

# Séries temporelles

Victor DUC

2021

# Table des matières

<b>I</b>	<b>Introduction aux séries temporelles</b>	<b>7</b>
<b>1</b>	<b>Décompositions</b>	<b>9</b>
1.1	Décomposition additive . . . . .	9
1.1.1	Filtres ou moyennes mobiles . . . . .	9
1.1.2	Chronogramme . . . . .	9
1.2	Décompositions multiplicatives . . . . .	10
1.2.1	Transformation de Box-Cox . . . . .	10
<b>2</b>	<b>a</b>	<b>11</b>
2.1	Opérateurs . . . . .	11
2.1.1	Opérateur identité . . . . .	11
2.1.2	Opérateur retard . . . . .	11
2.1.3	Opérateur différence finie . . . . .	12
2.1.4	Opérateur différence saisonnière finie . . . . .	12
<b>3</b>	<b>Critères d'information</b>	<b>14</b>
3.1	Critère d'information d'Akaike AIC . . . . .	14
3.1.1	Critère d'information d'Akaike corrigé AICc . . . . .	14
3.2	Critère d'information bayésien BIC . . . . .	14
<b>4</b>	<b>R squared sigma</b>	<b>15</b>
4.1	Test de vraisemblance . . . . .	15
4.2	Décomposition usuelle . . . . .	15
4.3	Séries temporelles financières . . . . .	15
<b>5</b>	<b>Paramètres graphiques</b>	<b>16</b>
<b>II</b>	<b>Séries temporelles stationnaires</b>	<b>17</b>
<b>6</b>	<b>Fonctions théoriques et empiriques d'un processus stationnaire</b>	<b>19</b>
6.1	Fonctions théoriques . . . . .	19
6.1.1	Autocovariance théorique . . . . .	19
6.1.1.1	Propriétés . . . . .	19
6.1.2	Autocorrélation théorique . . . . .	20
6.1.2.1	Autocorrélation partielle théorique . . . . .	20
6.1.2.1.1	Propriétés . . . . .	21
6.2	Fonctions empiriques . . . . .	21
6.2.1	Autocovariance empirique . . . . .	21
6.2.2	Autocorrélation empirique . . . . .	22
<b>7</b>	<b>Analyse spectrale</b>	<b>23</b>
7.1	Densité spectrale . . . . .	23
7.1.1	Propriétés . . . . .	23

<b>8</b>	<b>Test de stationnarité</b>	<b>24</b>
8.1	Test de Kwiatkowski-Phillips-Schmidt-Shin (KPSS)	24
<b>9</b>	<b>Bruit blanc</b>	<b>25</b>
9.1	Bruit blanc faible	25
9.2	Bruit blanc fort	25
9.3	Test de bruit blanc ou test de blancheur	26
9.3.1	Test d'un coefficient d'autocorrélation empirique d'ordre fixé à la fois	26
9.3.1.1	Principe	26
9.3.1.2	Implémentation	27
9.3.1.3	Limites	27
9.3.2	Test d'autocorrélation du Portemanteau	28
9.3.2.1	Principe	28
9.3.2.2	Test de Box-Pierce (1970)	28
9.3.2.3	Test de Ljung-Box (1978)	28
9.3.2.4	Implémentation	28
9.3.3	Test de normalité (des résidus)	29
9.3.3.1	Tests basés sur les quantiles	29
9.3.3.1.1	QQ-plot	29
9.3.3.1.2	Test de Shapiro-Wilk	29
9.3.3.2	Tests basés sur les coefficient d'asymétrie (skewness) et d'aplatissement (kurtosis)	29
9.3.3.2.1	Test de Jarque-Bera	29
9.3.3.2.2	Test d'Agostino	29
9.3.3.3	Test de Lilliefors basé sur la fonction de répartition (inspiré du test de Kolmogov-Smirnov)	29
9.3.4	Comparaison des tests de normalité	29
9.4	Modélisation par un bruit blanc	29
9.4.1	Méthode	30
9.5	Exemple de modélisation par un bruit blanc	31
9.5.1	Test de stationnarité	34
9.5.2	Test de blancheur	34
9.5.2.1	Test d'autocorrélation du Portemanteau	34
9.5.2.2	Test d'un coefficient d'autocorrélation empirique d'ordre fixé à la fois	35
9.5.3	Modèle	36
9.5.4	Test de normalité du bruit blanc	36
9.5.4.1	Test de Shapiro-Wilk	36
9.5.4.2	Test d'Agostino	36
9.5.5	Prévisions	37
<b>10</b>	<b>Modèles pour l'homoscédasticité : Modèle autorégressif et moyenne mobile ARMA</b>	<b>39</b>
10.1	Théorème de décomposition de Wold	39
10.2	Processus autorégressifs AR	41
10.2.1	Espérance	41
10.2.2	Fonction d'autocovariance et d'autocorrélation théoriques	42
10.2.2.1	Par MA infini	42
10.2.2.2	Par récurrence	42
10.2.3	Processus autorégressif d'ordre 1 AR(1)	42
10.2.3.1	Fonction d'autocovariance et d'autocorrélation théoriques	44
10.2.3.1.1	Méthode par MA()	44
10.2.3.1.2	Méthode par récurrence	44
10.3	Processus moyenne mobile MA	45
10.3.1	Espérance	45
10.3.1.1	Fonction d'autocovariance et d'autocorrélation théorique	45
10.3.1.2	Ordre fini	46
10.3.1.3	Ordre infini	46
10.3.2	Processus moyenne mobile d'ordre 1 MA(1)	47
10.3.2.1	Fonction d'autocovariance et d'autocorrélation théorique	47

10.4	Processus mixtes . . . . .	47
10.5	Modélisation par un modèle ARMA . . . . .	50
10.5.1	Modélisation automatique . . . . .	50
10.6	Exemple de modélisation par un modèle ARMA(p,q) . . . . .	51
10.6.1	Tests de stationnarité et test de blancheur . . . . .	54
10.6.1.1	Test de stationnarité . . . . .	54
10.6.1.2	Test de blancheur . . . . .	54
10.6.2	Modélisation manuelle . . . . .	55
10.6.2.1	Identification des degrés . . . . .	55
10.6.2.1.1	Identification du degré d'un MA et d'un AR . . . . .	55
10.6.2.1.2	Identification des degrés d'un processus mixte ARMA . . . . .	56
10.6.2.2	Estimation des paramètres . . . . .	56
10.6.2.2.1	Estimation des paramètres du modèle AR(1) . . . . .	56
10.6.2.2.2	Estimation des paramètres du modèle MA(3) . . . . .	56
10.6.2.3	Réduction du modèle . . . . .	57
10.6.2.3.1	Réduction du modèle AR(1) . . . . .	57
10.6.2.3.2	Réduction du modèle MA(3) . . . . .	57
10.6.2.4	Validation du modèle . . . . .	57
10.6.2.4.1	Validation du modèle AR(1) . . . . .	57
10.6.2.4.2	Validation du modèle MA(3) . . . . .	58
10.6.2.5	Explicitation des modèles . . . . .	60
10.6.2.5.1	Explicitation du modèle AR(1) . . . . .	60
10.6.2.5.2	Explicitation du modèle MA(3) . . . . .	61
10.6.3	Modélisation automatique . . . . .	62
10.6.3.1	Modélisation automatique par un modèle AR . . . . .	62
10.6.3.1.1	Identification du degré, estimation des paramètres et réduction du modèle . . . . .	62
10.6.3.1.2	Validation du modèle AR(3) . . . . .	63
10.6.3.2	Modélisation automatique par un processus AR(I)MA . . . . .	64
10.6.3.2.1	Identification du degré, estimation des paramètres et réduction du modèle . . . . .	64
10.6.4	Comparaison des modèles AR(1), MA(3) et AR(3) et sélection du modèle . . . . .	65
10.6.4.1	Critères d'information . . . . .	65
10.6.4.1.1	Tests de normalité . . . . .	65
10.6.4.1.1.1	Test de Shapiro-Wilk . . . . .	65
10.6.4.1.1.2	Test d'Agostino . . . . .	65
10.6.4.1.2	Comparaison des critères d'information . . . . .	67
10.6.4.2	Pouvoir prédictif . . . . .	67
10.6.4.2.1	Ajustement . . . . .	67
10.6.4.2.2	Sur échantillon témoin . . . . .	68
10.6.5	Prévisions du modèle AR(3) . . . . .	69
<b>11</b>	<b>Modèles pour l'hétéroscédasticité conditionnelle</b>	<b>71</b>
11.1	Modèles GARCH . . . . .	71
11.1.1	Modèles ARCH . . . . .	72
11.1.1.1	Modèle ARCH(1) . . . . .	72
11.2	Exemple de modélisation par processus GARCH . . . . .	72
<b>III</b>	<b>Séries temporelles non stationnaires</b>	<b>73</b>
<b>12</b>	<b>Taxinomie de la non stationnarité</b>	<b>75</b>
12.1	Non stationnarité du premier ordre non stationnarité en moyenne . . . . .	75
12.1.1	Non stationnarité déterministe ou stationnarité à une tendance près ( <i>Trend Stationnary</i> ) . . . . .	75
12.1.1.1	Processus à tendance déterministe . . . . .	75
12.1.1.1.1	Processus à tendance déterministe polynomiale . . . . .	75
12.1.1.1.1.1	Processus à tendance déterministe linéaire . . . . .	76
12.1.1.1.2	Processus à tendance déterministe périodique . . . . .	76
12.1.1.2	Processus à saisonnalité déterministe . . . . .	77

12.2	Non stationnarité du deuxième ordre ou en variance NSV . . . . .	79
12.2.1	Non stationnarité stochastique ou stationnarité en différences ( <i>Difference Stationnary</i> ) . . . .	79
12.2.1.1	Processus à tendance stochastique . . . . .	79
12.2.1.1.1	Marche aléatoire . . . . .	79
12.2.2	<i>Seasonally Difference Stationnary (SDS)</i> . . . . .	80
12.2.2.1	Processus à saisonnalité stochastique . . . . .	80
12.2.3	Variance inconditionnelle variante (VIV) . . . . .	81
<b>13</b>	<b>Test de non stationnarité ou test de racine unitaire</b>	<b>82</b>
13.1	Test de racine unitaire non saisonnière . . . . .	82
13.1.1	Test de Dickey-Fuller (DF) . . . . .	82
13.1.1.1	Test de Dickey-Fuller augmenté (ADF) . . . . .	82
13.1.2	Test de Phillips-Perron (PP) . . . . .	82
13.1.3	Test d'Elliott-Rothenberg-Stock (ERS) . . . . .	82
13.2	Test de racine unitaire saisonnière . . . . .	82
13.2.1	Test de Canova-Hansen (CH) . . . . .	82
13.2.2	Test de Osborn-Chui-Smith-Birchenhall (OCSB) . . . . .	82
<b>14</b>	<b>Tendance</b>	<b>83</b>
14.1	Détection d'une composante tendancielle . . . . .	83
14.2	Identification de la nature de la tendance . . . . .	84
14.2.1	Méthode d'identification avec les tests ADF et KPSS . . . . .	84
14.2.2	Méthode d'identification avec les tests PP et KPSS . . . . .	84
14.3	Modélisation d'une tendance déterministe : processus ARMAX . . . . .	84
14.3.1	Par régression linéaire . . . . .	84
14.3.1.1	Modélisation de la fonction déterministe . . . . .	84
14.3.1.2	Modélisation de la série résiduelle . . . . .	85
14.4	Modélisation d'une tendance stochastique : processus ARIMA . . . . .	87
14.5	Exemples de modélisation d'une tendance . . . . .	88
14.5.1	Détection d'une tendance . . . . .	91
14.5.2	Identification de la nature de la tendance . . . . .	91
14.5.3	Exemple de modélisation d'une tendance déterministe . . . . .	92
14.5.3.1	Modélisation de la fonction déterministe . . . . .	92
14.5.3.1.1	Par régression d'ordre 1 . . . . .	94
14.5.3.1.2	Par régression d'ordre 2 . . . . .	94
14.5.3.1.3	Comparaison des modèles de régression d'ordres 1 et 2 . . . . .	96
14.5.3.2	Analyse des résidus du modèle de régression d'ordre 2 . . . . .	98
14.5.3.2.1	Test de stationnarité des résidus . . . . .	98
14.5.3.2.2	Test de non corrélation et d'homoscédasticité . . . . .	99
14.5.3.2.3	Modélisation des résidus . . . . .	99
14.5.3.2.3.1	Test de blancheur . . . . .	100
14.5.3.2.3.2	Modélisation par processus ARMA . . . . .	102
14.5.3.2.3.2.1	Identification des degrés . . . . .	102
14.5.3.2.3.2.2	Estimation des paramètres du modèle ARMA(1,1) . . . . .	102
14.5.3.2.3.2.3	Simplification du modèle ARMA(1,1) en AR(1) . . . . .	102
14.5.3.2.3.2.4	Validation du modèle simplifié AR(1) . . . . .	103
14.5.3.3	Explicitation du modèle final ARMAX(2,1) . . . . .	104
14.5.3.4	Annexe - Modélisation par processus ARMA(2,2) . . . . .	106
14.5.3.4.1	Simplification du modèle ARMA(2,2) en ARMA(2,1) . . . . .	106
14.5.3.4.2	Validation du modèle simplifié ARMA(2,1) . . . . .	107
14.5.3.4.3	Comparaison des modèles ARMA(2,2) et ARMA(2,1) et sélection . . . . .	108
14.5.3.4.3.1	Test de normalité des résidus . . . . .	108
14.5.3.4.3.2	Critères d'information . . . . .	108
14.5.3.4.3.3	Test de rapport de vraisemblance . . . . .	108
14.5.3.5	Annexe 2 . . . . .	110
14.5.4	Exemple de modélisation d'une tendance stochastique . . . . .	113
14.5.4.1	Identification du degré de différenciation . . . . .	113

14.5.4.2	Modélisation de la série résiduelle par processus ARMA . . . . .	114
14.5.4.2.1	Identification du degré . . . . .	114
14.5.4.2.2	Estimation des paramètres . . . . .	114
14.5.4.2.3	Simplification du modèle . . . . .	114
14.5.4.2.4	Validation du modèle . . . . .	114
14.5.4.3	Explicitation du modèle final ARIMA(0,1,1) . . . . .	115
14.5.5	Comparaison des modélisations déterministe et stochastique . . . . .	117
14.5.5.1	Test de normalité des résidus . . . . .	117
14.5.5.2	Prévisions des modèles . . . . .	118
14.5.5.3	Critères d'information . . . . .	120
14.5.5.4	Indicateurs d'écart . . . . .	120
14.6	Exemple de modélisation d'une tendance déterministe . . . . .	121
14.6.1	Modélisation de la fonction déterministe . . . . .	121
14.6.2	Modélisation des résidus . . . . .	122
<b>15</b>	<b>Saisonnalité</b>	<b>123</b>
15.1	Détection d'une composante saisonnière . . . . .	123
15.2	Identification de la saisonnalité . . . . .	123
15.2.1	Périodogramme . . . . .	123
15.2.2	Validation de la saisonnalité . . . . .	123
15.3	Identification de la nature de la saisonnalité . . . . .	123
15.4	Modélisation d'une saisonnalité déterministe : processus SARMAX . . . . .	124
15.5	Modélisation d'une saisonnalité stochastique : processus SARIMA . . . . .	124
15.6	Exemple de modélisation d'une saisonnalité . . . . .	124
<b>16</b>	<b>Tendance et saisonnalité : processus SARIMA(p,d,q)(P,D,Q)[r]</b>	<b>125</b>
16.1	Exemple de modélisation d'une tendance et d'une saisonnalité . . . . .	126
16.1.1	Modélisation par transformation de Box-Cox (tendance puis saisonnalité) . . . . .	130
16.1.1.1	Détection d'une tendance (et d'une saisonnalité) . . . . .	133
16.1.1.2	Étude de la composante tendancielle . . . . .	134
16.1.1.2.1	Identification de la nature de la tendance . . . . .	134
16.1.1.2.2	Modélisation de la tendance (déterministe) linéaire . . . . .	136
16.1.1.3	Étude de la composante saisonnière . . . . .	139
16.1.1.3.1	Identification de la saisonnalité . . . . .	141
16.1.1.3.2	Validation de la saisonnalité . . . . .	141
16.1.1.3.3	Identification de la nature de la saisonnalité . . . . .	142
16.1.1.3.4	Modélisation de la saisonnalité déterministe . . . . .	142
16.1.1.3.4.1	Par moyennes saisonnières . . . . .	142
16.1.1.3.4.1.1	Analyse des résidus . . . . .	143
16.1.1.3.4.1.2	Modèle complet SARIMA(1,)(1,0,4)[12] avec fonction Arima() . . . . .	146
16.1.1.3.4.1.3	Prévisions . . . . .	150
16.1.1.3.4.2	Par régression harmonique 1 . . . . .	152
16.1.1.3.4.2.1	Analyse des résidus . . . . .	152
16.1.1.3.4.3	Par régression harmonique 2 . . . . .	156
16.1.1.3.4.3.1	Analyse des résidus . . . . .	156
16.1.1.3.4.3.2	Modèle complet SARIMA(1,)(14,0,2)[12] avec fonction Arima() . . . . .	162
16.1.1.3.5	Annexe : Modélisation par saisonnalité stochastique . . . . .	166
16.1.1.3.5.1	Explicitation du modèle SARIMA(1,0,12)(0,1,0)[12] . . . . .	168
16.1.1.3.5.2	Analyse des résidus . . . . .	169
16.1.1.3.5.3	Prévisions . . . . .	170
16.1.2	Modélisation par transformation de Box-Cox (saisonnalité puis tendance) . . . . .	170
16.1.2.1	Détection d'une saisonnalité (et d'une tendance) . . . . .	171
16.1.2.2	Étude de la composante saisonnière . . . . .	171
16.1.2.2.1	Identification de la saisonnalité . . . . .	171
16.1.2.2.2	Validation de la saisonnalité . . . . .	172
16.1.2.2.3	Identification de la nature de la saisonnalité . . . . .	173
16.1.2.2.4	Modélisation de la saisonnalité déterministe . . . . .	173

16.1.2.2.4.1	Par moyennes saisonnières . . . . .	173
16.1.2.2.4.1.1	Analyse des résidus . . . . .	174
16.1.2.2.4.1.2	Étude de la composante tendancielle . . . . .	175
16.1.2.2.4.1.2.1	Identification de la nature de la tendance . . . . .	175
16.1.2.2.4.1.2.2	Identification du degré de différenciation . . . . .	177
16.1.2.2.4.1.2.3	Simplification du modèle . . . . .	179
16.1.2.2.4.1.2.4	Comparaison des modèles . . . . .	180
16.1.2.2.4.1.2.5	Explicitation du modèle SARIMA(12,0,1)(0,0,0)[12] final	182
16.1.2.2.4.1.2.6	Prévisions . . . . .	183
16.1.2.2.4.2	Par régression harmonique . . . . .	184
16.1.2.2.4.2.1	Analyse des résidus . . . . .	185
16.1.2.2.4.2.2	Étude de la composante tendancielle . . . . .	186
16.1.2.2.4.2.2.1	Identification de la nature de la tendance . . . . .	186
16.1.2.2.4.2.2.2	Identification du degré de différenciation . . . . .	188
16.1.2.2.4.2.2.3	Comparaison des modèles . . . . .	192
16.1.2.2.4.2.2.4	Modèle complet SARIMA(13,0,2)(0,1,0)[12] . . . . .	193
16.1.2.2.4.2.2.5	Prévisions . . . . .	195
16.1.3	Modélisation par transformation de Box-Cox (Tendance et saisonnalité stochastiques) . . . .	196
16.1.3.1	Prévisions . . . . .	198
16.1.4	Tendance et saisonnalité stochastiques . . . . .	199

## Démonstrations

204

## Première partie

# Introduction aux séries temporelles

Dans **R** la classe **ts** permet de gérer facilement les séries temporelles. L'attribut **time** stocke les dates successives des observations. L'attribut **frequency** indique la fréquence annuelle des relevés. Si les données ne sont pas de classe **ts**, alors on les convertit en classe **ts**.

**Implémentation** On utilise la fonction `as.ts()` du package **tolBasis**.

# Chapitre 1

## Décompositions

### 1.1 Décomposition additive

Soit  $(X_t)_{t \in T}$  une série temporelle. On décompose usuellement la série temporelle  $(X_t)_{t \in T}$  en une **composante déterministe** et une **composante stochastique** ou **aléatoire**. Plus concrètement on suppose que la composante déterministe est formée de deux fonctions  $(m, s) \in \mathbb{R}^T \times \mathbb{R}^T$  et que la composante stochastique est formée d'un processus stochastique  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t = \underbrace{m(t) + s(t)}_{\text{Composante déterministe}} + \underbrace{B_t}_{\text{Composante stochastique}}$$

- La **tendance** ou **composante tendancielle** de  $(X_t)_{t \in T}$  est la fonction  $m$ .
- La **saisonnalité** ou **composante saisonnière** de  $(X_t)_{t \in T}$  est la fonction  $s$ .
- Le **bruit** de  $(X_t)_{t \in T}$  est le processus  $(B_t)_{t \in T}$ .

La tendance est généralement définie par une fonction polynomiale, la saisonnalité par une fonction périodique et le bruit constitue des variations de faible intensité, de courte durée et de nature aléatoire.

Pour une bonne identifiabilité des termes on suppose que le bruit  $(B_t)_{t \in T}$  est centré. Notons  $r$  la période de  $s$ . On suppose de plus que

$$\forall t \in T \quad \sum_{i=1}^r s(t+i) = 0$$

**Implémentation** On utilise la fonction `decompose()` du package `stats` qui s'applique à des données de classe `ts()`. La méthode de décomposition implémentée dans la fonction `decompose()` a été proposée par Kendall et Stuart et repose sur des calculs de moyennes mobiles. En utilisant ensuite la fonction `plot()` on trace 4 sous-graphiques : de haut en bas on obtient le tracé du chronogramme, de la tendance, de la saisonnalité et de bruit. À cause du zoom sur chaque sous-graphique, toutes les composantes semblent jouer un rôle équivalent. Seule la lecture des échelles permet d'identifier la ou les composantes prépondérantes.

#### 1.1.1 Filtres ou moyennes mobiles

#### 1.1.2 Chronogramme

Lorsqu'on dispose d'une série chronologique, la première chose à faire, avant de tenter tout calcul, consiste à tracer son évolution, le chronogramme. On parle aussi de trajectoire du processus au sens où on visualise une de ses réalisations.

**Implémentation**

- Si les données sont de classe `ts`, alors on utilise la fonction `plot()` du package `graphics`.
- Si les données ne sont pas de classe `ts`, alors on dispose de trois fonctions.
  - \* On utilise la fonction `plot()` du package `graphics` avec l'argument optionnel `type="l"` (pour *lines*).
  - \* On utilise la fonction `ts.plot()` du package `stats`.
  - \* On utilise la fonction `plot.ts()` du package `stats`.

