) que l'intervalle qui incorpore l'hétéroscédasticité conditionnelle est un peu plus large que celui qui l'ignore, mais que la série reste largement à l'intérieur des deux intervalles. Les limites de l'intervalle pour le modèle MA sont des droites à partir de l'horizon 4, résultat attendu, vu les formules de prévision de ce modèle (cf. section 4.4.4).
Les proportions observées sont de 0.96 pour la modélisation ARIMA et 0.98 pour la modélisation ARMA-GARCH (SiteST). Elles sont bien supérieures à 0.8.

Exercice 12.4
Dans le traitement de L'Oréal, on a enchaîné la modélisation ARMA de la série et la modélisation GARCH du résidu de cet ajustement.
- Effectuer une modélisation GARCH du rendement sans passer par l'étape de modélisation par un ARMA.
- Superposer sur un même graphique : la série à prédire et les intervalles de prévision (à 90%) par un ARMA et par un GARCH. Commenter.

12.9 Etude du rendement de Danone

Examinons le rendement de l'action Danone à partir du 2 juin 2007. Comme précédemment, nous utilisons comme trajectoire d'apprentissage la série du rendement moins les 50 dernières valeurs. Celles-ci serviront à évaluer la qualité prédictive du modèle.

```
> r.dan=rangeIts(r.csdl[,"Danone"],start="2007-12-28")
> r.dan.0=r.dan[1:(length(r.dan)-50)]
> r.dan.1=r.dan[(length(r.dan)-49):length(r.dan)]
```

12.9.1 Modélisation

Modélisation du rendement. Commençons par examiner s'il doit être modélisé par un ARMA.

```
> xy.acfb(r.dan.0,numer=FALSE)
```

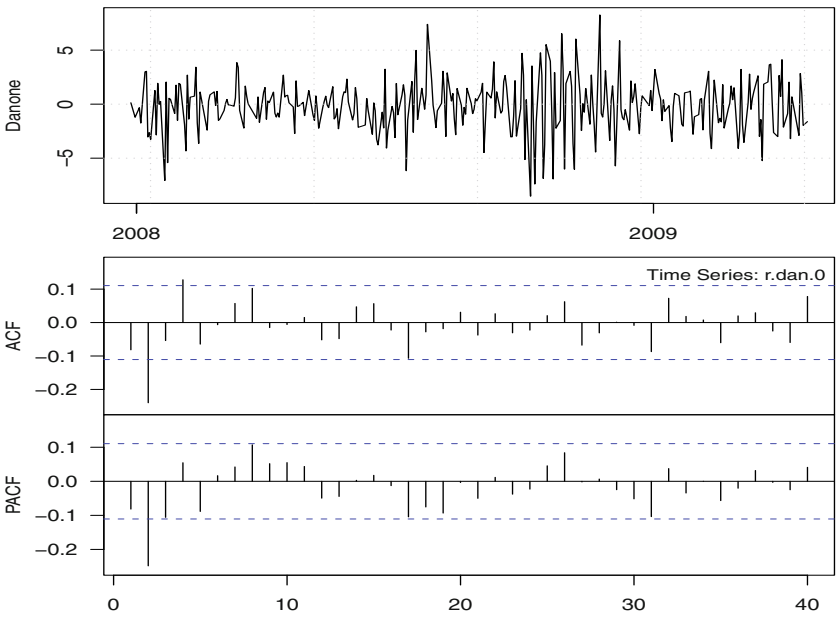

Fig. 12.12 – ACF et PACF du rendement de Danone.

La figure 12.12 suggère un AR(2). Après quelques essais autour de ce modèle, nous concluons qu'un AR(2) avec une moyenne nulle convient.

```
> mod3=Arima(r.dan.0@.Data order=c(2,0,0),include.mean= FALSE)
> summary(mod3)

...
Coefficients:
          ar1      ar2
      -0.0954  -0.2414
s.e.   0.0536   0.0536

sigma^2 estimated as 5.488:  log likelihood = -742.42
AIC = 1490.84   AICc = 1490.91   BIC = 1502.21

In-sample error measures:
         ME        RMSE         MAE         MPE        MAPE
 -0.2214985   2.3426107   1.7783679  91.4725643 137.0712264
```

```
      MASE
  0.6688398
```

```
> rr.dan=residuals(mod3)
> t(Box.test.2(rr.dan,c(6,12,18),type="Box-Pierce",decim=8,fitdf=2))
```

```
             [,1]       [,2]       [,3]
Retard  6.0000000 12.0000000 18.0000000
p-value 0.2318165  0.2564239  0.2719584
```

```
> t_stat(mod3)
```

```
             ar1       ar2
t.stat -1.777352 -4.502893
p.val   0.075510  0.000007
```

Nous conservons les résidus `rr.dan` pour y tester la présence d'hétéroscédasticité conditionnelle :

```
> ArchTest(rr.dan,lag=20)

        ARCH LM-test; Null hypothesis: no ARCH effects

data:  rr.dan
Chi-squared = 42.0472, df = 20, p-value = 0.002727
```

Nous rejetons l'hypothèse d'homoscédasticité des résidus et tâchons de modéliser cette hétéroscédasticité conditionnelle.