## 1.2 Décompositions multiplicatives

Il arrive que la série temporelle  $(X_t)_{t \in T}$  provienne d'un modèle multiplicatif.

**Exemple**

$$\forall t \in T \quad X_t = m(t)(1 + s(t)) \times (B_t + 1)$$

Dans les modèles multiplicatifs, la variance (donc la variabilité des observations) augmente avec le temps. On procède alors à une transformation des données pour stabiliser la variance.

### 1.2.1 Transformation de Box-Cox

#### Définition (Transformation de Box-Cox)

Soit  $(X_t)_{t \in T}$  un processus stochastique.

La **transformation de Box-Cox** de  $(X_t)_{t \in T}$  est la transformation de  $(X_t)_{t \in T}$  par la fonction suivante.

$$f_\lambda : \mathbb{R}_+ \longrightarrow \mathbb{R} \\ x \longmapsto \begin{cases} \ln(x) & \text{si } \lambda = 0 \\ \frac{x^\lambda - 1}{\lambda} & \text{si } \lambda > 0 \end{cases}$$

La transformation de Box-Cox est généralement utilisée dans le cadre de la régression linéaire pour se rapprocher au mieux des conditions de validité du modèle linéaire.

$$\forall t \in T \quad f_\lambda(X_t) = a_\lambda t + b_\lambda + \varepsilon_t$$

où  $(\varepsilon_t)_{t \in T}$  sont des variables indépendantes de même loi (gaussienne) et de variance constante.

Afin de déterminer le coefficient  $\lambda$  le mieux adapté pour transformer les données, on peut calculer la vraisemblance du modèle linéaire généré en supposant les conditions de validité du modèle linéaire vérifiées et choisir le coefficient  $\lambda$  correspondant à la vraisemblance maximale.

**Implémentation** On utilise la fonction `boxcox()` du package **MASS** automatise cette démarche.

Les transformations logarithme ( $\lambda = 0$ ) et racine sont des cas particuliers des transformations de Box-Cox.

# Chapitre 2

## a

### 2.1 Opérateurs

#### 2.1.1 Opérateur identité

**Définition (Opérateur identité)**

Soit  $(X_t)_{t \in T}$  un processus stochastique.

L'**opérateur identité** associé à  $(X_t)_{t \in T}$ , noté  $\mathbf{1}$ , est l'application suivante.

$$\begin{aligned} \mathbf{1} : \{X_t \mid t \in T\} &\longrightarrow \{X_t \mid t \in T\} \\ X_t &\longmapsto \mathbf{1}X_t := X_t \end{aligned}$$

#### 2.1.2 Opérateur retard

**Définition (Opérateur retard)**

Soit  $(X_t)_{t \in T}$  un processus stochastique.

L'**opérateur retard** associé à  $(X_t)_{t \in \mathbb{N}}$ , noté  $L$ , est l'application suivante.

$$\begin{aligned} L : \{X_t \mid t \in \mathbb{N}^*\} &\longrightarrow \{X_t \mid t \in \mathbb{N}\} \\ X_t &\longmapsto LX_t := X_{t-1} \end{aligned}$$

L'opérateur retard est l'application qui associe à la valeur présente d'un processus stochastique, la valeur précédente.

**Proposition (Puissances de l'opérateur retard)**

Soit  $(X_t)_{t \in T}$  un processus stochastique.

$$\forall (t, n) \in T \times \mathbb{N}^* \quad t - n \in T \implies L^n X_t = X_{t-n}$$

Polynôme d'opérateur retard

**Proposition (Propriétés de l'opérateur retard)**

Soit  $(X_t)_{t \in T}$  un processus stochastique.

$$\forall a \in ]-1, 1[ \quad \left( \sum_{k=0}^{+\infty} a^k L^k \right) X_t = \frac{1}{1 - aL} X_t$$

Soit  $(X_t)_{t \in T}$  un processus stochastique et  $(n, a) \in \mathbb{N} \times ]-1, 1[$ .

**Définition (Série en l'opérateur retard)**

Soit  $(a_t)_{t \in T} \in \mathbb{R}^T$ .

La série en l'opérateur retard de terme général  $a_t L^t$  est la suite  $\left( \sum_{k=0}^t a_k L^k \right)_{t \in T}$ .

### 2.1.3 Opérateur différence finie

**Définition (Opérateur différence première)**

Soit  $(X_t)_{t \in T}$  un processus stochastique.

L'opérateur différence première associé à  $(X_t)_{t \in \mathbb{N}}$ , noté  $\Delta$ , est l'application  $\mathbf{1} - L$ .

$$\begin{aligned} \Delta := \mathbf{1} - L : \quad \{X_t \mid t \in \mathbb{N}^*\} &\longrightarrow \{X_t \mid t \in \mathbb{N}\} \\ X_t &\longmapsto \Delta X_T := (\mathbf{1} - L)X_t = \mathbf{1}X_t - LX_t = X_t - X_{t-1} = \end{aligned}$$

**Définition (Opérateur différence seconde)**

Soit  $(X_t)_{t \in T}$  un processus stochastique.

L'opérateur différence seconde associé à  $(X_t)_{t \in \mathbb{N}}$ , noté  $\Delta^2$ , est l'application  $(\mathbf{1} - L)^2$ .

$$\begin{aligned} \Delta^2 := (\mathbf{1} - L)^2 : \quad \{X_t \mid t \in \mathbb{N} \setminus \{0, 1\}\} &\longrightarrow \{X_t \mid t \in \mathbb{N}\} \\ X_t &\longmapsto \Delta^2 X_T := (\mathbf{1} - L)^2 X_t = \mathbf{1}X_t - 2LX_t + L^2 X_t = X_t - 2X_{t-1} + X_{t-2} \end{aligned}$$

**Définition (Opérateur différence  $n$ -ième)**

Soit  $n \in \mathbb{N}^*$  et  $(X_t)_{t \in T}$  un processus stochastique.

L'opérateur différence  $n$ -ième associé à  $(X_t)_{t \in \mathbb{N}}$ , noté  $\Delta^n$ , est l'application  $(\mathbf{1} - L)^n$ .

$$\begin{aligned} \Delta^n := (\mathbf{1} - L)^n : \quad \{X_t \mid t \in \mathbb{N} \setminus \{0, \dots, n-1\}\} &\longrightarrow \{X_t \mid t \in \mathbb{N}\} \\ X_t &\longmapsto \Delta^n X_T := (\mathbf{1} - L)^n X_t = \end{aligned}$$

### 2.1.4 Opérateur différence saisonnière finie

**Définition (Opérateur différence saisonnière première)**

Soit  $(X_t)_{t \in T}$  un processus stochastique.

L'opérateur différence saisonnière première associé à  $(X_t)_{t \in \mathbb{N}}$ , noté  $\Delta$ , est l'application  $(\mathbf{1} - L)$ .

$$\begin{aligned} \Delta := (\mathbf{1} - L) : \quad \{X_t \mid t \in \mathbb{N} \setminus \{0\}\} &\longrightarrow \{X_t \mid t \in \mathbb{N}\} \\ X_t &\longmapsto \Delta X_T := (\mathbf{1} - L)X_t = \mathbf{1}X_t - LX_t = X_t - X_{t-1} \end{aligned}$$

**Définition (Opérateur différence saisonnière seconde)**

Soit  $(X_t)_{t \in T}$  un processus stochastique.

L'opérateur différence saisonnière seconde associé à  $(X_t)_{t \in \mathbb{N}}$ , noté  $\Delta_2$ , est l'application  $(\mathbf{1} - L^2)$ .

$$\begin{aligned} \Delta_2 := (\mathbf{1} - L^2) : \quad \{X_t \mid t \in \mathbb{N} \setminus \{0, 1\}\} &\longrightarrow \{X_t \mid t \in \mathbb{N}\} \\ X_t &\longmapsto \Delta_2 X_t := (\mathbf{1} - L^2)X_t = \mathbf{1}X_t - L^2 X_t = X_t - X_{t-2} \end{aligned}$$

**Définition (Opérateur différence saisonnière  $r$ -ième)**

Soit  $r \in \mathbb{N}^*$  et  $(X_t)_{t \in T}$  un processus stochastique.

L'opérateur différence saisonnière  $r$ -ième associé à  $(X_t)_{t \in \mathbb{N}}$ , noté  $\Delta_r$ , est l'application  $(\mathbf{1} - L^r)$ .

$$\begin{aligned} \Delta_r := (\mathbf{1} - L^r) : \quad \{X_t \mid t \in \mathbb{N} \setminus \{0, \dots, r-1\}\} &\longrightarrow \{X_t \mid t \in \mathbb{N}\} \\ X_t &\longmapsto \Delta_r X_t := (\mathbf{1} - L^r)X_t = \mathbf{1}X_t - L^r X_t \\ &= X_t - X_{t-r} \end{aligned}$$

## Chapitre 3

# Critères d'information

Les critères d'information reposent sur le calcul de la vraisemblance des données en supposant qu'elles sont gaussiennes. Si on veut utiliser les critères d'information pour comparer les modèles, il est préférable de tester au préalable si les résidus du modèle peuvent être considérées gaussiens.

### 3.1 Critère d'information d'Akaike AIC

#### 3.1.1 Critère d'information d'Akaike corrigé AICc

### 3.2 Critère d'information bayésien BIC

Le critère BIC pénalise plus fortement le nombre de paramètres du modèle.

## Chapitre 4

# R squared sigma

il faut maximiser r squared il faut minimiser sigma

### 4.1 Test de vraisemblance

Fonction `lrtest()` du package `lmtest`.  
Validation d'un modèle `gof.lag round longueur divisé par 4`.  
La mémoire = fonction, d'autocorrélation théorique.

### 4.2 Décomposition usuelle

### 4.3 Séries temporelles financières

Les séries temporelles financières sont particulièrement concernées par les modèles ARCH, car on constate des périodes de forte spéculation (variabilité élevée) suivies de périodes d'accalmie (variabilité faible).

## Chapitre 5

# Paramètres graphiques

`las` numeric in  $\{1, 2, 3, 4\}$ ; the style of axis labels.

- \* 0 : always parallel to the axis [default]
- \* 1 : always horizontal
- \* 2 : always perpendicular to the axis
- \* 3 : always vertical

Deuxième partie

**Séries temporelles stationnaires**

La notion de stationnarité est essentielle pour l'étude des séries temporelles. On cherchera en effet à se ramener à un processus stationnaire, quitte à transformer la série, par exemple en la différenciant ou en soustrayant des composantes de tendance et/ou de saisonnalité qui auront été estimées.

**Définition (Série stationnaire)**

Une **série stationnaire** est une série temporelle (ou série chronologique) qui suit un processus stationnaire *i.e.* qui est un échantillon (ou une réalisation) d'un processus stationnaire.

On peut reformuler cette définition sous la forme équivalente suivante.

Une **série stationnaire** est une série dont le processus générateur est un processus stationnaire.

Intuitivement une série stationnaire est une série dont la trajectoire fluctue autour de sa moyenne théorique constante avec une variance stable dans le temps.

Les processus stationnaires d'ordre 2 sont des processus générateurs de séries temporelles sans tendance en moyenne et sans tendance en variance. Cela ne signifie pas qu'elles présentent une représentation graphique stable.

Etudions plusieurs classes de modèles (ou processus) stationnaires.

# Chapitre 6

## Fonctions théoriques et empiriques d'un processus stationnaire

### 6.1 Fonctions théoriques

#### 6.1.1 Autocovariance théorique

**Définition (Fonction d'autocovariance théorique)**

Soit  $(X_t)_{t \in T}$  un processus stationnaire.

La **fonction d'autocovariance théorique** de  $(X_t)_{t \in T}$  est la fonction

$$\begin{aligned} \forall t \in T \quad \gamma : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \text{Cov}(X_t, X_{t+|h|}) \end{aligned}$$

**Définition (Autocovariance théorique d'ordre  $h$ )**

Soit  $(X_t)_{t \in T}$  un processus stationnaire et  $h \in \mathbb{Z}$ .

L'**autocovariance théorique d'ordre  $h$** , notée  $\gamma(h)$ , de  $(X_t)_{t \in T}$  est l'image de  $h$  par la fonction d'autocovariance théorique de  $(X_t)_{t \in T}$ .

$$\forall t \in T \quad \gamma(h) = \text{Cov}(X_t, X_{t+|h|})$$

##### 6.1.1.1 Propriétés

La fonction d'autocovariance théorique de tout processus stationnaire est paire.

$$\forall h \in \mathbb{Z} \quad \gamma(-h) = \text{Cov}(X_t, X_{t+|-h|}) = \text{Cov}(X_t, X_{t+|h|}) = \gamma(h)$$

Soit  $(X_t)_{t \in T}$  un processus stationnaire.

$$\forall t \in T \quad \gamma(0) = \text{Cov}(X_t, X_t) = \text{Var}(X_t) \geq 0$$

$$\forall h \in \mathbb{Z} \quad |\gamma(h)| \leq \gamma(0)$$

Pour tout  $r \in \mathbb{N}^*$ , la matrice  $(\gamma(i-j))_{1 \leq i, j \leq r}$  est symétrique et semi-définie positive.

$$\begin{pmatrix} \gamma(0) & \gamma(-1) & \dots & \gamma(1-r) \\ \gamma(1) & \gamma(0) & \dots & \gamma(2-r) \\ \vdots & \vdots & \ddots & \vdots \\ \gamma(r-1) & \gamma(r-2) & \dots & \gamma(0) \end{pmatrix}$$

## 6.1.2 Autocorrélation théorique

Le calcul des acf ne peut pas s'appliquer à une série temporelle présentant des valeurs manquantes.  
La **mémoire** d'un processus est quantifiée par la corrélation qui existe entre les observations successives.

### Définition (Fonction d'autocorrélation théorique)

Soit  $(X_t)_{t \in T}$  un processus stationnaire.

La **fonction d'autocorrélation théorique** de  $(X_t)_{t \in T}$  est la fonction

$$\begin{aligned} \forall t \in T \quad \rho : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \frac{\gamma(h)}{\gamma(0)} = \frac{\text{Cov}(X_t, X_{t+|h|})}{\text{Var}(X_t)} \end{aligned}$$

La fonction d'autocorrélation théorique de tout processus stationnaire est paire.

### Définition (Autocorrélation théorique d'ordre $h$ )

Soit  $(X_t)_{t \in T}$  un processus stationnaire et  $h \in \mathbb{Z}$ .

L'**autocorrélation théorique d'ordre  $h$** , notée  $\rho(h)$ , de  $(X_t)_{t \in T}$  est l'image de  $h$  par la fonction d'autocorrélation théorique de  $(X_t)_{t \in T}$ .

$$\forall t \in T \quad \rho(h) = \frac{\gamma(h)}{\gamma(0)} = \frac{\text{Cov}(X_t, X_{t+|h|})}{\text{Var}(X_t)}$$

Soit  $(X_t)_{t \in T}$  un processus stationnaire.

$$\forall t \in T \quad \rho(0) = \frac{\gamma(0)}{\gamma(0)} = 1$$

### 6.1.2.1 Autocorrélation partielle théorique

Il arrive que deux variables soient corrélées sans qu'il n'existe de lien de cause à effet, car c'est une variable externe qui crée un lien artificiel entre elles. L'autocorrélation partielle permet de mesurer le lien entre deux variables, lorsqu'on neutralise l'effet de la variable externe.

### Définition (Coefficient de corrélation partielle théorique)

Soit  $X, Y$  et  $Z$  trois variables aléatoires.

Le **coefficient de corrélation partielle théorique** entre  $Y$  et  $Z$  à  $X$  connu, noté  $\text{Cor}_p(Y, Z \mid X)$ , est

$$\text{Cor}_p(Y, Z \mid X) = \frac{\text{Cor}(Y, Z) - \text{Cor}(X, Y)\text{Cor}(X, Z)}{\sqrt{(1 - \text{Cor}(X, Y)^2)(1 - \text{Cor}(X, Z)^2)}}$$

La **mémoire partielle** d'un processus est la mémoire du processus lorsqu'on neutralise l'effet des observations intermédiaires.

### Définition (Fonction d'autocorrélation partielle théorique)

Soit  $(X_n)_{n \in \mathbb{N}}$  un processus stationnaire.

La **fonction d'autocorrélation partielle théorique** de  $(X_t)_{t \in T}$ , notée  $\rho_p$ , est

$$\begin{aligned} \rho_p : \mathbb{N} &\longrightarrow \mathbb{R} \\ h &\longmapsto \begin{cases} 1 & \text{si } h = 0 \\ \text{Cor}_p(X_0, X_h \mid X_1, \dots, X_{h-1}) & \text{si } h > 0 \end{cases} \end{aligned}$$

**Définition (Fonction d'autocorrélation partielle théorique d'ordre  $h$ )**

Soit  $(X_n)_{n \in \mathbb{N}}$  un processus stationnaire et  $h \in \mathbb{N}$ .

La **fonction d'autocorrélation partielle théorique d'ordre  $h$**  de  $(X_t)_{t \in T}$ , notée  $\rho_p(h)$ , est l'image de  $h$  par la fonction d'autocorrélation partielle théorique de  $(X_t)_{t \in T}$ .

$$\rho_p(h) = \begin{cases} 1 & \text{si } h = 0 \\ \text{Cor}_p(X_0, X_h \mid X_1, \dots, X_{h-1}) & \text{si } h > 0 \end{cases}$$

**6.1.2.1.1 Propriétés**

Soit  $(X_n)_{n \in \mathbb{N}}$  un processus stationnaire.

$$\rho_p(1) = \text{Cor}_p(X_0, X_1 \mid X_1, \dots, X_{-1}) = \text{Cor}(X_0, X_1) = \frac{\text{Cov}(X_0, X_1)}{\sigma_{X_0} \sigma_{X_1}} = \frac{\text{Cov}(X_0, X_1)}{\text{Var}(X_0)} = \frac{\gamma(1)}{\gamma(0)} = \rho(1)$$

A mettre dans la définition de la fonction d'autocorrélation partielle théorique cas  $h = 1$  non ?

$$\rho_p(2) = \text{Cor}_p(X_0, X_2 \mid X_1) = \frac{\text{Cor}(X_0, X_2) - \text{Cor}(X_1, X_0)\text{Cor}(X_1, X_2)}{\sqrt{(1 - \text{Cor}(X_1, X_0)^2)(1 - \text{Cor}(X_1, X_2)^2)}} = \frac{\rho(2) - \rho(1)^2}{1 - \rho(1)^2}$$

**6.2 Fonctions empiriques****6.2.1 Autocovariance empirique****Définition (Fonction d'autocovariance empirique)**

Soit  $(X_t)_{t \in T}$  un processus stationnaire.

La **fonction d'autocovariance empirique** de  $(X_t)_{t \in T}$  est la fonction

$$\begin{aligned} \forall n \in \mathbb{N}^* \quad \hat{\gamma} : \{1 - n, \dots, n - 1\} &\longrightarrow \mathbb{R} \\ h &\longmapsto \frac{1}{n} \sum_{k=1}^{n-|h|} (X_{k+|h|} - \overline{X_n})(X_k - \overline{X_n}) \end{aligned}$$

**Définition (Autocovariance empirique d'ordre  $h$ )**

Soit  $(X_t)_{t \in T}$  un processus stationnaire et  $h \in \mathbb{Z}$ .

L'**autocovariance empirique d'ordre  $h$** , notée  $\hat{\gamma}(h)$ , de  $(X_t)_{t \in T}$  est l'image de  $h$  par la fonction d'autocovariance empirique de  $(X_t)_{t \in T}$ .

$$\forall n \in \mathbb{N}^* \quad \hat{\gamma}(h) = \frac{1}{n} \sum_{k=1}^{n-|h|} (X_{k+|h|} - \overline{X_n})(X_k - \overline{X_n})$$

## 6.2.2 Autocorrélation empirique

### Définition (Fonction d'autocorrélation empirique)

Soit  $(X_t)_{t \in T}$  un processus stationnaire.

La **fonction d'autocorrélation empirique** de  $(X_t)_{t \in T}$  est la fonction

$$\begin{aligned} \forall n \in \mathbb{N}^* \quad \hat{\rho} : \{1 - n, \dots, n - 1\} &\longrightarrow \mathbb{R} \\ h &\longmapsto \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)} \end{aligned}$$

### Définition (Autocorrélation empirique d'ordre $h$ )

Soit  $(X_t)_{t \in T}$  un processus stationnaire et  $h \in \mathbb{Z}$ .

L'**autocorrélation empirique d'ordre  $h$** , notée  $\hat{\rho}(h)$ , de  $(X_t)_{t \in T}$  est l'image de  $h$  par la fonction d'autocorrélation empirique de  $(X_t)_{t \in T}$ .

$$\forall n \in \mathbb{N}^* \quad \hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}$$

# Chapitre 7

## Analyse spectrale

### 7.1 Densité spectrale

La densité spectrale permet de caractériser différents types de processus stationnaires.

**Définition (Densité spectrale)**

Soit  $(X_t)_{t \in T}$  un processus stationnaire.

La **densité spectrale** de  $(X_t)_{t \in T}$  est la fonction

$$\begin{aligned} f : [-\pi, \pi] &\longrightarrow \mathbb{R} \\ \lambda &\longmapsto \frac{1}{2\pi} \sum_{h \in \mathbb{Z}} \gamma(h) e^{-i\lambda h} \end{aligned}$$

La formule qui relie la densité spectrale à la fonction d'autocovariance théorique est inversible.

Soit  $(X_t)_{t \in T}$  un processus stationnaire.

$$\forall h \in \mathbb{Z} \quad \gamma(h) = \int_{-\pi}^{\pi} f(\lambda) e^{ih\lambda} d\lambda$$

#### 7.1.1 Propriétés

La densité spectrale de tout processus stationnaire est paire.

La densité spectrale de tout processus stationnaire est  $2\pi$ -périodique.

Si  $\sum_{h \in \mathbb{Z}} \gamma(h) e^{-i\lambda h} < +\infty$ , alors la densité spectrale est continue.

La densité spectrale permet de caractériser la nature du processus.

- La densité spectrale est identiquement nulle si, et seulement si, le processus est nul.
- La densité spectrale est constante et non nulle si, et seulement si, le processus est un bruit blanc faible.

## Chapitre 8

# Test de stationnarité

### 8.1 Test de Kwiatkowski-Phillips-Schmidt-Shin (KPSS)

$H_0$  : Processus stationnaire

# Chapitre 9

## Bruit blanc

### 9.1 Bruit blanc faible

**Définition (Bruit blanc (faible))**

Un **bruit blanc (faible)**, noté BB, est un processus stochastiques à accroissements non corrélés.

**Définition équivalente (Bruit blanc (faible))**

Un **bruit blanc (faible)**, noté BB, est un processus stochastique  $(\mathcal{E}_t)_{t \in T}$  tel que :

- $\exists \mu \in \mathbb{R}, \forall t \in T \quad \mathbb{E}[\mathcal{E}_t] = \mu$
- $\forall t \in T \quad \text{Var}(\mathcal{E}_t) = \sigma^2$
- $\forall (t_1, t_2) \in T^2 \quad t_1 \neq t_2 \implies \text{Cov}(\mathcal{E}_{t_1}, \mathcal{E}_{t_2}) = 0$   
ou encore (vérifier que  $t + h \neq t$  non ?) 1  
 $\forall (t, h) \in T^2 \quad t + h \in T \implies \text{Cov}(\mathcal{E}_t, \mathcal{E}_{t+h}) = 0$

Un **bruit blanc (faible)** est une suite de variables aléatoires non corrélées, homoscedastique et de moyenne constante. J'ai le droit d'écrire ça ?

Remarquons que la notion de bruit blanc (faible) ne se restreint pas à des variables indépendantes contrairement à celle de bruit blanc fort.

**Définition (Bruit blanc centré)**

Un **bruit blanc centré** est un bruit blanc dont les variables sont centrées.

Soit  $(\mathcal{E}_t)_{t \in T}$  un bruit blanc.

Le processus  $(\mathcal{E}_t)_{t \in T}$  est un bruit blanc centré si, et seulement si,

$$\forall t \in T \quad \mathbb{E}[\mathcal{E}_t] = 0$$

Soit  $(\varepsilon_n)_{n \in \mathbb{Z}}$  un bruit blanc (faible).

$$\forall h \in \mathbb{N}^* \quad \rho_p(h) = 0$$

Soit  $(\varepsilon_n)_{n \in \mathbb{Z}} \sim BB(0, \sigma^2)$ .

$$\forall \lambda \in [-\pi, \pi] \quad f(\lambda) = \frac{1}{2\pi} \sigma^2$$

### 9.2 Bruit blanc fort

**Définition (Bruit blanc fort)**

Un **bruit blanc fort** est un bruit blanc (faible) dont les accroissements sont indépendants.

Un bruit blanc fort, noté IID, est un processus discret formé de variables mutuellement indépendantes et identiquement distribuées.

Un bruit blanc fort est une suite de variables aléatoires homoscédastiques et indépendantes.

Tout bruit blanc fort est un bruit blanc faible. Tout bruit blanc fort est stationnaire.

Soit  $(\varepsilon_n)_{n \in \mathbb{N}} \sim IID(0, 1)$ . La série  $(\varepsilon_n)_{n \in \mathbb{N}}$  est stationnaire. On a

$$\forall n \in \mathbb{N} \quad \mathbb{E}(\varepsilon_n) = 0$$

Des variables indépendantes constituent un processus à de mémoire nulle.

$$\gamma(h) = \begin{cases} 1 & \text{si } h = 0 \\ 0 & \text{sinon} \end{cases}$$

Cela signifie que des prévisions produites pour une série modélisée par un bruit blanc fort ne dépendent pas des observations passées, mais simplement de la loi qui constitue le bruit blanc fort.

**Définition (Bruit blanc gaussien)**

Un **bruit blanc gaussien**, noté NID (normalement et identiquement distribué), est un bruit blanc dont la loi de probabilité est normale.

Tout bruit blanc gaussien est un bruit blanc fort. Tout bruit blanc gaussien est stationnaire.

Tout processus stationnaire n'est pas un bruit blanc fort ou un bruit blanc gaussien.

**Définition (Processus à mémoire)**

Un **processus stochastique à mémoire** est un processus stationnaire qui n'est pas un bruit blanc fort ni un bruit blanc gaussien *i.e.* qu'il existe une loi de reproduction interne au processus qui est donc modélisable.

La mémoire d'un processus est quantifié par la corrélation qui existe entre les observations successives.

Toute suite de variables aléatoires indépendantes constitue un processus de mémoire nulle.

Processus à mémoire nulle.

## 9.3 Test de bruit blanc ou test de blancheur

Construtions des tests statistiques d'identification de bruits blancs par analyse des fonctions d'autocorrélation.

Lorsque nous étudions la fonction d'autocorrélation d'une série temporelle, la question qui se pose est de savoir quels sont les termes qui sont significativement différents de 0.

- Si aucun terme n'est significativement différent de 0, alors on peut en conclure que le processus est sans mémoire et donc qu'il n'est affecté ni de tendance ni de saisonnalité.
- S'il existe un terme présentant une valeur élevée, alors la série est certainement affectée d'un mouvement saisonnier.

### 9.3.1 Test d'un coefficient d'autocorrélation empirique d'ordre fixé à la fois

#### 9.3.1.1 Principe

Soit  $h \in \mathbb{N}^*$  et  $(X_t)_{t \in T}$  un processus stochastique. Définissons les hypothèses suivantes.

**Hypothèse nulle**  $H_0$  : « Le processus  $(X_t)_{t \in T}$  est un bruit blanc ou Le coefficient d'autocorrélation théorique d'ordre  $h$  est nul. »

$$\rho(h) = 0$$

**Hypothèse alternative**  $H_1$  : « Le processus  $(X_t)_{t \in T}$  n'est pas un bruit blanc ou Le coefficient d'autocorrélation théorique d'ordre  $h$  est non nul. »

$$\rho(h) \neq 0$$

Par théorème, si  $(X_t)_{t \in T}$  est un bruit blanc tel que, pour tout  $t \in T$ ,  $\mathbb{E}[X_t^4] < +\infty$  alors, pour tout  $h \in \mathbb{N}^*$ , la variable d'autocorrélation empirique d'ordre  $h$  converge en loi vers une loi normale de moyenne 0 et d'écart type  $\frac{1}{\sqrt{n}}$ .

$$\hat{\Xi}(h) \xrightarrow[n \rightarrow +\infty]{\mathcal{L}} \mathcal{N}\left(0, \frac{1}{\sqrt{n}}\right)$$

**Notation** Notons  $\alpha$  le seuil de tolérance ou indice de confiance et  $u_\alpha$  le quantile tel que  $\mathbb{P}(|U| > u_\alpha) = \alpha$  pour la loi gaussienne standard  $U$ . On utilise usuellement  $\alpha = 0,05$ .

Si  $n$  est assez grand, alors, pour  $(1 - \alpha)\%$  des réalisations de ce bruit blanc, n'importe quelle fonction d'autocorrélation empirique d'ordre non nul est comprise dans l'intervalle  $\left[-\frac{u_\alpha}{\sqrt{n}}, \frac{u_\alpha}{\sqrt{n}}\right]$ .

- Si  $\hat{\rho}(h) \in \left[-\frac{u_\alpha}{\sqrt{n}}, \frac{u_\alpha}{\sqrt{n}}\right]$ , alors on considère que l'autocorrélation théorique  $\rho(h)$  est nulle et donc que la série observée est issue d'un bruit blanc.
- Si  $\hat{\rho}(h) \notin \left[-\frac{u_\alpha}{\sqrt{n}}, \frac{u_\alpha}{\sqrt{n}}\right]$ , alors on considère que l'autocorrélation théorique  $\rho(h)$  est non nulle et donc que la série observée n'est pas issue d'un bruit blanc.

### 9.3.1.2 Implémentation

La fonction `acf()` trace le corrélogramme ainsi que l'intervalle  $\left[-\frac{u_\alpha}{\sqrt{n}}, \frac{u_\alpha}{\sqrt{n}}\right]$  en pointillé bleu.

- Si la fonction d'autocorrélation empirique d'ordre  $h$  est à l'intérieur de l'intervalle au niveau de confiance  $\alpha$ , alors on considère que l'autocorrélation théorique  $\rho(h)$  est nulle et donc que la série observée est issue d'un bruit blanc.
- Si la fonction d'autocorrélation empirique d'ordre  $h$  est à l'extérieur de l'intervalle au niveau de confiance  $\alpha$ , alors on considère que l'autocorrélation théorique  $\rho(h)$  est non nulle et donc que la série observée n'est pas issue d'un bruit blanc.

### 9.3.1.3 Limites

Mais on peut réaliser ce test pour chacun des ordres  $h$ , et les réponses ne sont pas forcément compatibles. Nous devons souligner une limite des tests à 5%. En effet, lorsqu'une fonction d'autocorrélation est calculée pour un nombre important de retards, nous pouvons nous attendre à ce que quelques-uns soient, de manière fortuite, significativement différents de 0. Si  $r$  est le nombre de retards, le nombre possible de faux rejets est alors  $0,05 \times r$ , pour un seuil de confiance de 5%. Si l'une des autocorrélations empiriques se trouve en dehors de la bande, doit-on en conclure que la série n'est pas issue d'un bruit blanc ? *A priori* non, car le théorème indique que sous  $H_0$ , les variables d'autocorrélation empiriques convergent vers des variables indépendantes.

Soit  $k \in \mathbb{N}$ . La probabilité, sous  $H_0$ , d'observer au moins  $k + 1$  autocorrélations empiriques en dehors de la bande sur  $N$  valeurs tracées est la  $k + 1$ -ième coordonnée du vecteur obtenu par :

```
round(1-pbinom(0:4, prob=0.05, size=N), dig=4)
```

**Exemple** La probabilité d'observer au moins une autocorrélation empirique en dehors de la bande sur 20 tracés est alors égale à 0,642.

Si on rejetait  $H_0$  dès qu'au moins une des autocorrélations empiriques dépasse le seuil usuel, on se tromperait dans  $\alpha_{\text{global}} = (1 - (1 - \alpha)^N)\%$  des cas.

**Exemple** Reprenons les mêmes valeurs. On se tromperait dans 64% des cas.

- Si on observe  $k + 1$  autocorrélations empiriques en dehors de la bande et que la probabilité correspondante est inférieure à 0,05, alors il convient de rejeter  $H_0$ .
- Si on observe  $k + 1$  autocorrélations empiriques en dehors de la bande et que la probabilité correspondante est supérieure à 0,05, que doit-on en conclure ?

Cela dépend si les autocorrélations sont très e, dehors de l'intervalle ou pas. Le seuil  $\alpha$  associé au risque  $\alpha_{\text{global}}$  qui permette de rejeter  $H_0$  dès qu'au moins une des autocorrélations dépasse ce nouveau seuil (procédure de correction dans le cas de tests multiples) est

$$\alpha = 1 - (1 - \alpha_{\text{global}})^{\frac{1}{N}}$$

La fonction `acfG()` prend en compte ce nouveau seuil.

## 9.3.2 Test d'autocorrélation du Portemanteau

### 9.3.2.1 Principe

Étant donné l'indépendance asymptotiques des variables d'autocorrélations empiriques sous  $H_0$ , on peut considérer une statistique de test qui fait intervenir plusieurs autocorrélations empiriques à la fois. Il permet d'identifier les processus de marche au hasard.

Soit  $h \in \mathbb{N}^*$  et  $(X_t)_{t \in T}$  un processus stochastique. Définissons les hypothèses suivantes.

**Hypothèse nulle**  $H_0$  : « Il n'y a pas autocorrélation des erreurs d'ordre 1 à  $N$ . »

$$\forall i \in \{1, \dots, N\} \quad \rho_i = 0$$

**Hypothèse alternative**  $H_1$  : « Il y a autocorrélation des erreurs d'ordre 1 à  $N$ . »

$$\exists i \in \{1, \dots, N\} \quad \rho_i \neq 0$$

Pour effectuer ce test on recourt à l'une des deux statistiques ci-dessous suivant asymptotiquement, sous  $H_0$ , une loi  $\chi^2(m)$ .

### 9.3.2.2 Test de Box-Pierce (1970)

#### Définition (Statistique de Box-Pierce (1970))

Soit  $(X_t)_{t \in T}$  un processus stochastique et  $(h, m) \in \mathbb{N}^{*2}$ .

La **statistique de Box-Pierce à l'ordre  $m$** , notée  $Q_{BP}(m)$ , est

$$Q_{BP}(m) = n \sum_{h=1}^m \hat{\Xi}^2(h)$$

- Si la statistique  $Q_{BP}(m)$  est supérieure au  $\chi^2$  lu dans la table au seuil  $(1 - \alpha)$  et  $m$  degrés de libertés, alors nous rejetons l'hypothèse  $H_0$ .

### 9.3.2.3 Test de Ljung-Box (1978)

#### Définition (Statistique de Ljung-Box (1978))

Soit  $(X_t)_{t \in T}$  un processus stochastique et  $(h, m) \in \mathbb{N}^{*2}$ .

La **statistique de Ljung-Box à l'ordre  $m$** , notée  $Q_{LB}(m)$ , est

$$Q_{LB}(m) = n(n+2) \sum_{h=1}^m \frac{\hat{\Xi}^2(h)}{n-m}$$

- Si la statistique  $Q_{BP}(m)$  est supérieure au  $\chi^2$  lu dans la table au seuil  $(1 - \alpha)$  et  $m$  degrés de libertés, alors nous rejetons l'hypothèse  $H_0$ .

### 9.3.2.4 Implémentation

La fonction `Box.test()` réalise le test du Portemanteau. Le paramètre `type` permet de spécifier la statistique utilisée : `type="Box-Pierce"` ou `type="Ljung-Box"`.

- Si la p-valeur obtenue est supérieure à 0,05, alors on ne rejette pas l'hypothèse  $H_0$ .

Par défaut à l'ordre  $m = 1$ , ce qui revient à tester si  $\rho(1) = 0$  à partir de  $\hat{\rho}(1)$ . Pour utiliser au mieux l'information, on peut réaliser le test pour des ordres  $m \leq \frac{n}{4}$ , comme conseillé par Box-Jenkins.

Les propriétés asymptotiques de la statistique de Ljung Box sont meilleures que celle de Box Pierce.

### 9.3.3 Test de normalité (des résidus)

Il est intéressant de tester la normalité d'un bruit blanc. On pourra ainsi ajouter des intervalles de confiance autour des prévisions. Définissons les hypothèses suivantes.

**Hypothèse nulle**  $H_0$  : « Le processus  $(X_t)_{t \in T}$  est gaussien. »

**Hypothèse alternative**  $H_1$  : « Le processus  $(X_t)_{t \in T}$  n'est pas gaussien. »

#### 9.3.3.1 Tests basés sur les quantiles

##### 9.3.3.1.1 QQ-plot

Considérons le nuage de points formé par les quantiles théoriques de la loi  $\mathcal{N}(0, 1)$  et par les quantiles empiriques de la série. Sous  $H_0$ , c'est-à-dire si, pour tout  $t \in T$ ,  $X_t \simeq \mathcal{N}(0, 1)$ , alors le nuage est rectiligne et aligné sur la droite  $y = \sigma x + \mu$ . La normalité peut donc être acceptée.

La fonction `qnorm()` trace le nuage de points formé par les quantiles théoriques de la série.

##### 9.3.3.1.2 Test de Shapiro-Wilk

Ceci peut être conforté par le test de normalité Shapiro-Wilk dont la statistique de test est le coefficient de détermination de ce nuage.

La fonction `shapiro.test()` effectue ce test.

Il existe d'autres tests de normalité, basés sur d'autres propriétés de la loi normale.

#### 9.3.3.2 Tests basés sur les coefficient d'asymétrie (skewness) et d'aplatissement (kurtosis)

##### 9.3.3.2.1 Test de Jarque-Bera

La fonction `jarque.beta.test` du package `tseries` effectue le test de Jarque-Bera.

##### 9.3.3.2.2 Test d'Agostino

La fonction `jdagoTest()` du package `fBasics` effectue le test d'Agostino. La p-valeur du test d'Agostino se lit sur la ligne `Omnibus Test` et est accessible directement en tapant `dagoTest(serie)$test$p.value[1]`.

Sous  $H_0$ , les statistiques des tests de Jarque-Bera et d'Agostino suivent asymptotiquement une loi  $\chi^2(2)$ .

#### 9.3.3.3 Test de Lilliefors basé sur la fonction de répartition (inspiré du test de Kolmogov-Smirnov)

La fonction `lillie.test()` du package `nortest` effectue le test de Lilliefors.

### 9.3.4 Comparaison des tests de normalité

Certains auteurs ont montré que le test de Shapiro est plus puissant que les tests de Jarque-Bera et de Lilliefors. En outre, le test d'Agostino est plus puissant que le test de Jarque-Bera. Pour tester la normalité de données indépendantes, on utilisera donc en priorité les tests de Shapiro et d'Agostino.

## 9.4 Modélisation par un bruit blanc

Le bruit blanc est le modèle le plus élémentaire pour décrire une série. Mais du fait de sa mémoire nulle, ce modèle est assez peu informatif en termes de prévisions. En effet la valeur qui sera observée à l'instant suivant est non corrélée avec les observations passées. Les prévisions possibles sont donc les réalisations de la loi normale  $\mathcal{N}(0, v^2)$  où  $v^2$  est

- soit l'estimation par maximum de vraisemblance, et dans ce cas :
- soit l'estimation sans biais :

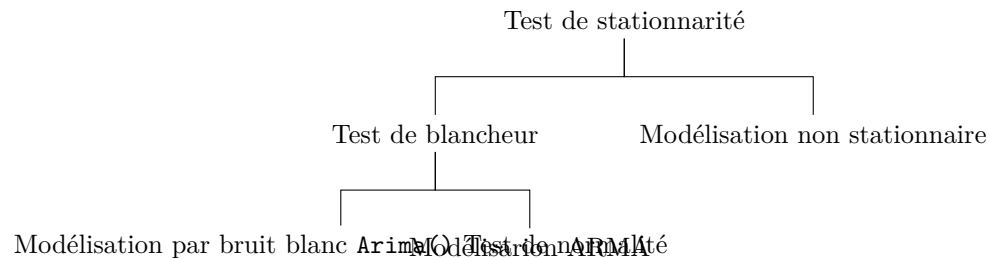
En général on indique comme valeur prédite la moyenne et l'intervalle de fluctuation qui correspond aux quantiles 2,5% et 97,5% de la loi gaussienne associée.

La fonction `Arima()` du package `forecast` estime le modèle du bruit blanc, calcule et trace les intervalles de prédiction avec une syntaxe très simple.

### 9.4.1 Méthode

(Test de stationnarité) Test de blancheur de la série.

- Si la série est issue d'un bruit blanc, alors utiliser la fonction `Arima()` et tester si le bruit blanc est gaussien.
- Si le bruit blanc est gaussien, alors tracer les intervalle de confiance par défaut.
- Sinon, calculer les quantiles obtenus lors de 5000 simulations de trajectoires futures, en tirant au hasard dans les résidus observés.
- Sinon, utiliser un modèle ARMA.



## 9.5 Exemple de modélisation par un bruit blanc

Considérons le jeu de données `airquality` du logiciel **R**.

```
data(airquality) # class(airquality) "data.frame"
```

Avant d'afficher les données il convient de prendre en compte les dimensions du jeu de données.

```
dim(airquality)
## [1] 153  6
```

L'affichage de l'ensemble des données obtenu avec la commande suivante nécessite plusieurs pages.

```
airquality
```

Affichons donc uniquement les 6 premières et 6 dernières observations.

```
head(airquality)

##   Ozone Solar.R Wind Temp Month Day
## 1    41    190  7.4   67     5   1
## 2    36    118  8.0   72     5   2
## 3    12    149 12.6   74     5   3
## 4    18    313 11.5   62     5   4
## 5     NA     NA 14.3   56     5   5
## 6    28     NA 14.9   66     5   6

tail(airquality)

##   Ozone Solar.R Wind Temp Month Day
## 148    14     20 16.6   63     9  25
## 149    30    193  6.9   70     9  26
## 150    NA    145 13.2   77     9  27
## 151    14    191 14.3   75     9  28
## 152    18    131  8.0   76     9  29
## 153    20    223 11.5   68     9  30
```

Extrayons la série temporelle `Solar.R`. Cette série temporelle quotidienne représente l'intensité du rayonnement mesurée à New-York entre mai 1973 et septembre 1973. Affichons uniquement les 6 premières et 6 dernières observations.

```
Soleil <- airquality$Solar.R # class "integer"
## Autres méthodes, indicées
# Soleil <- airquality[,2] # class "integer"
# Soleil <- airquality[[2]] # class "integer"
## Autres méthodes, par nom
# Soleil <- airquality["Solar.R"] # class "integer"
# Soleil <- airquality[["Solar.R"]] # class "integer"
head(Soleil) # class "integer"

## [1] 190 118 149 313  NA  NA

tail(Soleil) # class "integer"

## [1]  20 193 145 191 131 223
```

Remarquez la différence avec les commandes suivantes.

```

Soleil_df <- airquality[2] # class "data.frame"
## Autre méthode, par nom
# Soleil_df <- airquality["Solar.R"] # class "data.frame"
head(Soleil_df) # class "data.frame"

##      Solar.R
## 1         190
## 2         118
## 3         149
## 4         313
## 5          NA
## 6          NA

tail(Soleil_df) # class "data.frame"

##      Solar.R
## 148         20
## 149        193
## 150        145
## 151        191
## 152        131
## 153        223

```

Cette série présente 7 valeurs manquantes.

```

which(is.na(Soleil))

## [1]  5  6 11 27 96 97 98