Modélisation de l'hétéroscédasticité du rendement. Essayons l'ajustement par un modèle GARCH$(1,1)$ qui présente une certaine souplesse sans être compliqué :

```
> modarch=garchFit(~garch(1,1),data=rr.dan,trace=FALSE,include.mean=FALSE,
+                  na.action=na.pass)
> summary(modarch)

...
Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
omega    0.14855     0.09914    1.498  0.13402
alpha1   0.10296     0.03151    3.267  0.00109 **
beta1    0.87372     0.03520   24.821  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 -724.969    normalized:  -2.217031
```

```
Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  8.841329  0.01202624
 Shapiro-Wilk Test  R    W      0.993514  0.1729387
 Ljung-Box Test     R    Q(10)  13.52768  0.1956457
 Ljung-Box Test     R    Q(15)  14.67395  0.475148
 Ljung-Box Test     R    Q(20)  20.75077  0.4119305
 Ljung-Box Test     R^2  Q(10)  5.285152  0.8713339
 Ljung-Box Test     R^2  Q(15)  9.10421   0.8720058
 Ljung-Box Test     R^2  Q(20)  11.70921  0.9257213
 LM Arch Test       R    TR^2   6.769144  0.8724848

Information Criterion Statistics:
     AIC      BIC      SIC     HQIC
4.452410 4.487180 4.452244 4.466284
```

Mis à part la non-normalité des résidus suggérée par le test de Jarque-Bera, les résultats sont assez satisfaisants. Estimons maintenant la variance de long terme :

```
> var.marg.est(modarch)

variance marginale :  6.370827
```

L'estimation est positive. Nous pouvons combiner les deux modèles.

Modélisation simultanée du rendement et de son hétéroscédasticité. Nous estimons l'ARMA/GARCH combinant le modèle du rendement et celui de l'hétéroscédasticité de son résidu :

```
> mod.dan=garchFit(~arma(2,0)+garch(1,1),data=r.dan.0@.Data,trace=FALSE,
+                 include.mean=FALSE,na.action=na.pass)
> summary(mod.dan)

...
Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
ar1     -0.04620     0.05830   -0.792 0.428145
ar2     -0.18428     0.05633   -3.272 0.001069 **
omega    0.14293     0.09459    1.511 0.130766
alpha1   0.10282     0.03060    3.360 0.000778 ***
beta1    0.87461     0.03404   25.692  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log Likelihood:
 -724.0263    normalized:  -2.214148

Standardised Residuals Tests:
```

```
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  7.083343  0.02896486
 Shapiro-Wilk Test  R    W      0.9936353 0.1845585
 Ljung-Box Test     R    Q(10)  10.88640  0.3664366
 Ljung-Box Test     R    Q(15)  12.09044  0.6721713
 Ljung-Box Test     R    Q(20)  17.37392  0.6285767
 Ljung-Box Test     R^2  Q(10)  5.851919  0.827547
 Ljung-Box Test     R^2  Q(15)  10.73834  0.7709128
 Ljung-Box Test     R^2  Q(20)  13.1406   0.8712703
 LM Arch Test       R    TR^2   8.36496   0.7559994

Information Criterion Statistics:
     AIC      BIC      SIC     HQIC
4.458877 4.516827 4.458418 4.482000
```

La normalité du résidu est incertaine : rejetée par le test de Jarque-Bera et acceptée par celui de Shapiro-Wilk. Les résultats des tests de blancheur et d'homoscédasticité sont satisfaisants. Certes `omega` n'est pas significatif, mais le modèle n'a pas de sens sans un tel paramètre. Le paramètre `ar1` n'est pas significatif. Finalement, le modèle ARMA/GARCH retenu pour le rendement de Danone est :

$$y_t = -0.0462\, y_{t-1} - 0.1843\, y_{t-2} + \epsilon_t \tag{12.16}$$

$$\epsilon_t = \sigma_t z_t \tag{12.17}$$

$$\sigma_t^2 = 0.1429 + 0.1028\, \epsilon_{t-1} + 0.8746\, \sigma_{t-1}^2. \tag{12.18}$$

Le pourcentage d'observations dans l'intervalle à 80% est 79.82 (SiteST), chiffre satisfaisant.