```

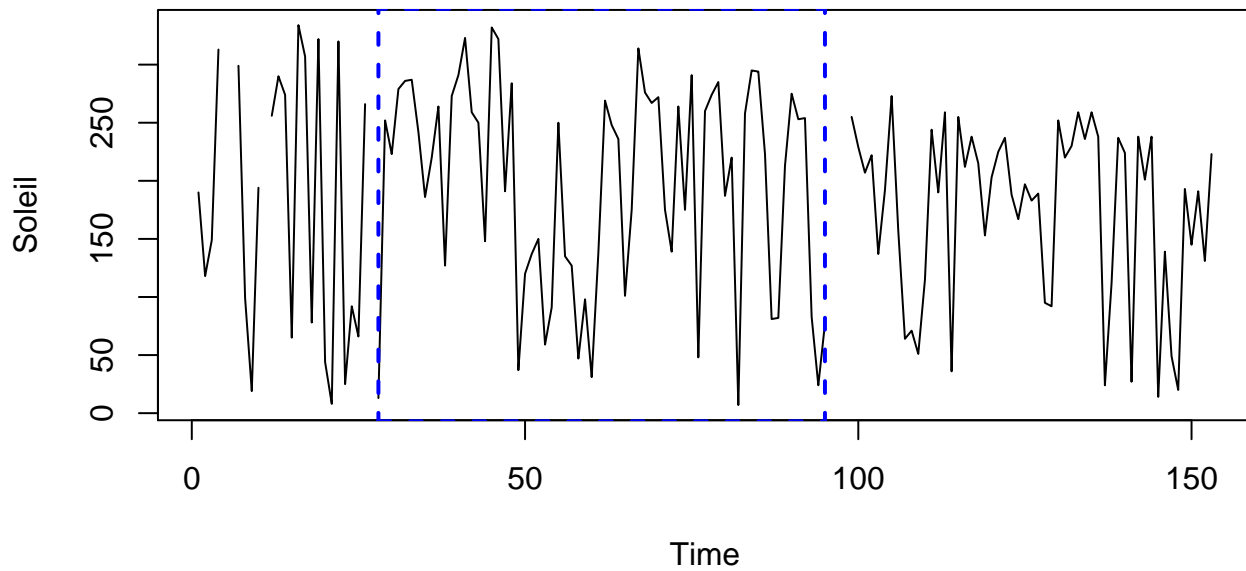
Traçons le chronogramme de la série Soleil et celui de la série tronquée Soleil[28:95].

```

ts.plot(Soleil, main="Chronogramme de la série Soleil")
#abline(v=c(28,95), col="blue", lwd=2, lty=2)
polygon(c(28, 95, 95, 28), c(-7,-7,348,348), border="blue", lwd=2, lty=2) # package graphics
ts.plot(Soleil[28:95], main="Chronogramme de la série Soleil[28:95]")

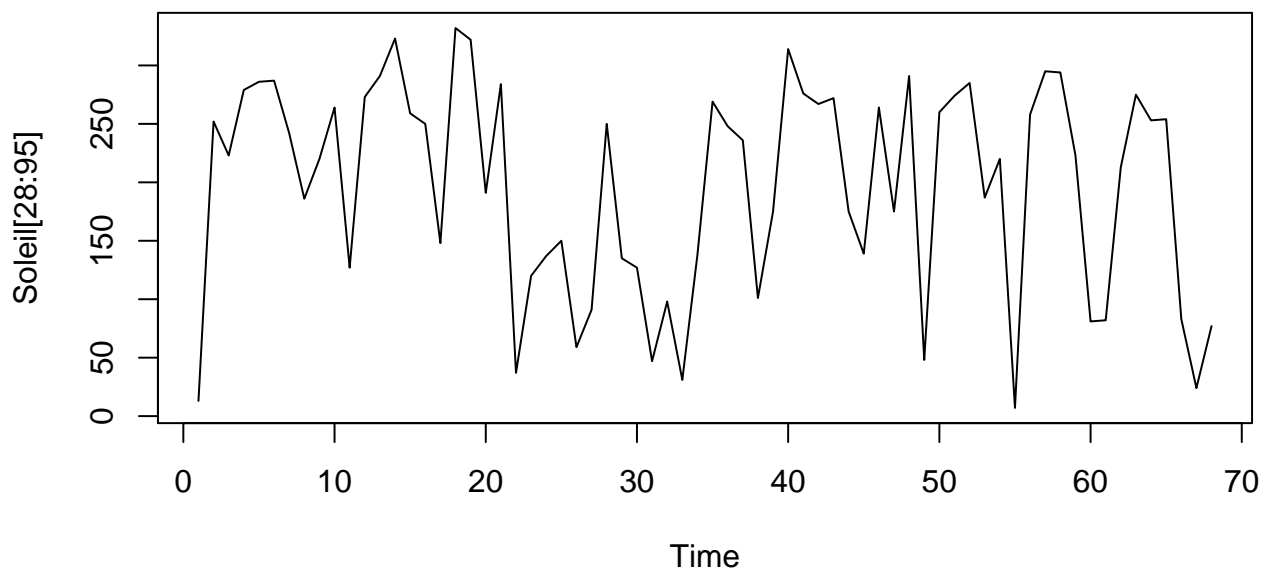
```

### Chronogramme de la série Soleil



(a) Chronogrammes de la série `Soleil`

### Chronogramme de la série `Soleil[28:95]`



(b) Chronogrammes de la série `Soleil[28:95]`

FIGURE 9.1 – Chronogrammes des séries `Soleil` et `Soleil[28:95]`

Réduisons la série `Soleil` à la sous-série des termes d'indice 28 à 95.

```
soleil <- airquality$Solar.R[28:95] # class "integer"
# ou soleil <- Soleil[28:95]
```

Cette série ne présente aucune valeur manquante et contient 68 observations.

```
which(is.na(soleil))
## integer(0)
length(soleil)
## [1] 68
```

Posons  $T := \{1, \dots, 68\}$ . Notons que, pour tout  $t \in T$ , `soleil[t] = Soleil[t+27]`. Posons, pour tout  $t \in T$ ,  $X_t := \text{soleil}[t]$ . On cherche à modéliser la série  $(X_t)_{t \in T}$ .

### 9.5.1 Test de stationnarité

```
library(tseries)

## Registered S3 method overwritten by 'quantmod':
## method      from
## as.zoo.data.frame zoo

kpss.test(soleil) # class "htest"

## Warning in kpss.test(soleil): p-value greater than printed p-value

##
## KPSS Test for Level Stationarity
##
## data:  soleil
## KPSS Level = 0.1596, Truncation lag parameter = 3, p-value = 0.1

# Extraction de la p-valeur
names(kpss.test(soleil))

## Warning in kpss.test(soleil): p-value greater than printed p-value

## [1] "statistic" "parameter" "p.value" "method" "data.name"

kpss.test(soleil)$p.value

## Warning in kpss.test(soleil): p-value greater than printed p-value

## [1] 0.1
```

On a `kpss.test(soleil)$p.value = 0.1 > 0.05` donc la série `soleil` est stationnaire.

### 9.5.2 Test de blancheur

#### 9.5.2.1 Test d'autocorrélation du Portemanteau

```
Box.test(soleil, lag=round(length(soleil)/4)) # class "htest"

##
```

```
## Box-Pierce test
##
## data:  soleil
## X-squared = 26.517, df = 17, p-value = 0.06554

# Extraction de la p-valeur
names(Box.test(soleil, lag=round(length(soleil)/4)))

## [1] "statistic" "parameter" "p.value" "method" "data.name"

Box.test(soleil, lag=round(length(soleil)/4))$p.value

## [1] 0.06553876
```

On a  $\text{Box.test(soleil, lag=round(length(soleil)/4))\$p.value} = 0.0655 > 0.05$  donc la série `soleil` est modélisable par un bruit blanc.

### 9.5.2.2 Test d'un coefficient d'autocorrélation empirique d'ordre fixé à la fois

```
acfG(soleil)
```

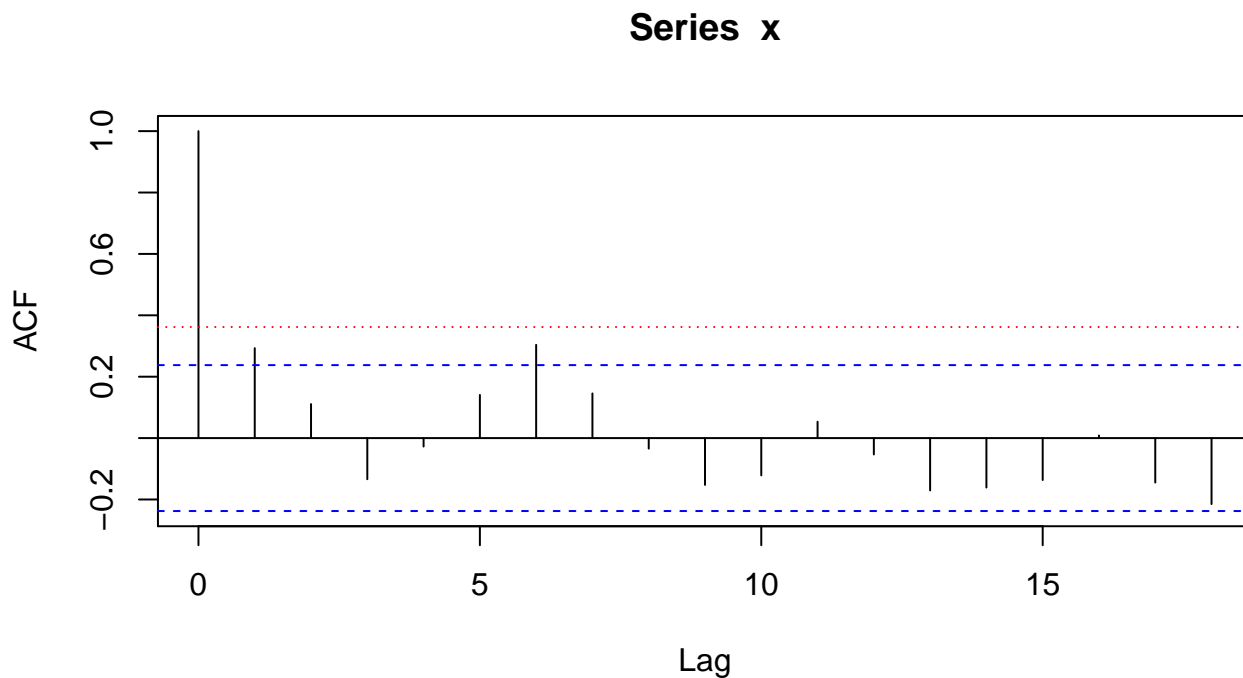


FIGURE 9.2 – Autocorrélogramme empirique de la série `soleil`

```
##
## One-sample Kolmogorov-Smirnov test
##
## data:  sqrt(n) * rho
## D = 0.28637, p-value = 0.08441
## alternative hypothesis: two-sided
```

Dans la Figure 9.2 les autocorrélations empiriques d'ordre 1,  $\hat{\rho}(1)$ , et d'ordre 6,  $\hat{\rho}(6)$  ne se trouvent pas à l'intérieur de la bande définie par les seuils classiques.

### 9.5.3 Modèle

```
library(forecast) # fonction Arima()
modele <- Arima(soleil)
```

### 9.5.4 Test de normalité du bruit blanc

#### 9.5.4.1 Test de Shapiro-Wilk

```
shapiro.test(soleil) # class "htest"

##
##  Shapiro-Wilk normality test
##
## data:  soleil
## W = 0.91826, p-value = 0.0002734

# Extraction de la p-valeur
names(shapiro.test(soleil))

## [1] "statistic" "p.value"    "method"     "data.name"

shapiro.test(soleil)$p.value

## [1] 0.0002733515
```

On a  $\text{shapiro.test(soleil)}\$p.value = 3 \times 10^{-4} < 0,05$  donc le bruit blanc n'est pas gaussien.

#### 9.5.4.2 Test d'Agostino

```
library(fBasics) # fonction dagoTest()

## Loading required package: timeDate
## Loading required package: timeSeries

dagoTest(soleil) # class "fHTEST", "attr(,"package)") "fBasics"

##
## Title:
##  D'Agostino Normality Test
##
## Test Results:
##  STATISTIC:
##    Chi2 | Omnibus: 11.1853
##    Z3   | Skewness: -1.8462
##    Z4   | Kurtosis: -2.7887
##  P VALUE:
##    Omnibus Test: 0.003725
##    Skewness Test: 0.06486
##    Kurtosis Test: 0.005292
##
## Description:
##  Sun May 8 12:33:34 2022 by user:
```

```

# Extraction de la p-valeur
dagoTest(soleil)@test

## $statistic
## Chi2 | Omnibus Z3 | Skewness Z4 | Kurtosis
##      11.185262      -1.846231      -2.788672
##
## $method
## [1] "D'Agostino Omnibus Normality Test"
##
## $p.value
## Omnibus Test Skewness Test Kurtosis Test
## 0.003725213 0.064858621 0.005292461
##
## $data.name
## [1] "soleil"

dagoTest(soleil)@test$p.value[1]

## Omnibus Test
## 0.003725213

```

On a `dagoTest(soleil)@test$p.value[1] = 0.0037 < 0,05` donc le bruit blanc n'est pas gaussien.

### 9.5.5 Prévisions

Par défaut le package `forecast` suppose les bruits blancs gaussiens. Il ne faut donc pas tracer les intervalles de confiance calculés par défaut, en supposant les résidus gaussiens. L'argument `bootstrap=T` permet de calculer les quantiles obtenus lors de 5000 simulations de trajectoires futures, en tirant au hasard dans les résidus observés.

```

nbP <- 58
prev <- forecast(modele, h=nbP, bootstrap=T)
plot(prev)

```

### Forecasts from ARIMA(0,0,0) with non-zero mean

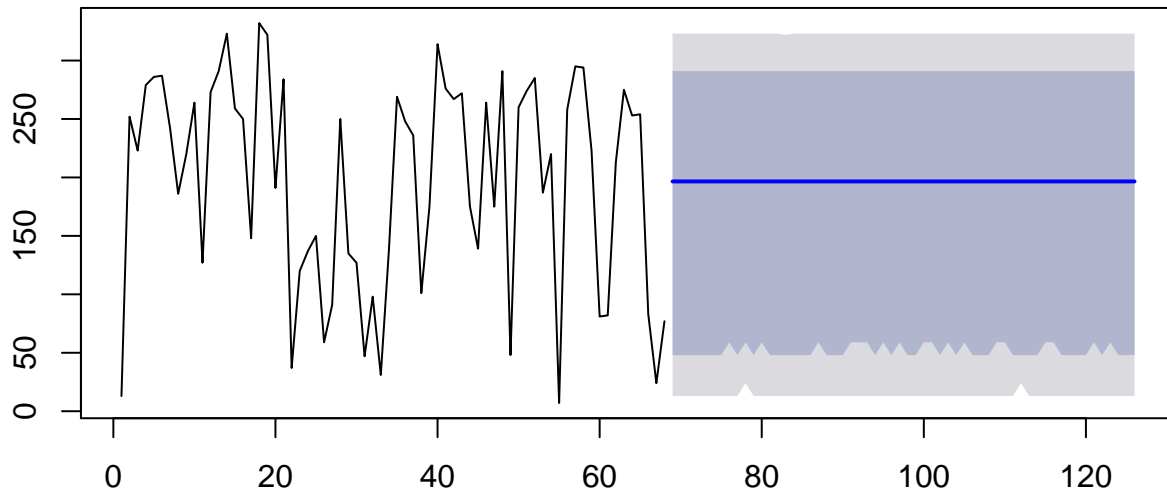


FIGURE 9.3 – Prévisions pour la série `soleil`

Lorsqu'on transforme une série pour la rendre stationnaire, le processus obtenu est rarement un bruit blanc. On doit faire appel à la classe des processus ARMA, plus vaste.

## Chapitre 10

# Modèles pour l'homoscédasticité : Modèle autorégressif et moyenne mobile ARMA

Les modèles classiques de prévision fondés sur les modèles ARMA supposent des séries temporelles à variance constante (hypothèse d'homoscédasticité). Cette modélisation néglige donc, éventuellement, l'information contenue dans le facteur résiduel de la chronique.

### 10.1 Théorème de décomposition de Wold

Le théorème de décomposition de Wold permet de décomposer tout processus (faiblement) stationnaire en la somme d'une partie déterministe et d'une partie stochastique.

**Définition (Processus autorégressif moyenne mobile d'ordres  $p$  et  $q$ )**

Soit  $(p, q) \in \mathbb{N}^2$ ,  $(\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q) \in \mathbb{R}^{p+q}$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible).

Un processus **autorégressif moyenne mobile** (en anglais, *Auto-Regressive Moving Average*) d'ordres  $p$  et  $q$ , noté  $\text{ARMA}(p, q)$ , de paramètres  $\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q$  et d'erreur  $(\varepsilon_t)_{t \in T}$  est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe  $c \in \mathbb{R}$  vérifiant

$$\forall t \in T \quad X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

**Convention** Soit  $(p, q) \in \mathbb{N}^2$  et  $(X_t)_{t \in T}$  un  $\text{ARMA}(p, q)$  de paramètres  $\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q$  et d'erreur  $(\varepsilon_t)_{t \in T}$ . Par convention on pose  $\theta_0 = 1$  de sorte à obtenir l'expression simplifiée suivante.

$$\forall t \in T \quad X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \sum_{i=0}^q \theta_i \varepsilon_{t-i}$$

Intuitivement, la valeur actuelle d'un processus ARMA peut s'écrire comme la somme de deux composantes :

- la première est une combinaison linéaire du passé du processus lui même,
- la deuxième composante est une combinaison linéaire d'un bruit blanc.

**Définition (Processus autorégressif moyenne mobile centré d'ordres  $p$  et  $q$ )**

Soit  $(p, q) \in \mathbb{N}^2$ ,  $(\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q) \in \mathbb{R}^{p+q}$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible).

Un processus **autorégressif moyenne mobile centré** d'ordres  $p$  et  $q$ , de paramètres  $\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q$  et d'erreur  $(\varepsilon_t)_{t \in T}$  est un processus stochastique  $(X_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t = \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

CNS de stationnarité, processus linéaire, CNS causalité et d'inversibilité, consistance (convergence des estimateurs empiriques)

## 10.2 Processus autorégressifs AR

### Définition (Processus autorégressif d'ordre $p$ )

Soit  $p \in \mathbb{N}^*$ ,  $(\varphi_1, \dots, \varphi_p) \in \mathbb{R}^{p-1} \times \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible).

Un processus **autorégressif** (en anglais, *Autoregressive*) d'ordre  $p$ , noté  $\text{AR}(p)$ , de paramètres  $\varphi_1, \dots, \varphi_p$  et d'erreur  $(\varepsilon_t)_{t \in T}$  est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe  $c \in \mathbb{R}$  vérifiant

$$\forall t \in T \quad X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t$$

### Définition (Processus autorégressif centré d'ordre $p$ )

Soit  $p \in \mathbb{N}^*$ ,  $(\varphi_1, \dots, \varphi_p) \in \mathbb{R}^{p-1} \times \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible).

Le processus **autorégressif centré** d'ordre  $p$ , de paramètres  $\varphi_1, \dots, \varphi_p$  et d'erreur  $(\varepsilon_t)_{t \in T}$  est le processus stochastique  $(X_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t = \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t$$

Soit  $p \in \mathbb{N}^*$ ,  $(\varphi_1, \dots, \varphi_p) \in \mathbb{R}^{p-1} \times \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible) centré. Notons  $(X_t)_{t \in T}$  le processus  $\text{AR}(p)$  centré de paramètres  $\varphi_1, \dots, \varphi_p$  et d'erreur  $(\varepsilon_t)_{t \in T}$  i.e. défini par

$$\forall t \in T \quad X_t = \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t$$

Introduisons l'opérateur retard  $L$ .

$$\forall t \in T \quad \varepsilon_t = X_t - \sum_{i=1}^p \varphi_i X_{t-i} = X_t - \sum_{i=1}^p \varphi_i L^i X_t = \left( \mathbf{1} - \sum_{i=1}^p \varphi_i L^i \right) X_t$$

Posons  $\Phi_p(L) := \mathbf{1} - \sum_{i=1}^p \varphi_i L^i$ . On a

$$\forall t \in T \quad \varepsilon_t = \Phi_p(L) X_t$$

Le polynôme  $\Phi_p(L)$  est inversible donc il existe  $(\psi_n)_{n \in \mathbb{N}} \in \mathbb{R}^{\mathbb{N}}$  tel que  $\Phi_p(L)^{-1} = \sum_{k=0}^{+\infty} \psi_k L^k$ . Finalement,

$$\forall t \in T \quad X_t = \Phi_p(L)^{-1} \varepsilon_t = \sum_{k=0}^{+\infty} \psi_k L^k \varepsilon_t = \sum_{k=0}^{+\infty} \psi_k \varepsilon_{t-k}$$

Le processus  $(X_t)_{t \in T}$  est donc un processus  $\text{MA}(\infty)$  centré.

### Proposition ( $\text{AR}(p)$ et $\text{MA}(\infty)$ )

Soit  $p \in \mathbb{N}^*$ .

Tout processus  $\text{AR}(p)$  est un processus  $\text{MA}(\infty)$ .

### 10.2.1 Espérance

$$\forall t \in T \quad \mathbb{E}(X_t) = \mathbb{E} \left( \varepsilon_t + \sum_{i=0}^p \varphi_i \varepsilon_{t-i} \right) = \mathbb{E}(\varepsilon_t) + \sum_{i=0}^p \varphi_i \mathbb{E}(\varepsilon_{t-i}) = 0$$

Remarquons que,

$$\forall (t, h) \in T \times \mathbb{N}^* \quad \mathbb{E}(\varepsilon_{t+h} X_t) = \mathbb{E} \left( \varepsilon_{t+h} \sum_{k=0}^{+\infty} \psi_k \varepsilon_{t-k} \right) = \mathbb{E} \left( \sum_{k=0}^{+\infty} \psi_k \varepsilon_{t+h} \varepsilon_{t-k} \right) = \sum_{k=0}^{+\infty} \psi_k \mathbb{E}(\varepsilon_{t+h} \varepsilon_{t-k}) = 0$$

## 10.2.2 Fonction d'autocovariance et d'autocorrélation théoriques

### 10.2.2.1 Par MA infini

Le processus  $(X_t)_{t \in T}$  est un processus MA( $\infty$ ) centré donc la fonction d'autocovariance de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \gamma : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \sigma^2 \sum_{k=0}^{+\infty} \psi_k \psi_{k+|h|} \end{aligned}$$

La fonction d'autocorrélation théorique de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \rho : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \frac{\sum_{i=0}^{+\infty} \psi_i \psi_{i+|h|}}{\sum_{j=0}^{+\infty} \psi_j^2} \end{aligned}$$

La mémoire ou fonction d'autocorrélation théorique de tout processus AR( $p$ ) ne dépend pas de la variance de son erreur.

### 10.2.2.2 Par récurrence

$$\begin{aligned} \forall (t, h) \in T \times \mathbb{N}^* \quad \gamma(h) &= \text{Cov}(X_t, X_{t+h}) \\ &= \mathbb{E}(X_{t+h} X_t) - \mathbb{E}(X_t) \mathbb{E}(X_{t+h}) \\ &= \mathbb{E}(X_{t+h} X_t) \\ &= \mathbb{E} \left( \left( \varepsilon_{t+h} + \sum_{i=1}^p \varphi_i X_{t+h-i} \right) X_t \right) \\ &= \mathbb{E}(\varepsilon_{t+h} X_t) + \sum_{i=1}^p \varphi_i \mathbb{E}(X_{t+h-i} X_t) \\ &= \sum_{i=1}^p \varphi_i (\text{Cov}(X_t, X_{t+h-i}) + \mathbb{E}(X_t) \mathbb{E}(X_{t+h-i})) \\ &= \sum_{i=1}^p \varphi_i \text{Cov}(X_t, X_{t+h-i}) \\ &= \sum_{i=1}^p \varphi_i \gamma(h-i) \end{aligned}$$

La fonction d'autocovariance théorique de  $(X_t)_{t \in T}$  vérifie une équation analogue à celle vérifiée par le processus  $(X_t)_{t \in T}$  dans laquelle on supprime le bruit.

## 10.2.3 Processus autorégressif d'ordre 1 AR(1)

Soit  $\varphi_1 \in \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible) centré. Notons  $(X_t)_{t \in T}$  le processus AR(1) centré de paramètre  $\varphi_1$  et d'erreur  $(\varepsilon_t)_{t \in T}$  i.e. défini par

$$\forall t \in T \quad X_t = \varphi_1 X_{t-1} + \varepsilon_t$$

Supposons que  $|\varphi_1| < 1$ .

- On a

$$\begin{aligned}
\forall t \in T \quad X_t &= \varepsilon_t + \varphi_1 X_{t-1} \\
&= \varepsilon_t + \varphi_1(\varepsilon_{t-1} + \varphi_1 X_{t-2}) = \varepsilon_t + \varphi_1 \varepsilon_{t-1} + \varphi_1^2 X_{t-2} \\
&= \varepsilon_t + \varphi_1 \varepsilon_{t-1} + \varphi_1^2(\varepsilon_{t-2} + \varphi_1 X_{t-3}) \\
&= \varepsilon_t + \varphi_1 \varepsilon_{t-1} + \varphi_1^2 X_{t-2} + \varphi_1^3 X_{t-3} \\
&\vdots \\
&= \sum_{i=0}^{+\infty} \varphi_1^i \varepsilon_{t-i}
\end{aligned}$$

- Reformulons ce raisonnement en introduisant l'opérateur retard. On a

$$\forall t \in T \quad \varepsilon_t = X_t - \varphi_1 X_{t-1} = X_t - \varphi_1 L X_t = (\mathbf{1} - \varphi_1 L) X_t$$

On a  $|\varphi_1| < 1$  donc  $\mathbf{1} - \varphi_1 L$  est inversible. On en déduit que

$$\forall t \in T \quad X_t = (\mathbf{1} - \varphi_1 L)^{-1} \varepsilon_t = \left( \sum_{i=0}^{+\infty} \varphi_1^i L^i \right) \varepsilon_t = \sum_{i=0}^{+\infty} \varphi_1^i L^i \varepsilon_t = \sum_{i=0}^{+\infty} \varphi_1^i \varepsilon_{t-i}$$

Le processus  $(X_t)_{t \in T}$  est un processus MA( $\infty$ ) centré.

**Corollaire (AR(1) et MA( $\infty$ ))**

Tout processus AR(1) est un processus MA( $\infty$ ).

### 10.2.3.1 Fonction d'autocovariance et d'autocorrélation théoriques

#### 10.2.3.1.1 Méthode par MA()

Le processus  $(X_t)_{t \in T}$  est un processus MA( $\infty$ ) donc la fonction d'autocorrélation théorique de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \rho: \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \frac{\sum_{i=0}^{+\infty} \varphi_1^i \varphi_1^{i+|h|}}{\sum_{j=0}^{+\infty} (\varphi_1^j)^2} = \frac{\sum_{i=0}^{+\infty} \varphi_1^{2i+|h|}}{\sum_{j=0}^{+\infty} \varphi_1^{2j}} = \varphi_1^h \end{aligned}$$

#### 10.2.3.1.2 Méthode par récurrence

$$\begin{aligned} \forall (t, h) \in T \times \mathbb{N}^* \quad \gamma(h) &= \text{Cov}(X_t, X_{t+h}) \\ &= \mathbb{E}(X_{t+h} X_t) - \mathbb{E}(X_t) \mathbb{E}(X_{t+h}) \\ &= \mathbb{E}(X_{t+h} X_t) \\ &= \mathbb{E}((\varphi_1 X_{t+h-1} + \varepsilon_{t+h}) X_t) \\ &= \varphi_1 \mathbb{E}(X_{t+h-1} X_t) + \mathbb{E}(\varepsilon_{t+h} X_t) \\ &= \varphi_1 \mathbb{E}(X_{t+h-1} X_t) + \mathbb{E}\left(\varepsilon_{t+h} \sum_{i=0}^{+\infty} \varphi_1^i \varepsilon_{t-i}\right) \\ &= \varphi_1 \mathbb{E}(X_{t+h-1} X_t) + \sum_{i=0}^{+\infty} \varphi_1^i \mathbb{E}(\varepsilon_{t+h} \varepsilon_{t-i}) \\ &= \varphi_1 \mathbb{E}(X_{t+h-1} X_t) \\ &= \varphi_1 (\mathbb{E}(X_{t+h-1} X_t) - \mathbb{E}(X_{t+h-1}) \mathbb{E}(X_t)) \\ &= \varphi_1 \text{Cov}(X_{t+h-1}, X_t) \\ &= \varphi_1 \gamma(h-1) \\ &\vdots \\ &= \varphi_1^h \gamma(0) \end{aligned}$$

La fonction d'autocovariance théorique de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \gamma: \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \begin{cases} \text{Var}(X_t) & \text{si } h = 0 \\ \varphi_1^h \text{Var}(X_t) & \text{si } h \neq 0 \end{cases} \end{aligned}$$

La fonction d'autocorrélation théorique est

$$\begin{aligned} \rho: \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \begin{cases} 1 & \text{si } h = 0 \\ \varphi_1^h & \text{si } h \neq 0 \end{cases} \end{aligned}$$

ou plus simplement

$$\begin{aligned} \rho: \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \varphi_1^h \end{aligned}$$

La mémoire de tout processus AR(1) est infinie.

En écrivant que,

$$\forall h \in \mathbb{Z}^* \quad \rho(h) = \text{sgn}(\varphi_1)^h e^{h \ln(|\varphi_1|)}$$

on remarque que la mémoire d'un processus AR(1) tend exponentiellement vers 0.

$$\rho(h) \xrightarrow{h \rightarrow +\infty} 0$$

## 10.3 Processus moyenne mobile MA

### Définition (Processus moyenne mobile d'ordre $q$ )

Soit  $q \in \mathbb{N}^* \cup \{+\infty\}$ ,  $(\theta_1, \dots, \theta_q) \in \mathbb{R}^{q-1} \times \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible).

Un processus **moyenne mobile** (en anglais, *Moving Average*) d'ordre  $q$ , noté  $\text{MA}(q)$ , de paramètres  $\theta_1, \dots, \theta_q$  et d'erreur  $(\varepsilon_t)_{t \in T}$  est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe  $c \in \mathbb{R}$  vérifiant

$$\forall t \in T \quad X_t = c + \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

ou encore, en posant  $\theta_0 = 1$ ,

$$\forall t \in T \quad X_t = c + \sum_{i=0}^q \theta_i \varepsilon_{t-i}$$

### Définition (Processus moyenne mobile centré d'ordre $q$ )

Soit  $q \in \mathbb{N}^*$ ,  $(\theta_1, \dots, \theta_q) \in \mathbb{R}^{q-1} \times \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible).

Le processus **moyenne mobile centré** d'ordre  $q$ , de paramètres  $\theta_1, \dots, \theta_q$  et d'erreur  $(\varepsilon_t)_{t \in T}$  est le processus stochastique  $(X_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t = \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

ou encore, en posant  $\theta_0 = 1$ ,

$$\forall t \in T \quad X_t = \sum_{i=0}^q \theta_i \varepsilon_{t-i}$$

Soit  $q \in \mathbb{N}^* \cup \{+\infty\}$ ,  $(\theta_1, \dots, \theta_q) \in \mathbb{R}^{q-1} \times \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc centré. Notons  $(X_t)_{t \in T}$  le processus  $\text{MA}(q)$  centré de paramètres  $\theta_1, \dots, \theta_p$  et d'erreur  $(\varepsilon_t)_{t \in T}$  i.e. défini par

$$\forall t \in T \quad X_t = \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

Introduisons l'opérateur retard  $L$ .

$$\forall t \in T \quad X_t = \varepsilon_t + \sum_{i=1}^q \theta_i L^i \varepsilon_t = \left( \mathbf{1} + \sum_{i=1}^q \theta_i L^i \right) \varepsilon_t$$

Posons  $\Theta_q(L) := \mathbf{1} + \sum_{i=1}^q \theta_i L^i$ . En posant  $\theta_0 = 1$ , on obtient  $\Theta_q(L) := \sum_{i=0}^q \theta_i L^i$ . On a

$$\forall t \in T \quad X_t = \Theta_q(L) \varepsilon_t$$

### 10.3.1 Espérance

$$\forall t \in T \quad \mathbb{E}(X_t) = \mathbb{E} \left( \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i} \right) = \mathbb{E}(\varepsilon_t) + \sum_{i=1}^q \theta_i \mathbb{E}(\varepsilon_{t-i}) = 0$$

#### 10.3.1.1 Fonction d'autocovariance et d'autocorrélation théorique

Notons  $\sigma$  l'écart type de  $(\varepsilon_t)_{t \in T}$ .

### 10.3.1.2 Ordre fini

La fonction d'autocovariance théorique de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \gamma : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \begin{cases} \sigma^2 \left( 1 + \sum_{k=1}^q \theta_k^2 \right) & \text{si } h = 0 \\ \sigma^2 \left( \theta_h + \sum_{k=1}^{q-|h|} \theta_k \theta_{k+|h|} \right) & \text{si } 0 < |h| \leq q \\ 0 & \text{si } |h| > q \end{cases} \end{aligned}$$

En posant  $\theta_0 = 1$  on obtient l'expression simplifiée suivante.

$$\begin{aligned} \gamma : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \begin{cases} \sigma^2 \sum_{k=0}^{q-|h|} \theta_k \theta_{k+|h|} & \text{si } 0 \leq |h| \leq q \\ 0 & \text{si } |h| > q \end{cases} \end{aligned}$$

On vérifie que

$$\forall t \in T \quad \gamma(0) = \text{Cov}(X_t, X_t) = \text{Var}(X_t) = \text{Var} \left( \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i} \right) = \text{Var}(\varepsilon_t) + \sum_{i=1}^q \theta_i^2 \text{Var}(\varepsilon_{t-i}) = \sigma^2 \left( 1 + \sum_{i=1}^q \theta_i^2 \right)$$

On a

$$\gamma(q) = \theta_q \sigma^2 \neq 0$$

#### Démonstration

##### Proposition (Mémoire d'un processus MA(q))

Soit  $q \in \mathbb{N}^*$ .

Tout processus MA(q) est à mémoire (finie) d'ordre  $q$ .

### 10.3.1.3 Ordre infini

La fonction d'autocovariance théorique de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \gamma : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \begin{cases} \sigma^2 \left( 1 + \sum_{k=1}^{+\infty} \theta_k^2 \right) & \text{si } h = 0 \\ \sigma^2 \left( \theta_h + \sum_{k=1}^{+\infty} \theta_k \theta_{k+|h|} \right) & \text{si } |h| > 0 \end{cases} \end{aligned}$$

En posant  $\theta_0 = 1$  on obtient l'expression simplifiée suivante.

$$\begin{aligned} \gamma : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \sigma^2 \sum_{k=0}^{+\infty} \theta_k \theta_{k+|h|} \end{aligned}$$

La fonction d'autocorrélation théorique de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \gamma : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \frac{\sigma^2 \sum_{i=0}^{+\infty} \theta_i \theta_{i+|h|}}{\sigma^2 \sum_{j=0}^{+\infty} \theta_j^2} = \frac{\sum_{i=0}^{+\infty} \theta_i \theta_{i+|h|}}{\sum_{j=0}^{+\infty} \theta_j^2} \end{aligned}$$

La mémoire ou fonction d'autocorrélation théorique de tout processus MA( $\infty$ ) ne dépend pas de la variance de son erreur.

### 10.3.2 Processus moyenne mobile d'ordre 1 MA(1)

Soit  $\theta_1 \in \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible) centré. Notons  $(X_t)_{t \in T}$  le processus centré MA(1) de paramètre  $\theta_1$  et d'erreur  $(\varepsilon_t)_{t \in T}$  *i.e.* défini par

$$\forall t \in T \quad X_t = \varepsilon_t + \theta_1 \varepsilon_{t-1}$$

Introduisons l'opérateur retard.

$$\forall t \in T \quad X_t = (\mathbf{1} + \theta_1 L) \varepsilon_t$$

Posons  $\Theta_1(L) = \mathbf{1} + \theta_1 L$ . On a

$$\forall t \in T \quad X_t = \Theta_1 \varepsilon_t$$

Si  $|\theta_1| < 1$

$$\forall t \in T \quad \varepsilon_t = \Theta_1(L)^{-1} X_t = (\mathbf{1} + \theta_1 L)^{-1} X_t = \left( \sum_{i=0}^{+\infty} (-\theta_1)^i L^i \right) X_t = \sum_{i=0}^{+\infty} (-\theta_1)^i L^i X_t = \sum_{i=0}^{+\infty} (-\theta_1)^i X_{t-i}$$

#### 10.3.2.1 Fonction d'autocovariance et d'autocorrélation théorique

Notons  $\sigma$  l'écart type de  $(\varepsilon_t)_{t \in T}$ .

La fonction d'autocovariance théorique de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \gamma : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \begin{cases} (1 + \theta_1^2) \sigma^2 & \text{si } h = 0 \\ \theta_1 \sigma^2 & \text{si } |h| = 1 \\ 0 & \text{si } |h| > 1 \end{cases} \end{aligned}$$

#### Démonstration

La fonction d'autocorrélation théorique de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \rho : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \begin{cases} 1 & \text{si } h = 0 \\ \frac{\theta_1}{(1 + \theta_1^2)} & \text{si } |h| = 1 \\ 0 & \text{sinon} \end{cases} \end{aligned}$$

#### Démonstration

La mémoire d'un processus MA(1) est (finie) d'ordre 1.

## 10.4 Processus mixtes

Soit  $(p, q) \in \mathbb{N}^2$ ,  $(\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q) \in \mathbb{R}^{p-1} \times \mathbb{R}^* \times \mathbb{R}^{q-1} \times \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc centré. Notons  $(X_t)_{t \in T}$  le processus ARMA( $p, q$ ) centré de paramètres  $\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q$  et d'erreur  $(\varepsilon_t)_{t \in T}$  *i.e.* défini par

$$\forall t \in T \quad X_t = \sum_{i=1}^p \varphi_i X_{t-i} + \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

On a

$$\forall t \in T \quad X_t - \sum_{i=1}^p \varphi_i X_{t-i} = \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$$

Introduisons l'opérateur retard  $L$ .

$$\forall t \in T \quad X_t - \sum_{i=1}^p \varphi_i L^i X_t = \varepsilon_t + \sum_{i=1}^q \theta_i L^i \varepsilon_t$$

On en déduit que

$$\forall t \in T \quad \left( \mathbf{1} - \sum_{i=1}^p \varphi_i L^i \right) X_t = \left( \mathbf{1} + \sum_{i=1}^q \theta_i L^i \right) \varepsilon_t$$

Finalement,

$$\forall t \in T \quad \Phi_p(L)X_t = \Theta_q(L)\varepsilon_t$$

**Théorème (Représentation canonique d'un processus ARMA)**

Soit  $(p, q) \in \mathbb{N}^{*2}$ ,  $(\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q) \in \mathbb{R}^{p-1} \times \mathbb{R}^* \times \mathbb{R}^{q-1} \times \mathbb{R}^*$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc faible.

Pour tout processus ARMA $(p, q)$  de paramètres  $\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q$  et d'erreur  $(\varepsilon_t)_{t \in T}$ ,  $(X_t)_{t \in T}$ , tel que  $\Phi_p$  et  $\Theta_q$  n'ont pas de racine de module égale à 1, il existe deux polynômes  $\tilde{\Phi}_p$  et  $\tilde{\Theta}_q$  n'ayant que des racines à l'extérieur du disque unité et un bruit blanc faible  $(\epsilon_t)_{t \in T}$  vérifiant

$$\forall t \in T \quad \tilde{\Phi}_p(L)X_t = \tilde{\Theta}_q \epsilon_t$$

Cette représentation est unique.

Notons  $p'$  le nombre de racine de  $\Phi_p$  de module strictement inférieur à 1,  $q'$  le nombre de racine de  $\Theta_q$  de module strictement inférieur à 1,  $\alpha_1, \dots, \alpha_{p'}$  les racines de  $\Phi_p$  et  $\beta_1, \dots, \beta_{q'}$  les racines de  $\Theta_q$  de sorte que

$$\forall (i, j) \in \{1, \dots, p'\} \times \{p' + 1, \dots, p\} \quad |\alpha_i| < 1 \text{ et } |\alpha_j| > 1$$

$$\forall (i, j) \in \{1, \dots, q'\} \times \{q' + 1, \dots, q\} \quad |\beta_i| < 1 \text{ et } |\beta_j| > 1$$

On a

$$\Phi_p(X) = \prod_{k=1}^p (1 - \alpha_k X) \quad \text{et} \quad \Theta_q(X) = \prod_{k=1}^q (1 - \beta_k X)$$

Posons

$$\tilde{\Phi}_p(X) = \prod_{i=1}^{p'} (1 - \alpha_i X) \prod_{i=p'+1}^p \left( 1 - \frac{1}{\alpha_i} X \right)$$

$$\tilde{\Theta}_q(X) = \prod_{k=1}^{q'} (1 - \beta_k X) \prod_{k=q'+1}^q \left( 1 - \frac{1}{\beta_k} X \right)$$

et

$$\epsilon_t^2 = \varepsilon_t^2 \frac{\prod_{i=q'+1}^q |\beta_i|^2}{\prod_{i=p'+1}^p |\alpha_i|^2}$$

Soit  $t \in T$ . Vérifions que,

$$\tilde{\Phi}_p(L)X_t = \left( \prod_{i=1}^{p'} (1 - \alpha_i L) \prod_{i=p'+1}^p \left( 1 - \frac{1}{\alpha_i} L \right) \right) X_t =$$

et que

$$\tilde{\Theta}_q(L)\epsilon_t = \left( \prod_{k=1}^{q'} (1 - \beta_k L) \prod_{k=q'+1}^q \left( 1 - \frac{1}{\beta_k} L \right) \right) \left( \varepsilon_t \frac{\prod_{i=q'+1}^q |\beta_i|}{\prod_{i=p'+1}^p |\alpha_i|} \right)$$

En posant  $\theta_0 = 1$ , la fonction d'autocovariance de  $(X_t)_{t \in T}$  est

$$\begin{aligned} \gamma : \mathbb{Z} &\longrightarrow \mathbb{R} \\ h &\longmapsto \begin{cases} \sum_{j=1}^p \varphi_j \gamma(k-j) + \sigma^2 \sum_{j=k}^q \theta_j \varphi_{j-k} & \text{si } 0 \leq k < \max(p, q+1) \\ \sum_{j=1}^p \varphi_j \gamma(k-j) & \text{si } k \geq \max(p, q+1) \end{cases} \end{aligned}$$

## 10.5 Modélisation par un modèle ARMA

### 10.5.1 Modélisation automatique

Dans le logiciel **R** les fonctions `ar()` et `auto.arima()` sont basées sur les critères d'information.

## 10.6 Exemple de modélisation par un modèle ARMA(p,q)

Considérons le jeu de données `airquality` du logiciel **R**.

```
data(airquality) # class(airquality) "data.frame"
```

Avant d'afficher les données il convient de prendre en compte les dimensions du jeu de données.

```
dim(airquality) # class "integer"
```

```
## [1] 153 6
```

L'affichage de l'ensemble des données obtenu avec la commande suivante nécessite plusieurs pages.

```
airquality
```

Affichons donc uniquement les 6 premières et 6 dernières observations.

```
head(airquality) # ou airquality[1:6,], class "data.frame"
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```
tail(airquality) # class "data.frame"
```

```
##   Ozone Solar.R Wind Temp Month Day
## 148    14     20 16.6   63     9  25
## 149    30    193  6.9   70     9  26
## 150    NA    145 13.2   77     9  27
## 151    14    191 14.3   75     9  28
## 152    18    131  8.0   76     9  29
## 153    20    223 11.5   68     9  30
```

Extrayons la série temporelle `Wind`. Cette série temporelle quotidienne représente la vitesse du vent mesurée à New-York de mai 1973 à septembre 1973. Affichons uniquement les 6 premières et 6 dernières observations.

```
Vent <- airquality$Wind # class "numeric"
```

```
## Autres méthodes, indicées
```

```
# Vent <- airquality[,3]
```

```
# Vent <- airquality[[3]]
```

```
## Autres méthodes, par nom
```

```
# Vent <- airquality[, "Wind"]
```

```
# Vent <- airquality[["Wind"]]
```

```
head(Vent) # class "numeric"
```

```
## [1] 7.4 8.0 12.6 11.5 14.3 14.9
```

```
tail(Vent) # class "numeric"
```

```
## [1] 16.6 6.9 13.2 14.3 8.0 11.5
```

Remarquez la différence avec les commandes suivantes.

```
Vent_df <- airquality[3] # class "data.frame"
## Autre méthode, par nom
# Vent_df <- airquality["Wind"]
head(Vent_df) # class "data.frame"

##      Wind
## 1   7.4
## 2   8.0
## 3  12.6
## 4  11.5
## 5  14.3
## 6  14.9

tail(Vent_df) # class "data.frame"

##      Wind
## 148 16.6
## 149  6.9
## 150 13.2
## 151 14.3
## 152  8.0
## 153 11.5
```

Cette série ne présente aucune valeur manquante.

```
which(is.na(Vent), arr.ind=TRUE)

## integer(0)
```

Cette série est constituée de 153 observations.

```
length(Vent) # class "integer"

## [1] 153
```

Posons  $T = \{1, \dots, 153\}$  et, pour tout  $t \in T$ ,  $X_t = \text{Vent}[t]$ . On cherche donc à modéliser la série temporelle  $(X_t)_{t \in T}$ . Traçons le chronogramme de cette série.

```
ts.plot(Vent, main="Chronogramme de la série Vent")
```

### Chronogramme de la série Vent

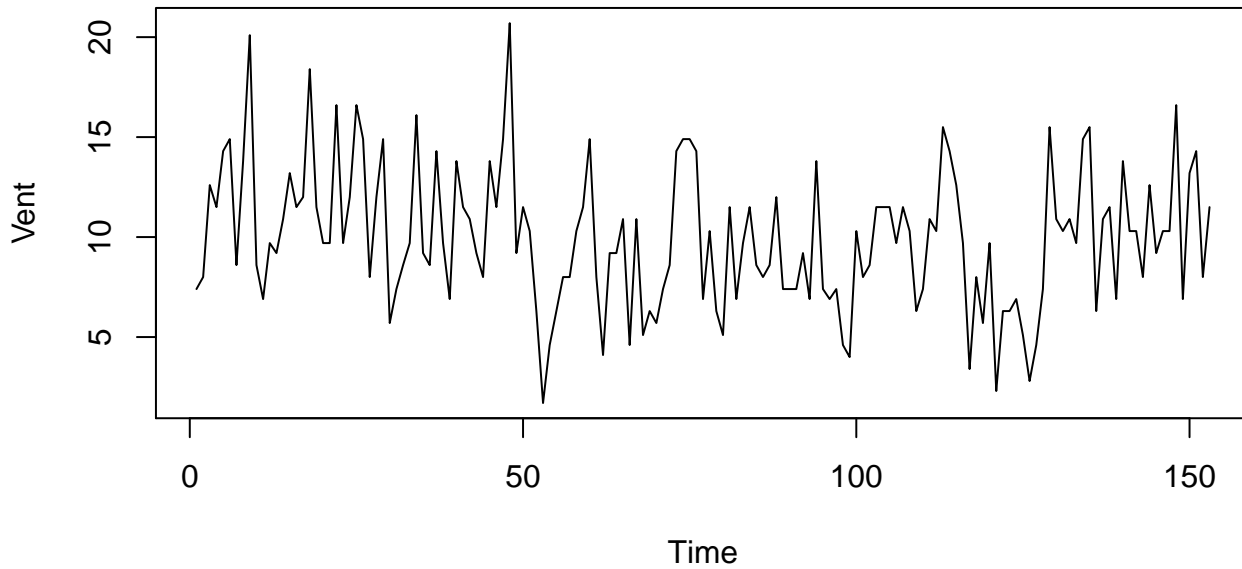


FIGURE 10.1 – Chronogramme de la série *Vent*

La figure montre bien que la série fluctue autour de sa moyenne, donc *a priori* on peut supposer qu'elle est stationnaire. Vérifions la stationnarité avec le test de Kwiatkowski-Phillips-Schmidt-Shin (KPSS).