12.9.2 Prédiction du rendement

Suivant la même démarche que pour L'Oréal, nous calculons des intervalles de prévision à 80%, à l'horizon 50, basés sur la modélisation (12.16) :

```
> npred=50
> pred.dan=predict(mod.dan,n.ahead=npred,plot=FALSE,nx=0)
> dem.garch=qnorm(0.9)*pred.dan$standardDeviation
> binf.garch=pred.dan$meanForecast-dem.garch
> bsup.garch=pred.dan$meanForecast+dem.garch
```

Nous calculons également des intervalles basés sur la modélisation par un AR(2), objet `mod3` :

```
> pred.dan.ar=predict(mod3,n.ahead=npred,se.fit=TRUE)
> dem.ar=qnorm(0.9)*pred.dan.ar$se
> binf.ar=pred.dan.ar$pred-dem.ar
> bsup.ar=pred.dan.ar$pred+dem.ar
```

Enfin nous superposons sur un même graphique la série observée et les deux intervalles de prédiction (fig. 12.13). Les pourcentages de points observés sont de 94 dans les deux intervalles (SiteST). Comme pour L'Oréal (fig. 12.11), nous observons des intervalles très larges.

```
> mat.p=cbind(bsup.garch,binf.garch,binf.ar,bsup.ar,r.dan.1[1:npred])
> matplot(1:npred,mat.p,type='l',col='black',lty=c(1,1,2,2,3),
+   lwd=2,xlab="horizon 50",ylab="rendement")
> leg.txt=c("ARMA-GARCH","AR","réalisation")
> legend(14,3,leg.txt,lty=c(1,2,3))
```

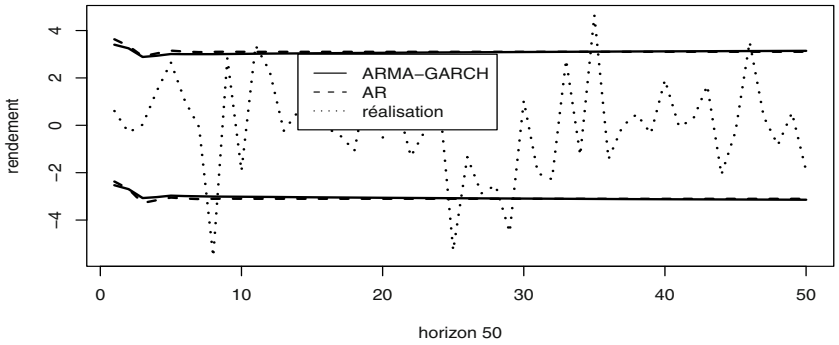

Fig. 12.13 – Danone : prévision (80%) et réalisation.

Exercice 12.5

Dans le traitement de Danone, on a enchaîné la modélisation ARMA de la série et la modélisation GARCH du résidu de cet ajustement.

- Effectuer une modélisation GARCH du rendement sans passer par l'étape de modélisation par un ARMA.
- Superposer sur un même graphique : la série à prédire et les intervalles de prévision (à 90%) par un ARMA, un ARMA-GARCH et un GARCH. Commenter.

Bibliographie

Alexander C. (2008). *Market Risk Analysis. Vol. II : Practical financial econometrics*. Wiley.

Anderson O.D. (1976). *Time Series Analysis and Forecasting : The Box-Jenkins approach*. Butterworths.

Bouleau N. (2002). *Probabilités de l'Ingénieur, Variables aléatoires et simulation*. Hermann, 2 ed.

Bourbonnais R. & Terraza M. (2008). *Analyse des séries temporelles - Applications à l'économétrie et à la gestion*. Dunod, 2 ed.

Box G.E.P., Jenkins G.M. & Reinsel G.C. (1999). *Time Series Analysis - Forecasting and Control*. Prentice Hall, 3 ed.

Brocklebank J.C. & Dickey D.A. (2003). *SAS for for Forecasting Time Series*. SAS Institute, 2 ed.

Brockwell P. & Davis R. (2002). *Introduction to Time Series and Forecasting*. Springer.

Brockwell P.J. & Davis R.A. (1991). *Time Series and Forecasting Methods*. Springer, 2 ed.