## 10.6.1 Tests de stationnarité et test de blancheur

### 10.6.1.1 Test de stationnarité

```
library(tseries)
kpss.test(Vent) # class "htest"

##
## KPSS Test for Level Stationarity
##
## data: Vent
## KPSS Level = 0.41563, Truncation lag parameter = 4, p-value = 0.07042

# Extraction de la p-valeur
names(kpss.test(Vent))

## [1] "statistic" "parameter" "p.value" "method" "data.name"

kpss.test(Vent)$p.value

## [1] 0.07041757
```

On a `kpss.test(Vent)$p.value = 0.0704 > 0.05` donc le test valide la stationnarité.

### 10.6.1.2 Test de blancheur

Par curiosité effectuons un test de blancheur.

```
Box.test(Vent, lag=round(length(Vent)/4))

##
## Box-Pierce test
##
## data: Vent
## X-squared = 45.745, df = 38, p-value = 0.1815

# Extraction de la p-valeur
names(Box.test(Vent, lag=round(length(Vent)/4)))

## [1] "statistic" "parameter" "p.value" "method" "data.name"

Box.test(Vent, lag=round(length(Vent)/4))$p.value

## [1] 0.1814873
```

On a `Box.test(Vent, lag=round(length(Vent)/4))$p.value = 0.1815 > 0.05` donc la série `soleil` est modélisable par un bruit blanc.

Toutefois, modélisons la série `Vent` par un modèle ARMA. Supposons donc qu'il existe  $(p, q) \in \mathbb{N}^2$ ,  $\mu \in \mathbb{R}$ ,  $(\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q) \in \mathbb{R}^{p-1} \times \mathbb{R}^* \times \mathbb{R}^{q-1} \times \mathbb{R}^*$  et un bruit blanc faible  $(\varepsilon_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t - \mu = \sum_{k=1}^p \varphi_k (X_{t-k} - \mu) + \varepsilon_t + \sum_{k=1}^q \theta_k \varepsilon_{t-k}$$

ou encore, en posant  $\theta_0 = 1$ ,

$$\forall t \in T \quad X_t - \mu = \sum_{k=1}^p \varphi_k (X_{t-k} - \mu) + \sum_{k=0}^q \theta_k \varepsilon_{t-k}$$

## 10.6.2 Modélisation manuelle

### 10.6.2.1 Identification des degrés

#### 10.6.2.1.1 Identification du degré d'un MA et d'un AR

Modélisons la série **Vent** par un modèle AR et par un modèle MA.

- **Modèle AR** Supposons donc qu'il existe  $p \in \mathbb{N}^*$ ,  $\mu \in \mathbb{R}$ ,  $(\varphi_1, \dots, \varphi_p) \in \mathbb{R}^{p-1} \times \mathbb{R}^*$  et un bruit blanc faible  $(\varepsilon_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t - \mu = \sum_{k=1}^p \varphi_k (X_{t-k} - \mu) + \varepsilon_t$$

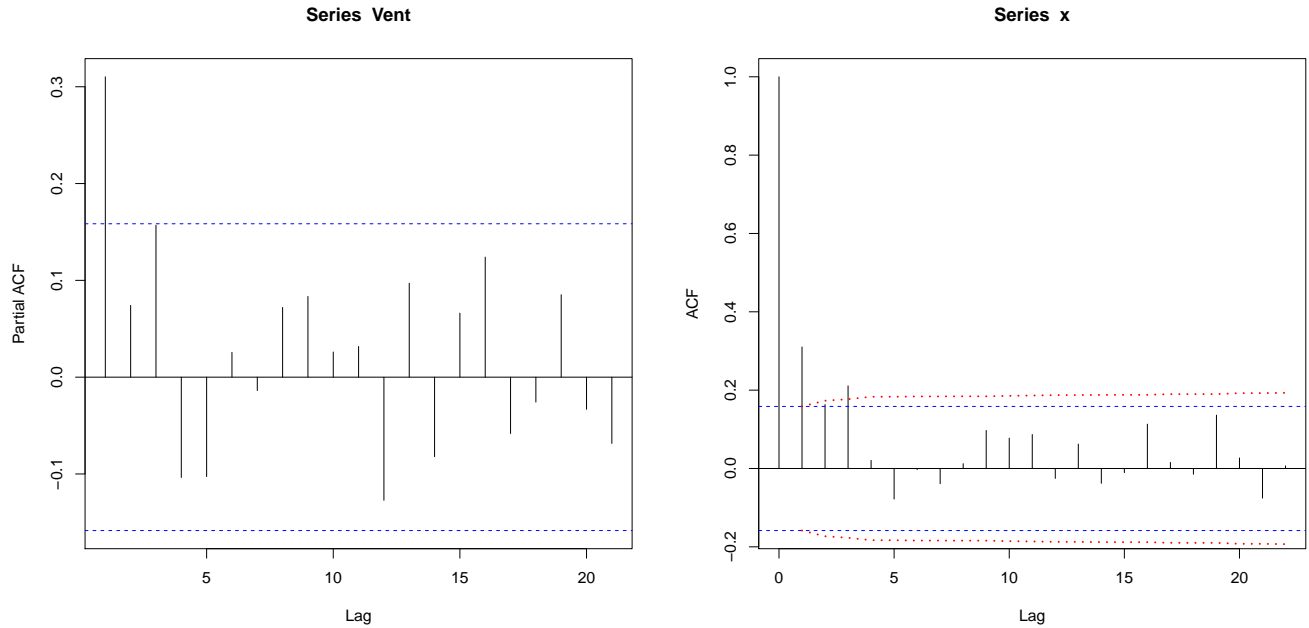
Identifions le degré  $p$  du modèle AR en représentant les mémoires partielles empiriques.

- **Modèle MA** Supposons donc qu'il existe  $q \in \mathbb{N}^*$ ,  $\mu \in \mathbb{R}$ ,  $(\theta_1, \dots, \theta_q) \in \mathbb{R}^{q-1} \times \mathbb{R}^*$  et un bruit blanc faible  $(\varepsilon_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t = \mu + \sum_{k=1}^q \theta_k \varepsilon_{t-k}$$

Identifions le degré  $q$  du modèle MA en représentant les mémoires empiriques.

```
# Par défaut, lag = 10log(length(Vent)) = 10log(153) ~ 21
pacf(Vent)
acfMA(Vent)
```



(a) Autocorrélogramme partiel empirique

(b) Autocorrélogramme empirique

FIGURE 10.2 – Autocorrélogrammes de la série **Vent**

- D'après la Figure 10.2 (a), pour tout  $h \in \{2, \dots, 21\}$ , l'autocorrélation partielle empirique d'ordre  $h$  de  $(X_t)_{t \in T}$ ,  $\hat{\rho}_p(h)$ , est comprise dans l'intervalle délimité par les pointillés bleus. On identifie donc le degré  $p = 1$  i.e. qu'on modélisera la série **Vent** par un modèle AR(1).  
Explicitons notre modèle. On suppose qu'il existe  $\mu \in \mathbb{R}$ ,  $\varphi_1 \in \mathbb{R}^*$  et un bruit blanc faible  $(\varepsilon_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t - \mu = \varphi_1 (X_{t-1} - \mu) + \varepsilon_t$$

- D'après la Figure 10.2 (b), pour tout  $h \in \{4, \dots, 21\}$ , l'autocorrélation empirique d'ordre  $h$  de  $(X_t)_{t \in T}$ ,  $\hat{\rho}(h)$ , est comprise dans l'intervalle délimité par les pointillés rouges. On identifie donc le degré  $q = 3$  i.e. qu'on modélisera la série **Vent** par un modèle MA(3).

Explicitons notre modèle. On suppose qu'il existe  $\mu \in \mathbb{R}$ ,  $(\theta_1, \theta_2, \theta_3) \in \mathbb{R}^3 \times \mathbb{R}^*$  et un bruit blanc faible  $(\varepsilon_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t = \mu + \varepsilon_t + \sum_{k=1}^3 \theta_k \varepsilon_{t-k} = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \theta_3 \varepsilon_{t-3}$$

ou encore, en posant  $\theta_0 = 1$ ,

$$\forall t \in T \quad X_t = \mu + \sum_{k=0}^q \theta_k \varepsilon_{t-k} = \mu + \theta_0 \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \theta_3 \varepsilon_{t-3}$$

#### 10.6.2.1.2 Identification des degrés d'un processus mixte ARMA

Identifions les degrés  $p$  et  $q$  du modèle ARMA en utilisant les autocorrélations étendues de  $(X_t)_{t \in T}$ .

```
eacf(Vent, ar.max=5)

## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x o o o o o o o o o o
## 1 x o x o o o o o o o o o o
## 2 x x x o x o o o o o o o o
## 3 x x o o o o o o o o o o o
## 4 x x x x o o o o o o o o o
## 5 x o o x x o o o o o o o o
```

Les autocorrélations étendues confirment que le modèle MA(3) convient bien mais n'identifie pas le modèle AR(1) comme un modèle potentiel. Dans la suite nous allons considérer les modèles MA(3) et AR(1).

#### 10.6.2.2 Estimation des paramètres

##### 10.6.2.2.1 Estimation des paramètres du modèle AR(1)

```
library(forecast) # fonction Arima()
modele1 <- Arima(Vent, order=c(1,0,0))
modele1 # class "forecast_ARIMA" "ARIMA" "Arima"

## Series: Vent
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1      mean
##          0.3097  9.9546
## s.e.      0.0767  0.3897
##
## sigma^2 estimated as 11.29: log likelihood=-401.54
## AIC=809.08   AICc=809.24   BIC=818.17
```

##### 10.6.2.2.2 Estimation des paramètres du modèle MA(3)

```
modele2 <- Arima(Vent, order=c(0,0,3))
modele2 # class "forecast_ARIMA" "ARIMA" "Arima"

## Series: Vent
## ARIMA(0,0,3) with non-zero mean
##
## Coefficients:
##          ma1      ma2      ma3      mean
```

```
##      0.2584  0.1318  0.2231  9.9462
## s.e.  0.0789  0.0753  0.0791  0.4248
##
## sigma^2 estimated as 11:  log likelihood=-398.59
## AIC=807.18   AICc=807.59   BIC=822.33
```

### 10.6.2.3 Réduction du modèle

#### 10.6.2.3.1 Réduction du modèle AR(1)

```
t_stat(modele1) # class "matrix"

##           ar1 intercept
## t.stat 4.038835 25.54192
## p.val  0.000054  0.00000
```

On a `as.data.frame(t_stat(modele1))["p.val", "ar1"] =  $5.4 \times 10^{-5} < 0.05$`  donc le paramètre  $\varphi_1$  ne peut pas être supprimé. Le modèle AR(1) n'est donc pas simplifiable.

#### 10.6.2.3.2 Réduction du modèle MA(3)

```
t_stat(modele2) # class "matrix"

##           ma1      ma2      ma3 intercept
## t.stat 3.274910 1.750207 2.820555 23.41412
## p.val  0.001057 0.080083 0.004794  0.00000
```

On a `as.data.frame(t_stat(modele2))["p.val", "ma3"] = 0.004794 < 0.05` donc le paramètre  $\theta_3$  ne peut pas être supprimé. Le modèle MA(3) n'est donc pas simplifiable.

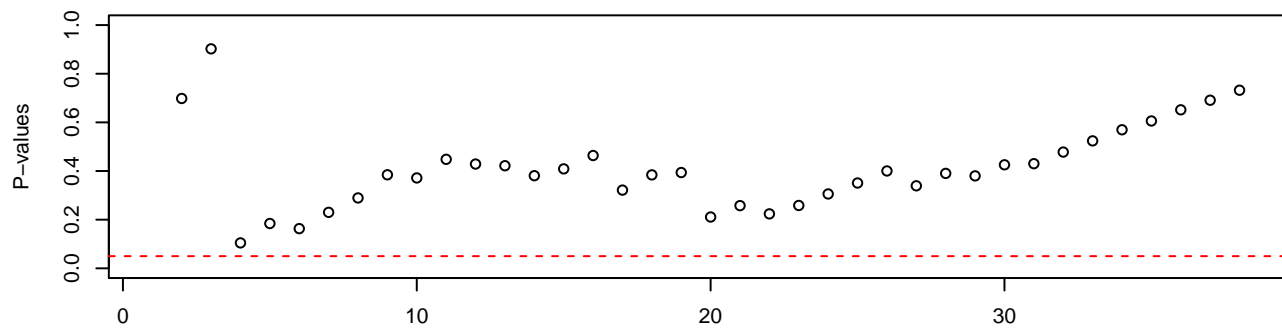
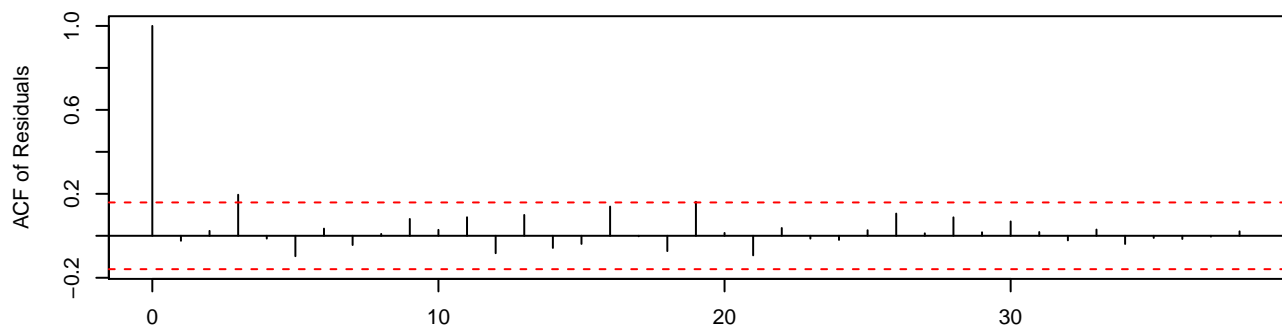
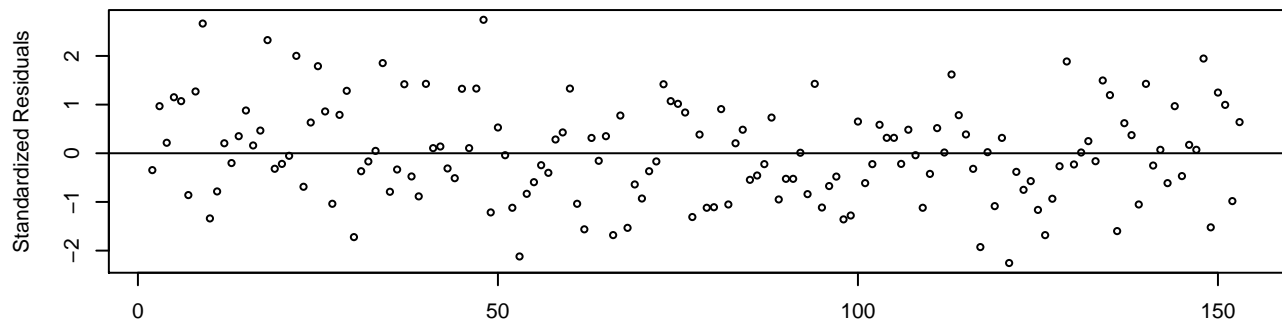
### 10.6.2.4 Validation du modèle

Validons les modèles à l'aide tests de Ljung-Box.

**Remarque** Le package TSA n'étant plus disponible il faut extraire le code source de la fonction `tsdiag.Arima()` et donc de la fonction `LB.test()`.

#### 10.6.2.4.1 Validation du modèle AR(1)

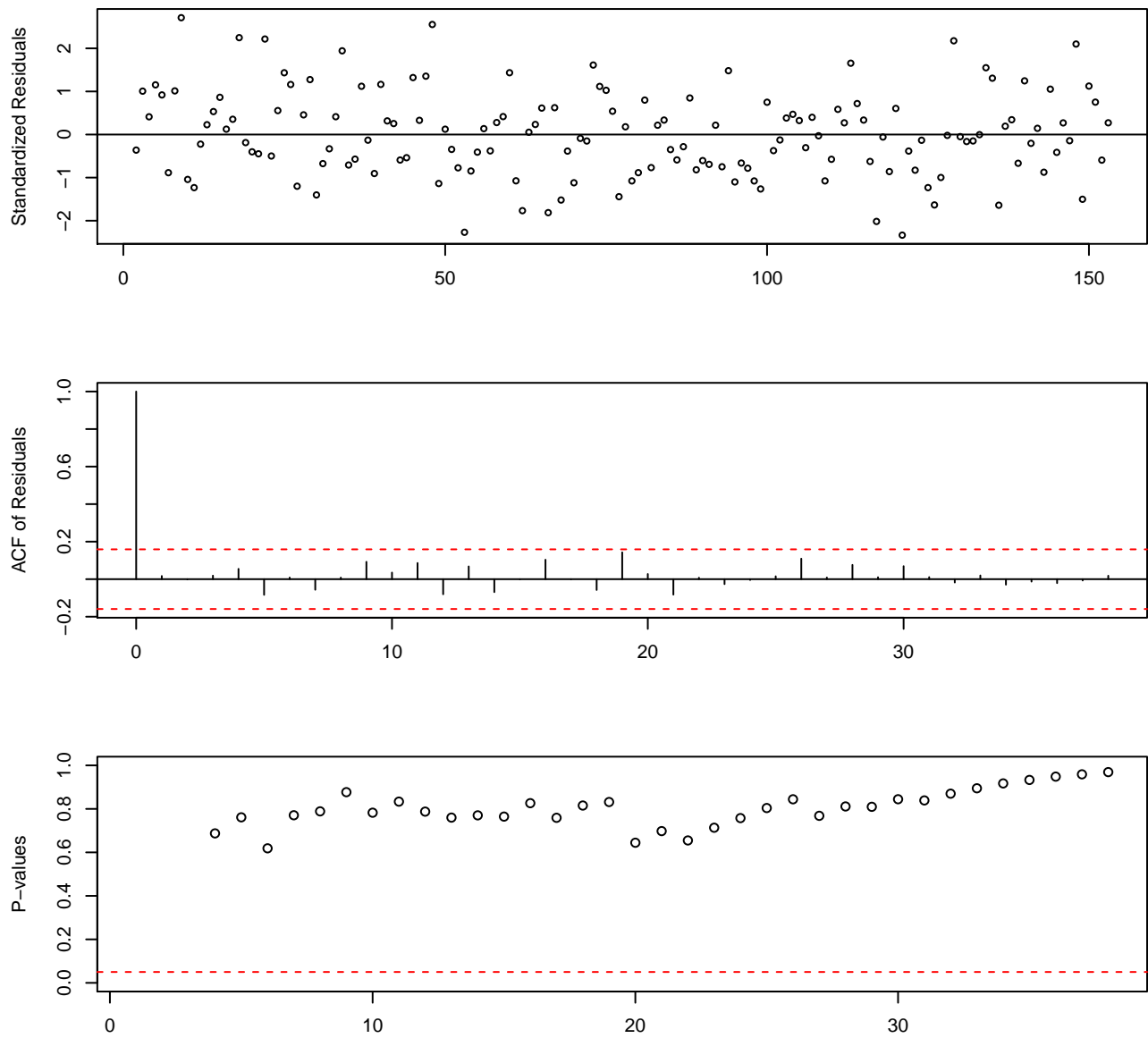
```
tsdiag.Arima(modele1, gof.lag=round(length(Vent)/4))
```



Les p-valeurs étant supérieures à 5%, on conclut que les résidus peuvent être considérés comme issus d'un bruit blanc. Le modèle AR(1) estimé est donc valide.

#### 10.6.2.4.2 Validation du modèle MA(3)

```
tsdiag.Arima(modele2, gof.lag=round(length(Vent)/4))
```



Les p-valeurs étant supérieures à 5%, on conclut que les résidus peuvent être considérés comme issus d'un bruit blanc. Le modèle  $MA(3)$  estimé est donc valide.

## 10.6.2.5 Explicitation des modèles

### 10.6.2.5.1 Explicitation du modèle AR(1)

```
## Extraction des coefficients
library(lmtest) # fonction coeftest()

## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following object is masked from 'package:timeSeries':
##
##   time<-
## The following objects are masked from 'package:base':
##
##   as.Date, as.Date.numeric

coeftest(modele1) # class "coeftest"

##
## z test of coefficients:
##
##           Estimate Std. Error z value Pr(>|z|)
## ar1           0.30974    0.07669  4.0388 5.372e-05 ***
## intercept    9.95456    0.38973 25.5419 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Paramètre varphi_1
coeftest(modele1)[1,1]

## [1] 0.3097385

# Coefficient mu
coeftest(modele1)[2,1]

## [1] 9.954557

# Variance du bruit
names(modele1)

## [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
## [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
## [13] "nobs"      "model"     "aicc"      "bic"       "x"         "fitted"

modele1$sigma2

## [1] 11.28513
```

Finalement, la série temporelle **Vent** peut être modélisée par un processus AR(1) défini par

$$\forall t \in T \quad X_t - \mu = \varphi_1(X_{t-1} - \mu) + \varepsilon_t$$

avec

- $\mu = \text{coeftest}(\text{modele1})[2,1] = 9.9546$
- $\varphi_1 = \text{coeftest}(\text{modele1})[1,1] = 0.3097$
- $(\varepsilon_t)_{t \in T}$  un bruit blanc faible de variance  $\text{modele1}\$sigma2 = 11.2851$ .

### 10.6.2.5.2 Explicitation du modèle MA(3)

```
## Extraction des coefficients
coeftest(modele2) # class "coeftest"

##
## z test of coefficients:
##
##           Estimate Std. Error z value Pr(>|z|)
## ma1         0.258414   0.078907  3.2749  0.001057 **
## ma2         0.131777   0.075292  1.7502  0.080083 .
## ma3         0.223072   0.079088  2.8206  0.004794 **
## intercept   9.946240   0.424797 23.4141 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

# Paramètre theta_1, theta_2, theta_3
c(coeftest(modele2)[1,1], coeftest(modele2)[2,1], coeftest(modele2)[3,1])

## [1] 0.2584138 0.1317765 0.2230716

# Coefficient mu
coeftest(modele2)[4,1]

## [1] 9.94624

# Variance du bruit
names(modele2)

## [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
## [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
## [13] "nobs"      "model"     "aicc"      "bic"       "x"         "fitted"

modele2$sigma2

## [1] 10.99604
```

Finalement, la série temporelle **Vent** peut être modélisée par un processus MA(3) défini par

$$\forall t \in T \quad X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \theta_3 \varepsilon_{t-3}$$

avec

- $\mu = \text{coeftest}(\text{modele2})[4,1] = 9.9462$
- $\theta_1 = \text{coeftest}(\text{modele2})[1,1] = 0.2584$
- $\theta_2 = \text{coeftest}(\text{modele2})[2,1] = 0.1318$
- $\theta_3 = \text{coeftest}(\text{modele2})[3,1] = 0.2231$
- $(\varepsilon_t)_{t \in T}$  un bruit blanc faible de variance  $\text{modele2}\$sigma2 = 10.996$ .

### 10.6.3 Modélisation automatique

La modélisation automatique fait appel à un système expert qui choisira un modèle basé sur une stratégie fixée par l'utilisateur.

#### 10.6.3.1 Modélisation automatique par un modèle AR

Supposons qu'il existe  $p \in \mathbb{N}^*$ ,  $\mu \in \mathbb{R}$ ,  $(\varphi_1, \dots, \varphi_p) \in \mathbb{R}^{p-1} \times \mathbb{R}^*$  et un bruit blanc faible  $(\varepsilon_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t - \mu = \sum_{k=1}^p \varphi_k (X_{t-k} - \mu) + \varepsilon_t$$

##### 10.6.3.1.1 Identification du degré, estimation des paramètres et réduction du modèle

```
modele3 <- ar(Vent) # package stats
modele3 # class "ar"
```

```
##
## Call:
## ar(x = Vent)
##
## Coefficients:
##      1      2      3
## 0.2756 0.0291 0.1570
##
## Order selected 3  sigma^2 estimated as 11.1
```

```
## Extraction des valeurs estimées
names(modele3)
```

```
## [1] "order"      "ar"          "var.pred"    "x.mean"      "aic"
## [6] "n.used"     "n.obs"       "order.max"   "partialacf"  "resid"
## [11] "method"     "series"      "frequency"   "call"        "asy.var.coef"
```

```
# Coefficient mu
modele3$x.mean
```

```
## [1] 9.957516
```

```
# Paramètres phi_1, phi_2, phi_3
modele3$ar
```

```
## [1] 0.27564626 0.02905691 0.15697134
```

```
# Variance du bruit
modele3$var.pred
```

```
## [1] 11.09905
```

D'après la fonction `ar()`, la série `Vent` est modélisable par un processus AR et le processus AR minimisant le critère d'information d'Akaike (AIC) est le processus  $\text{AR}(\text{modele3\$order}) = \text{AR}(3)$  défini par

$$\forall t \in T \quad X_t - \mu = \sum_{k=1}^3 \varphi_k (X_{t-k} - \mu) + \varepsilon_t$$

avec

- $\mu = \text{modele3\$x.mean} = 9.9575$

- $\varphi_1 = \text{modele3\$ar}[1] = 0.2756$   
 $\varphi_2 = \text{modele3\$ar}[2] = 0.0291$   
 $\varphi_3 = \text{modele3\$ar}[3] = 0.157$
- $(\varepsilon_t)_{t \in T}$  un bruit blanc de variance  $\text{modele3\$var.pred} = 11.099$

#### 10.6.3.1.2 Validation du modèle AR(3)

Vérifions que le modèle proposé par la fonction `ar()` est valide.

```
#tsdiag.Arima(modele3, gof.lag=round(length(Vent)/4))
```

Le modèle AR(3) proposé est valide.

### 10.6.3.2 Modélisation automatique par un processus AR(I)MA

Supposons qu'il existe  $(p, q) \in \mathbb{N}^2$ ,  $\mu \in \mathbb{R}$ ,  $(\varphi_1, \dots, \varphi_p, \theta_1, \dots, \theta_q) \in \mathbb{R}^{p-1} \times \mathbb{R}^* \times \mathbb{R}^{q-1} \times \mathbb{R}^*$  et un bruit blanc faible  $(\varepsilon_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t - \mu = \sum_{k=1}^p \varphi_k (X_{t-k} - \mu) + \varepsilon_t + \sum_{k=1}^q \theta_k \varepsilon_{t-k}$$

ou encore, en posant  $\theta_0 = 1$ ,

$$\forall t \in T \quad X_t - \mu = \sum_{k=1}^p \varphi_k (X_{t-k} - \mu) + \sum_{k=0}^q \theta_k \varepsilon_{t-k}$$

#### 10.6.3.2.1 Identification du degré, estimation des paramètres et réduction du modèle

```
modele4 <- auto.arima(Vent, d=0, max.p=5, max.q=5, ic="aic") # package forecast
modele4 # class "forecast_ARIMA" "ARIMA" "Arima"

## Series: Vent
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1      mean
##          0.3097  9.9546
## s.e.      0.0767  0.3897
##
## sigma^2 estimated as 11.29: log likelihood=-401.54
## AIC=809.08   AICc=809.24   BIC=818.17
```

La procédure `auto.arima()` propose le même modèle AR(1) que celui sélectionné manuellement.

## 10.6.4 Comparaison des modèles AR(1), MA(3) et AR(3) et sélection du modèle

### 10.6.4.1 Critères d'information

#### 10.6.4.1.1 Tests de normalité

Vérifions que les résidus des modèles sont gaussiens.

##### 10.6.4.1.1.1 Test de Shapiro-Wilk

```
# class "htest"
shapiro.test(modele1$residuals) # ou plus simplement shapiro.test(modele1$res)

##
## Shapiro-Wilk normality test
##
## data:  modele1$residuals
## W = 0.99089, p-value = 0.4322

shapiro.test(modele2$residuals) # ou plus simplement shapiro.test(modele1$res)

##
## Shapiro-Wilk normality test
##
## data:  modele2$residuals
## W = 0.99119, p-value = 0.4624

shapiro.test(modele3$resid) # ou plus simplement shapiro.test(modele1$res)

##
## Shapiro-Wilk normality test
##
## data:  modele3$resid
## W = 0.9917, p-value = 0.5317
```

```
# Extractions des p-valeurs
names(shapiro.test(modele1$residuals))

## [1] "statistic" "p.value" "method" "data.name"
```

- On a `shapiro.test(modele1$residuals)$p.value = 0.4322 > 0.05` donc les résidus du modèle AR(1) sont gaussiens.
- On a `shapiro.test(modele2$residuals)$p.value = 0.4624 > 0.05` donc les résidus du modèle MA(3) sont gaussiens.
- On a `shapiro.test(modele3$resid)$p.value = 0.5317 > 0.05` donc les résidus du modèle AR(3) sont gaussiens.

##### 10.6.4.1.1.2 Test d'Agostino

```
library(fBasics) # fonction dagoTest()
# class "fHTEST" attr("package") "fBasics"
dagoTest(modele1$residuals) # ou plus simplement dagoTest(modele1$res)

##
## Title:
## D'Agostino Normality Test
##
## Test Results:
```

```
## STATISTIC:
## Chi2 | Omnibus: 2.3604
## Z3 | Skewness: 1.4782
## Z4 | Kurtosis: -0.4187
## P VALUE:
## Omnibus Test: 0.3072
## Skewness Test: 0.1394
## Kurtosis Test: 0.6754
##
## Description:
## Sun May 8 12:33:37 2022 by user:

dagoTest(modele2$residuals) # ou plus simplement dagoTestt(modele1$res)

##
## Title:
## D'Agostino Normality Test
##
## Test Results:
## STATISTIC:
## Chi2 | Omnibus: 2.0537
## Z3 | Skewness: 1.4311
## Z4 | Kurtosis: 0.0756
## P VALUE:
## Omnibus Test: 0.3581
## Skewness Test: 0.1524
## Kurtosis Test: 0.9397
##
## Description:
## Sun May 8 12:33:37 2022 by user:

dagoTest(modele3$resid) # ou plus simplement dagoTest(modele1$res)

##
## Title:
## D'Agostino Normality Test
##
## Test Results:
## STATISTIC:
## Chi2 | Omnibus: 1.7697
## Z3 | Skewness: 1.2888
## Z4 | Kurtosis: -0.3299
## P VALUE:
## Omnibus Test: 0.4128
## Skewness Test: 0.1975
## Kurtosis Test: 0.7415
##
## Description:
## Sun May 8 12:33:37 2022 by user:

# Extractions des p-valeurs (consulter la documentation)
dagoTest(modele1$residuals)$test

## $statistic
## Chi2 | Omnibus Z3 | Skewness Z4 | Kurtosis
## 2.3603593 1.4781787 -0.4187447
```

```
##
## $method
## [1] "D'Agostino Omnibus Normality Test"
##
## $p.value
## Omnibus Test Skewness Test Kurtosis Test
##      0.3072235      0.1393600      0.6754027
##
## $data.name
## [1] "modele1$residuals"

dagoTest(modele1$residuals)$test$p.value

## Omnibus Test Skewness Test Kurtosis Test
##      0.3072235      0.1393600      0.6754027
```

- On a `dagoTest(modele1$residuals)$test$p.value[1] = 0.3072 > 0.05` donc les résidus du modèle AR(1) sont gaussiens.
- On a `dagoTest(modele2$residuals)$test$p.value[1] = 0.3581 > 0.05` donc les résidus du modèle MA(3) sont gaussiens.
- On a `dagoTest(modele3$resid)$test$p.value[1] = 0.4128 > 0.05` donc les résidus du modèle AR(3) sont gaussiens.

#### 10.6.4.1.2 Comparaison des critères d'information

La Table résume les critères d'informations obtenus par chaque modèle.

```
## Extraction des critères d'information du modèle AR(1)
c(modele1$aic, modele1$aicc, modele1$bic)

## [1] 809.0762 809.2373 818.1675

## Extraction des critères d'information du modèle MA(3)
c(modele2$aic, modele2$aicc, modele2$bic)

## [1] 807.1815 807.5897 822.3337
```

Comment obtenir les critères d'information du modèle obtenu avec la fonction `ar()` ?

Méthode	Modèle	AIC	AICc	BIC
Manuelle <code>Arima()</code> et automatique <code>auto.arima()</code>	AR(1)	809.1	809.2	<b>818.2</b>
Manuelle <code>Arima()</code>	MA(3)	<b>807.2</b>	<b>807.6</b>	822.3
Automatique <code>ar()</code>	AR(3)			

TABLE 10.1 – Comparaison des modèles pour la série **Vent** par critères d'information

D'après le critère d'information d'Akaike (AIC) et le critère d'information d'Akaike corrigé (AICc) le modèle MA(3) obtenue par la méthode manuelle avec la fonction `Arima()` est légèrement préférable aux autres. En revanche, le critère d'information bayésien (BIC) privilégie le modèle AR(1) obtenus par les deux méthodes.

Rappelons que le critère d'information bayésien (BIC) a tendance à privilégier un modèle avec moins de paramètres. En effet la pénalisation des paramètres est plus forte en comparaison avec l'AIC ou même l'AICc.

#### 10.6.4.2 Pouvoir prédictif

##### 10.6.4.2.1 Ajustement

```
accuracy(modele1) # class "matrix"
```

```
##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.00599268 3.337306 2.693458 -15.92486 35.07431 0.8329717
##           ACF1
## Training set -0.02190852

accuracy(modele2)

##           ME      RMSE      MAE      MPE      MAPE      MASE      ACF1
## Training set 0.0100193 3.272394 2.623673 -15.61257 34.3922 0.8113902 0.01876507

accuracy(modele3)

## Error in accuracy.default(modele3): First argument should be a forecast object or a time series.
```

Méthode	Modèle	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Manuelle Arima() et automatique auto.arima()	AR(1)	<b>0.006</b>	3.3373	2.6935	-15.9249	35.0743	0.833	<b>-0.0219</b>
Manuelle Arima()	MA(3)	0.01	<b>3.2724</b>	<b>2.6237</b>	-15.6126	<b>34.3922</b>	<b>0.8114</b>	0.0188
Automatique ar()	AR(3)							

TABLE 10.2 – Comparaison des modèles pour la série **Vent** par indicateurs d'écart

Les critères RMSE, MAE, MASE et MAPE s'accordent pour établir que le modèle MA(3) est celui qui fournit le meilleur ajustement. Il faut aussi remarquer que ces critères indiquent que le modèle AR(3) a un meilleur ajustement aux données que le modèle AR(1); résultat attendu car théoriquement plus on augmente le nombre de paramètres plus les résidus diminuent.

#### 10.6.4.2.2 Sur échantillon témoin

Afin de comparer correctement les modèles, il est préférable de considérer un échantillon d'apprentissage et un échantillon témoin contenant 5% des observations. On applique les critères d'ajustement aux prévisions fournies par les modèles.

```
modele1b <- Arima(Vent[1:145], order=c(0,0,3))
modele2b <- Arima(Vent[1:145], order=c(1,0,0))
modele3b <- Arima(Vent[1:145], order=c(3,0,0))
fit1 <- forecast(modele1b, h=8)
fit2 <- forecast(modele2b, h=8)
fit3 <- forecast(modele3b, h=8)
```

Puis on calcule les indicateurs d'écart sur les prévisions grâce à la fonction `accuracy()` du package `forecast`.

```
accuracy(fit1, Vent[146:153])

##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.009718661 3.254422 2.613189 -15.770556 34.61967 0.8230516
## Test set     1.568391966 3.479914 2.816009  6.945897 23.90177 0.8869321
##           ACF1
## Training set 0.0172828
## Test set     NA

accuracy(fit2, Vent[146:153])

##           ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.006768966 3.309626 2.660416 -15.928380 34.97119 0.8379264
## Test set     1.562672331 3.392180 2.769166  7.036918 23.59212 0.8721784
```

```
##                               ACF1
## Training set -0.01788322
## Test set      NA

accuracy(fit3, Vent[146:153])

##                               ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.009499774 3.268540 2.631800 -15.655410 34.61333 0.8289133
## Test set     1.560760626 3.385856 2.747195  7.064515 23.33594 0.8652583
##                               ACF1
## Training set 0.01802019
## Test set     NA
```

Méthode	Modèle	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Manuelle <code>Arima()</code> et automatique <code>auto.arima()</code>	AR(1)	1.5627	3.3922	2.7692	7.0369	23.5921	0.8722	<b>-15.9284</b>
Manuelle <code>Arima()</code>	MA(3)	1.5684	3.4799	2.816	<b>6.9459</b>	23.9018	0.8869	NA
Automatique <code>ar()</code>	AR(3)	<b>1.5608</b>	<b>3.3859</b>	<b>2.7472</b>	7.0645	<b>23.3359</b>	<b>0.8653</b>	NA

TABLE 10.3 – Comparaison des modèles pour la série **Vent** par indicateurs d'écart sur échantillon témoin

Les critères RMSE, MAE, MASE et MAPE calculés sur l'échantillon témoin indiquent que le modèle AR(3) a le meilleur pouvoir prédictif. On choisit donc ce dernier comme modèle final pour la série **Vent**.

### 10.6.5 Prévisions du modèle AR(3)

On a vérifié que les résidus des modèles sont gaussiens donc on peut inclure les intervalles de précision.

```
prev <- forecast(modele3, h=20)
plot(prev)
```

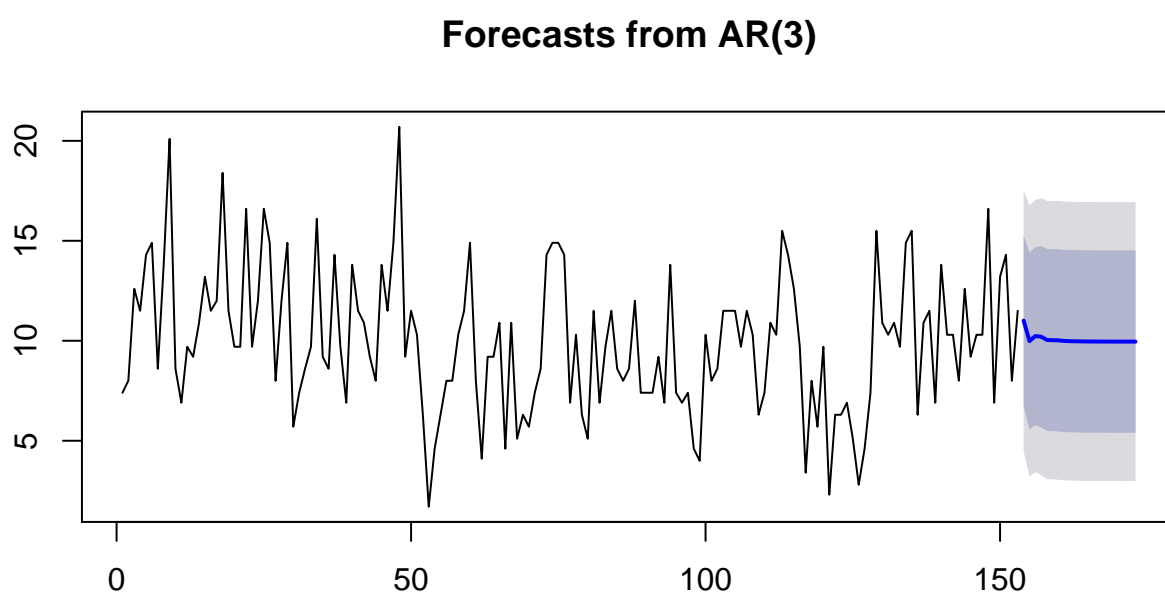


FIGURE 10.3 – Prédiction du modèle AR(3) pour la série *Vent*

## Chapitre 11

# Modèles pour l'hétéroscédasticité conditionnelle

### Définition (Filtration naturelle d'un processus stochastique)

Soit  $(\mathcal{E}_t)_{t \in T}$  un processus stochastique.

La **filtration naturelle** de  $(\mathcal{E}_t)_{t \in T}$ , notée  $(\mathcal{F}_t)_{t \in T}$ , est la suite croissante des tribus engendrés par  $\{\mathcal{E}_s \mid s \in T \text{ et } s \leq t\}$  (sur  $\{\mathcal{E}_t \mid t \in T\}$ ) *i.e.*

$$\forall t \in T \quad \mathcal{F}_t = \sigma(\mathcal{E}_1, \dots, \mathcal{E}_t)$$

revoir la notation 1 et à mettre en partie 1.

Soit  $(\mathcal{E}_t)_{t \in T}$  un processus stochastique. Intuitivement, pour tout  $t \in T$ ,  $\mathcal{F}_t$  représente l'information que nous possédons en observant les valeurs du processus pour tous les instants antérieurs ou égaux à  $t$ .

### Définition (Hétéroscédasticité conditionnelle)

Un **processus conditionnellement hétéroscédastique** est un processus stochastique  $(\mathcal{E}_t)_{t \in T}$  dont la variance conditionnelle dépend du temps *i.e.* tel qu'il existe une fonction non constante  $\sigma_t \in \mathbb{R}^T$  vérifiant

$$\forall t \in T \quad \text{Var}(\mathcal{E}_t \mid \mathcal{F}_{t-1}) = \sigma_t^2$$

Les séries conditionnellement hétéroscédastiques sont caractérisées par des régimes ayant des valeurs différentes. Pour décrire cette hétéroscédasticité conditionnelle deux approches sont considérées dans la littérature.

- Dans la première approche, la variance  $\sigma_t^2$  est supposée déterministe. Cette approche a été considérée par plusieurs auteurs (Starica et Mikosh et Granger et Starica (2006) entre autres), mais elle est assez peu utilisée en pratique.
- Dans la seconde approche, la variance  $\sigma_t^2$  est supposée stochastique. Cette approche a été suggérée par Engle (1982), en introduisant le modèle ARCH, modèle qui a été beaucoup utilisé dans la modélisation des séries financières.

## 11.1 Modèles GARCH

### Définition (Modèle à hétéroscédasticité conditionnelle autorégressive généralisé)

Soit  $(p, q) \in \mathbb{N}^{*2}$ ,  $(\alpha_1, \dots, \alpha_p, \beta_1, \dots, \beta_q) \in \mathbb{R}^{p-1} \times \mathbb{R}^* \times \mathbb{R}^{q-1} \times \mathbb{R}^*$  et  $(z_t)_{t \in T}$  un bruit blanc standard

Un modèle à **hétéroscédasticité conditionnelle autorégressive généralisé** (en anglais, *Generalized Autoregressive Conditional Heteroscedasticity*) est un processus stochastique  $(\varepsilon_t)_{t \in T}$  tel qu'il existe  $\alpha_0 \in \mathbb{R}^*$  vérifiant

$$\forall t \in T \quad \begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \alpha_0 + \sum_{i=1}^p \alpha_i \varepsilon_{t-i}^2 \end{cases} \quad \text{où} \quad \sigma_t^2 := \mathbb{E}(\varepsilon_t^2 \mid \mathcal{F}_{t-1})$$

### 11.1.1 Modèles ARCH

#### Définition (Modèle à hétéroscédasticité conditionnelle autorégressive)

Soit  $p \in \mathbb{N}^*$ ,  $(\alpha_1, \dots, \alpha_p) \in \mathbb{R}^{p-1} \times \mathbb{R}^*$  et  $(z_t)_{t \in T}$  un bruit blanc standard

Un modèle à **hétéroscédasticité conditionnelle autorégressive** (en anglais, *Autoregressive Conditional Heteroscedasticity*) est un processus stochastique  $(\varepsilon_t)_{t \in T}$  tel qu'il existe  $\alpha_0 \in \mathbb{R}^*$  vérifiant

$$\forall t \in T \quad \begin{cases} \varepsilon_t = \sigma_t z_t \\ \sigma_t^2 = \alpha_0 + \sum_{i=1}^p \alpha_i \varepsilon_{t-i}^2 \end{cases} \quad \text{où} \quad \sigma_t^2 := \mathbb{E}(\varepsilon_t^2 \mid \mathcal{F}_{t-1})$$

Le(s) modèle(s) à **hétéroscédasticité conditionnelle autorégressive** ou **autorégressif à hétéroscédasticité conditionnelle** (en anglais *AutoRegressive Conditional Heteroscedasticity*), noté ARCH, permettent de modéliser des chroniques qui ont une volatilité (ou variance ou encore variabilité) instantanée qui dépend du passé. Il est ainsi possible d'élaborer une prévision dynamique de la chronique en termes de moyenne et de variance.