Cornillon P.A., Guyader A., Husson F., Jégou N., Josse J., Kloareg M., Matzner-Løber E. & Rouvière L. (2010). *Statistiques avec R*. PUR Rennes.

Cornillon P.A. & Matzner-Løber E. (2010). *Régression avec R*. Springer.

CRAN (2010a). *Contributed Documentation*. http://cran.r-project.org/other-docs.html.

CRAN (2010b). Task View : Time Series Analysis. http://cran.r-project.org/web/views/TimeSeries.html.

Cryer J.D. & Chan K.S. (2008). *Time Series Analysis With Applications in R*. Springer, 2 ed.

Dalgaard P. (2008). *Introductory Statistics with R*. Springer, 2 ed.

Enders W. (1995). *Applied Econometric Time Series*. John Wiley.

Farnsworth G.V. (2008). Econometrics in R. http://cran.r-project.org/doc/contrib/Farnsworth-EconometricsInR.pdf.

Franses P.H. (1998). *Time series models for business and economic forecasting*. Camridge University Press.

Fuller W.A. (1996). *Introduction to Statistical Time Series*. Wiley, 2 ed.

Gourieroux C. & Monfort A. (1995). *Séries temporelles et modèles dynamiques*. Economica, 2 ed.

Hamilton J.D. (1994). *Time Series Analysis*. Princeton University Press.

Hiebeler D. (2010). Matlab/R reference. http://cran.r-project.org/doc/contrib/Hiebeler-matlabR.pdf.

Hyndman R.. & Khandakar Y. (2008). Automatic time series forecasting : The forecast package for R. *Journal of Statistical Software*, **27**(3).

Hyndman R.J., Koehler A.B., Ord J.K. & Snyder R.D. (2008). *Forecasting with exponential smoothing : the state space approach*. Springer.

Kennedy P. (2003). *A guide to Econometrics*. Blackwell, 5 ed.

Kennedy W.J. & Gentle J.E. (1980). *Statistical Computing*. Marcel Dekker.

Kuhn M. (2010). A summary of the international standard date and time notation. http://www.cl.cam.ac.uk/~mgk25/iso-time.html.

Kwiatkowski D., Phillips P.C.B., Schmidt P. & Shin Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root : How sure are we that economic time series have a unit root ? *Journal of Econometrics*, **54**, 159–178.

Ladiray D. & Quenneville B. (2001). *Seasonal Adjustment with the X-11 Method*. Lecture Notes on Statistics, Springer.

Maindonald J. (2010). *Using R for Data Analysis and Graphics - An Introduction*. Cambridge Univ. Press., 3 ed.

Maindonald J. & Braun J. (2003). *Data Analysis Using R*. Cambridge Univ. Press.

McLeod A.I. & Zhang Y. (2008). Improved subset autoregression : With R package. *Journal of Statistical Software*, **28**(2).

Muenchen B. (2009). *R for SAS and SPSS Users*. Springer.

NIST/SEMATECH (2006). e-handbook of statistical methods. http://www.itl.nist.gov/div898/handbook/.

Pankratz A. (1991). *Forecasting with dynamic regression models*. Wiley.

Paradis E. (2005). *R pour les débutants*. http://cran.r-project.org/doc/contrib/Paradis-rdebuts_fr.pdf.

Pfaff B. (2006). *Analysis of Integrated and Cointegrated Time Series with R*. Springer.

Ricci V. (2010). R functions for time series analysis. http://cran.r-project.org/doc/contrib/Ricci-refcard-ts.pdf.

Robinson A. & Schloesing A. (2008). Brise glace-R (ouvrir la voie aux pôles statistiques). http://cran.r-project.org/doc/contrib/IceBreak_fr.pdf.

Ruppert D. (2004). *Statistics and Finance : an introduction*. Springer.

Sheather S.J. (2009). *A Modern Approach to Regression with R*. Springer.

Shumway R. & Stoffer D. (2006). *Time Series Analysis and Its Applications - With R Examples*. Springer, 2 ed.

Tsay R.S. (2005). *Analysis of Financial Time Series*. Wiley, 2 ed.

Verzani J. (2002). SimpleR, Using R for Introductory Statistics. http://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf.

Wapedia (2010). Wiki : Heure unix. http://wapedia.mobi/fr/Heure_Unix.

Wei W. (2006). *Time Series Analysis : Univariate and Multivariate Methods*. Addison-Wesley, 2 ed.

Wuertz D. & Rmetrics Foundation (2010). http://www.rmetrics.org.

Zivot E. & Wang J. (2006). *Modeling financial time series with S-Plus*. Springer, 2 ed.

Index