**Histoire** Présentés initialement par Engle (1982), ces modèles ont connu des développements et des applications très importants par la suite.

L'étude des séries temporelles financières s'est donc trouvée confrontée à deux types de problèmes :

- la non stationnarité des séries
- le caractère leptokurtique (= les queues de probabilité sont plus épaisses que celles d'une loi normale aux extrémités, les valeurs anormales sont donc plus fréquentes) de la distribution des données.

L'intérêt de cette formulation réside dans l'interdépendance d'une variable endogène au modèle. Pour Engle, cette notion est très importante en finance car le risque d'erreur n'est pas le même selon les périodes : il y a alternance de périodes d'accalmie et de périodes d'euphorie.

#### 11.1.1.1 Modèle ARCH(1)

## 11.2 Exemple de modélisation par processus GARCH

## Troisième partie

# Séries temporelles non stationnaires

**Définition (Série non stationnaire)**

Une **série non stationnaire** est une série temporelle (ou série chronologique) qui n'est pas stationnaire.

On peut reformuler cette définition sous la forme équivalente suivante.

Une **série non stationnaire** est une série temporelle  $(X_t)_{t \in T}$  telle que

Pour modéliser une série temporelle non stationnaire, nous devons d'abord identifier la nature de la non stationnarité afin de la neutraliser.

# Chapitre 12

## Taxinomie de la non stationnarité

### 12.1 Non stationnarité du premier ordre non stationnarité en moyenne

#### Définition (Processus non stationnaire en moyenne ou non stationnaire du premier ordre)

Un **processus non stationnaire en moyenne ou non stationnaire du premier ordre**, noté NSM, est un processus aléatoire  $(X_t)_{t \in T}$  tel qu'il existe une fonction déterministe (non aléatoire) non constante  $\mu \in \mathbb{R}^T$  tel que, pour tout  $t \in T$ ,  $\mathbb{E}[X_t] = \mu(t)$ .

#### 12.1.1 Non stationnarité déterministe ou stationnarité à une tendance près (*Trend Stationnary*)

#### Définition (Processus *Trend Stationnary*)

Un **processus *Trend Stationnary***, noté TS, est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe une fonction déterministe  $\mu \in \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $X_t = \mu(t) + B_t$ .

**Exemple** Modèle additif

#### 12.1.1.1 Processus à tendance déterministe

#### Définition (Processus à tendance déterministe)

Un **processus à tendance déterministe** est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe une fonction déterministe (sans composante périodique)  $m \in \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $X_t = m(t) + B_t$ .

#### Définition (Processus à tendance déterministe)

Un **processus à tendance déterministe** est un processus *Trend Stationnary* (TS) dont la fonction déterministe ne présente pas de composante saisonnière.

La variance d'un processus à tendance déterministe est stable dans le temps.

**Exemple**  $X_t = e^{\frac{t}{30}} + (\mathcal{E}_t + \mathcal{E}_{t-1} + 1, 6 \mathcal{E}_{t-2})$

**Exemple** Processus à tendance déterministe périodique (variance stable)

#### 12.1.1.1.1 Processus à tendance déterministe polynomiale

**Définition (Processus à tendance déterministe polynomiale)**

Un **processus à tendance déterministe polynomiale** est un processus à tendance déterministe donc la fonction déterministe est polynomiale.

**Définition (Processus à tendance déterministe polynomiale)**

Un **processus à tendance déterministe polynomiale** est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe une fonction déterministe polynomiale  $m \in \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $X_t = m(t) + B_t$ .

**12.1.1.1.1 Processus à tendance déterministe linéaire****Définition (Processus à tendance déterministe linéaire)**

Un **processus à tendance déterministe linéaire** est un processus à tendance déterministe (polynomiale) donc la fonction déterministe est linéaire.

**Définition (Processus à tendance déterministe linéaire)**

Un **processus à tendance déterministe linéaire** est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe une fonction déterministe linéaire  $m \in \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $X_t = m(t) + B_t$ .

On peut reformuler cette définition sous la forme équivalente suivante.

Un **processus à tendance déterministe linéaire** est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe  $(a, b) \in \mathbb{R}^2$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $X_t = at + b + B_t$ .

**Exemple** Soit  $(a, b, \sigma) \in \mathbb{R}^3$ ,  $(\mathcal{E}_t)_{t \in T}$  un processus stochastique tel que  $\mathcal{E}_t \simeq BB(0, \sigma^2)$ .

Le processus  $(at + b + \mathcal{E}_t)_{t \in T}$  est un processus à tendance déterministe linéaire.

On a

$$\mathbb{E}[a + bt + \mathcal{E}_t] = at + b$$

Donc  $(at + b + \mathcal{E}_t)_{t \in T}$  est un processus non stationnaire en moyenne (ou non stationnaire du premier ordre). De plus,

$$\text{Var}(at + b + \mathcal{E}_t) = \sigma^2 \quad \text{et} \quad \forall h \in T^* \quad \text{Cov}(at + b + \mathcal{E}_t, a(t+h) + b + \mathcal{E}_{t+h}) = 0$$

On en déduit que  $(at + b + \mathcal{E}_t)_{t \in T}$  est stationnaire en covariance, de covariance constante.

**12.1.1.1.2 Processus à tendance déterministe périodique****Définition (Processus à tendance déterministe périodique)**

Un **processus à tendance déterministe périodique** est un processus à tendance déterministe donc la fonction déterministe est périodique.

**Définition (Processus à tendance déterministe périodique)**

Un **processus à tendance déterministe périodique** est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe une fonction déterministe périodique  $m \in \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $X_t = m(t) + B_t$ .

**Exemple** Soit  $\sigma \in \mathbb{R}$ ,  $(\mathcal{E}_t)_{t \in T}$  un processus stochastique tel que  $\mathcal{E}_t \sim BB(0, \sigma^2)$ .

Le processus  $(\cos(\frac{2\pi t}{12}) + \mathcal{E}_t)_{t \in T}$  est un processus à tendance déterministe périodique.

On a

$$\mathbb{E} \left[ \cos \left( \frac{2\pi t}{12} \right) + \mathcal{E}_t \right] = \cos \left( \frac{2\pi t}{12} \right)$$

Donc  $(\cos(\frac{2\pi t}{12}) + \mathcal{E}_t)_{t \in T}$  est un processus non stationnaire en moyenne (ou non stationnaire du premier ordre). De plus,

$$\text{Var} \left( \cos \left( \frac{2\pi t}{12} \right) + \mathcal{E}_t \right) = \sigma^2 \quad \text{et} \quad \forall h \in T^* \quad \text{Cov} \left( \cos \left( \frac{2\pi t}{12} \right) + \mathcal{E}_t, \cos \left( \frac{2\pi(t+h)}{12} \right) + \mathcal{E}_{t+h} \right) = 0$$

On en déduit que  $(\cos(\frac{2\pi t}{12}) + \mathcal{E}_t)_{t \in T}$  est stationnaire en covariance.

Stationnariser par méthode des moindres carrés ordinaires (=régression sur le temps)

### 12.1.1.2 Processus à saisonnalité déterministe

#### Définition (Processus à saisonnalité déterministe)

Un **processus à saisonnalité déterministe** est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe  $c \in \mathbb{R}$ ,  $s \in \mathbb{R}^T$  une fonction périodique déterministe et  $(B_t)_{t \in T}$  un processus stationnaire centré vérifiant

$$\forall t \in T \quad X_t = c + s(t) + B_t$$

#### Définition (Processus à saisonnalité déterministe)

Un **processus à saisonnalité déterministe** est un processus *Trend Stationnary* (TS) dont la fonction déterministe est périodique et ne présente pas de composante tendancielle.

#### Définition (Processus à saisonnalité déterministe de période $r$ )

Un **processus à saisonnalité déterministe de période  $r$**  est un processus stochastique  $(X_t)_{t \in T}$  dont la fonction déterministe est périodique de période  $r$ .

On peut reformuler cette définition sous la forme équivalente suivante.

Un **processus à saisonnalité déterministe de période  $r$**  est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe  $(r, c, s_1, \dots, s_r) \in \mathbb{N}^* \times \mathbb{R}^{r+1}$  et  $(B_t)_{t \in T}$  un processus stationnaire centré vérifiant

$$\forall t \in T \quad X_t = c + \sum_{i=1}^r s_i \chi_{i+\mathbb{Z}r}(t) + B_t$$

ou encore

$$\forall t \in T \quad X_t = c + \sum_{i=1}^r s_i \chi_{\{k \in \mathbb{Z} \mid k \equiv i[r]\}}(t) + B_t$$

**Convention** Par convention on suppose que

$$\sum_{i=1}^r s_i \chi_{i+\mathbb{Z}r}(t) + B_t = \sum_{i=1}^r s_i \chi_{\{k \in \mathbb{Z} \mid k \equiv i[r]\}}(t) + B_t = 0$$

#### Proposition (Processus à saisonnalité déterministe)

Tout processus à saisonnalité déterministe est un processus *Seasonally Difference Stationnary* (SDS).

Soit  $(X_t)_{t \in T}$  un processus à saisonnalité déterministe. Par définition, il existe  $c \in \mathbb{R}$ ,  $s \in \mathbb{R}^T$  une fonction périodique déterministe et  $(B_t)_{t \in T}$  un processus stationnaire centré vérifiant

$$\forall t \in T \quad X_t = c + s(t) + B_t$$

On a

$$\begin{aligned}
\Delta_r X_t &= \Delta_r(c + s(t) + B_t) \\
&= \Delta_r(c) + \Delta_r(s(t)) + \Delta_r(B_t) \\
&= s(t) - s(t-r) + \Delta_r(B_t) \\
&= s(t) - s(t) + \Delta_r B_t \\
&= \Delta_r B_t
\end{aligned}$$

Le processus  $(\Delta_r B_t)_{t \in T}$  est stationnaire centré donc le processus  $(\Delta_r X_t)_{t \in T}$  est stationnaire. On en conclut que le processus  $(X_t)_{t \in T}$  est un processus *Seasonally Differencial Stationnary* (SDS).

**Exemple** Processus à composante (périodique) trigonométrique bruitée.

Considérons le processus  $(\cos(\frac{2\pi t}{12}) + \mathcal{E}_t)_{t \in T}$  avec  $\mathcal{E}_t \sim BB(0, \sigma^2)$ .

Le processus est TS.

On a

$$\Delta_{12} \left( \cos \left( \frac{2\pi t}{12} \right) + \mathcal{E}_t \right) = \Delta_{12} \mathcal{E}_t = \mathcal{E}_t - \mathcal{E}_{t-12}$$

Le processus  $(\mathcal{E}_t - \mathcal{E}_{t-12})_{t \in T}$  est un processus MA(12) centré donc stationnaire. On en déduit que  $(\cos(\frac{2\pi t}{12}) + \mathcal{E}_t)_{t \in T}$  avec  $\mathcal{E}_t \sim BB(0, \sigma^2)$  est un processus *Seasonally Difference Stationary* (SDS).

De plus il est NSM mais de covariance constante.

Il existe des processus à composante périodique non trigonométriques. Ex Processus à composante périodique triangulaire.

## 12.2 Non stationnarité du deuxième ordre ou en variance NSV

### Définition (Processus non stationnaire en variance ou non stationnaire du deuxième ordre)

Un **processus non stationnaire en variance** ou **processus non stationnaire du deuxième ordre**, noté NSV, est un processus aléatoire  $(X_t)_{t \in T}$  tel qu'il existe une fonction déterministe (non aléatoire) non constante  $\sigma \in \mathbb{R}^T$  tel que, pour tout  $t \in T$ ,  $\text{Var}(X_t) = \sigma^2(t)$ .

### 12.2.1 Non stationnarité stochastique ou stationnarité en différences (*Difference Stationnary*)

#### Définition (Processus *Difference Stationnary*)

Un **processus *Difference Stationnary***, noté DS, est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe  $(d, c) \in \mathbb{N}^* \times \mathbb{R}$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $\Delta^d X_t = c + B_t$  ou encore  $(1 - L)^d X_t = c + B_t$ .

#### Définition (Processus *Difference Stationnary*)

Un **processus *Difference Stationnary***, noté DS, est un processus stochastique tel qu'il existe  $d \in \mathbb{N}^*$  et qui différencié  $d$  fois conduit à un processus stationnaire.

#### 12.2.1.1 Processus à tendance stochastique

##### Définition (Processus à tendance stochastique)

Un **processus à tendance stochastique** est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe une fonction déterministe  $m \in \mathbb{R}^T$  présentant une composante saisonnière et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $X_t = m(t) + B_t$ .

##### Définition (Processus à tendance stochastique)

Un **processus à tendance stochastique** est un processus *Trend Stationnary* (TS) qui n'est pas à tendance déterministe *i.e.* dont la fonction déterministe présente une composante saisonnière.

La variance d'un processus à tendance stochastique semble être non bornée.

#### 12.2.1.1.1 Marche aléatoire

**Exemple** Soit  $\sigma \in \mathbb{R}$  et  $(\mathcal{E}_t)_{t \in T}$  un processus stochastique tel que  $\mathcal{E}t \simeq BB(0, \sigma^2)$ .

Le processus  $(X_t)_{t \in \mathbb{N}}$  tel que

$$\forall t \in T \quad X_t = \begin{cases} X_{t-1} + \mathcal{E}_t & \text{si } t \in \mathbb{N}^* \\ 0 & \text{si } t = 0 \end{cases}$$

*i.e.* le processus  $(\mathcal{E}_1 + \dots + \mathcal{E}_t)_{t \in T}$  est une marche aléatoire.

On a

$$\mathbb{E}[\mathcal{E}_1 + \dots + \mathcal{E}_t] = 0$$

Donc le processus  $(\mathcal{E}_1 + \dots + \mathcal{E}_t)_{t \in T}$  est stationnaire en moyenne. Par ailleurs, la fonction d'autocovariance est donnée par

$$\forall (t_1, t_2) \in T^2 \quad \text{Cov}(\mathcal{E}_1 + \dots + \mathcal{E}_{t_1}, \mathcal{E}_1 + \dots + \mathcal{E}_{t_2}) = \mathbb{E}[(\mathcal{E}_1 + \dots + \mathcal{E}_{t_1})(\mathcal{E}_1 + \dots + \mathcal{E}_{t_2})] = \min(t_1, t_2)\sigma^2$$

Donc  $\text{Var}(\mathcal{E}_1 + \dots + \mathcal{E}_t) = t\sigma^2$  dépend de  $t$ , c'est-à-dire que  $(\mathcal{E}_1 + \dots + \mathcal{E}_t)_{t \in T}$  est non stationnaire en variance.

## 12.2.2 Seasonnaly Difference Stationnary (SDS)

### Définition (Processus Seasonnaly Difference Stationnary)

Un processus *Seasonnaly Difference Stationnary*, noté SDS, est un processus stochastique  $(X_t)_{t \in T}$  tel qu'il existe  $(r, D, c) \in \mathbb{N}^{*2} \times \mathbb{R}$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $\Delta_r^D X_t = c + B_t$ .

### Définition (Processus Seasonnaly Difference Stationnary)

Un processus *Seasonnaly Difference Stationnary*, noté SDS, est un processus stochastique tel qu'il existe  $D \in \mathbb{N}^*$  et qui différencié saisonnièrement  $D$  fois conduit à un processus stationnaire.

### 12.2.2.1 Processus à saisonnalité stochastique

#### Définition (Processus à saisonnalité stochastique)

Un processus à saisonnalité stochastique est un processus stochastique  $(X_t)_{t \in T}$  tel :

- qu'il existe  $(r, D, c) \in \mathbb{N}^{*2} \times \mathbb{R}$  et un processus stationnaire centré  $(B_t)_{t \in T}$  vérifiant, pour tout  $t \in T$ ,  $\Delta_r^D X_t = c + B_t$ ,
- que pour tout  $c' \in \mathbb{R}$  ? POUR toute fonction périodique déterministe  $s' \in \mathbb{R}^T$  et pour tout processus stationnaire centré  $(B'_t)_{t \in T}$ , il existe  $t \in T$  vérifiant  $X_t \neq c' + s'(t) + B'_t$ .

#### Définition (Processus à saisonnalité stochastique)

Un processus à saisonnalité stochastique est un processus *Seasonnaly Difference Stationary* (SDS) qui n'est pas à saisonnalité déterministe.

#### Définition (Saisonnalité d'un processus à saisonnalité stochastique)

Soit  $(X_t)_{t \in T}$  un processus à saisonnalité stochastique.

La **saisonnalité** de  $(X_t)_{t \in T}$  est l'élément  $r \in \mathbb{N}^*$  tel qu'il existe  $(D, c) \in \mathbb{N}^* \times \mathbb{R}$  et  $(B_t)_{t \in T}$  un processus stationnaire centré vérifiant

$$\forall t \in T \quad \Delta_r^D X_t = c + B_t$$

**Exemple** Soit  $\sigma \in \mathbb{R}$  et  $(\mathcal{E}_t)_{t \in T}$  un processus stochastique tel que  $\mathcal{E}t \simeq BB(0, \sigma^2)$ .

Le processus  $(X_t)_{t \in \mathbb{N}}$  tel que

$$\forall t \in T \quad X_t = \begin{cases} X_{t-12} + \mathcal{E}_t & \text{si } t \in \mathbb{N}^* \\ 0 & \text{si } t = 0 \end{cases}$$

i.e. le processus  $(\mathcal{E}_k + \dots + \mathcal{E}_{t-12} + \mathcal{E}_t)_{t \in T}$  avec  $k \equiv t[12]$  est une marche aléatoire.

On a

$$\mathbb{E}[X_t] = 0$$

Donc le processus  $(X_t)_{t \in T}$  est stationnaire en moyenne. Par ailleurs, la fonction d'autocovariance est donnée par

$$\forall (t_1, t_2) \in T^2 \quad \text{Cov}(X_{t_1}, X_{t_2}) = \begin{cases} \mathbb{E}[X_{t_1} X_{t_2}] = \min(t_1, t_2) \sigma^2 & \text{si } t_1 \equiv t_2[12] \\ 0 & \text{sinon} \end{cases}$$

Donc  $\text{Var}(X_t) = t \sigma^2$  dépend de  $t$ , c'est-à-dire que  $(X_t)_{t \in T}$  est non stationnaire en variance.

### 12.2.3 Variance inconditionnelle variante (VIV)

**Définition (Processus à variance inconditionnelle variante (VIV))**

Un **processus à variance inconditionnelle variante**, noté VIV, est un processus stochastique tel qu'il existe une fonction déterministe  $\sigma \in \mathbb{R}^T$  et un processus stationnaire  $(Y_t)_{t \in T}$  vérifiant, pour tout  $t \in T$ ,  $X_t = \sigma(t)Y_t$ .

La modélisation par un processus VIV revient à estimer la fonction déterministe  $\sigma$  et à ajuster un modèle ARMA à  $(Y_t)_{t \in T}$ . Les processus VIV ont été utilisés par Starica et Granger pour modéliser les effets hétéroscédastiques.

**Exemple** Soit  $(\mathcal{E}_t)_{t \in T}$  un bruit blanc. Considérons la fonction suivante.

$$\begin{aligned} \sigma : T &\longrightarrow \mathbb{R} \\ t &\longmapsto \begin{cases} 1 & \text{si } 0 \leq t < 200 \\ 4 & \text{si } 200 \leq t < 400 \\ 1 & \text{si } t > 400 \end{cases} \end{aligned}$$

Le processus  $(\sigma(t)\mathcal{E}_t)_{t \in T}$  est un processus VIV.

## Chapitre 13

# Test de non stationnarité ou test de racine unitaire

De nombreux tests statistiques ont été construits afin de décider si la série temporelle étudiée est stationnaire ou non. Une première approche consiste à considérer l'hypothèse de présence d'une racine unitaire, ce sont les tests de non stationnarité.

### 13.1 Test de racine unitaire non saisonnière

#### 13.1.1 Test de Dickey-Fuller (DF)

##### 13.1.1.1 Test de Dickey-Fuller augmenté (ADF)

#### 13.1.2 Test de Phillips-Perron (PP)

#### 13.1.3 Test d'Elliott-Rothenberg-Stock (ERS)

### 13.2 Test de racine unitaire saisonnière

Nous allons étudier les processus avec saisonnalité et tester l'existence d'une racine unitaire saisonnière. Les tests présentés ici peuvent être interprétés comme équivalents du test de Dickey-Fuller augmenté pour les racines unitaires, mais adaptés aux données saisonnières.

#### 13.2.1 Test de Canova-Hansen (CH)

#### 13.2.2 Test de Osborn-Chui-Smith-Birchenhall (OCSB)

# Chapitre 14

## Tendance

### 14.1 Détection d'une composante tendancielle

On observe une décroissance très lente de la fonction d'autocorrélation empirique  $\hat{\rho}$ . Le comportement théorique de la fonction d'autocorrélation théorique  $\rho$  pour des processus avec tendance permet de détecter une tendance, mais sans identifier la nature de cette tendance.

Le chronogramme d'une série permet de détecter facilement une tendance si celle-ci est déterministe. En revanche, une tendance stochastique est plus difficilement identifiable à partir d'une trajectoire. Tracer la fonction d'autocorrélation empirique permet de s'assurer qu'on est bien en présence d'une tendance.

#### Proposition (Condition nécessaire à une tendance déterministe polynomiale pure)

Soit  $(X_t)_{t \in T}$  un processus stochastique.

Si  $(X_t)_{t \in T}$  est un processus à tendance déterministe polynomiale pure *i.e.* tel qu'il existe une fonction polynomiale  $m \in \mathbb{R}^T$  vérifiant, pour tout  $t \in T$ ,  $X_t = m(t)$ , alors :

- La fonction d'autocorrélation empirique  $\hat{\rho}$  de  $(X_t)_{t \in T}$  est décroissante.
- $\forall h \neq 0 \quad \hat{\rho}(h) \xrightarrow[n \rightarrow +\infty]{} 1$

#### Proposition (Condition nécessaire à une tendance déterministe polynomiale pure)

- La fonction d'autocorrélation empirique de tout processus à tendance déterministe polynomiale pure est décroissante.
- $\forall h \neq 0 \quad \hat{\rho}(h) \xrightarrow[n \rightarrow +\infty]{} 1$

#### Proposition ()

Soit  $(X_t)_{t \in T}$  un processus stochastique.

S'il existe  $(d, \delta) \in \mathbb{N}^{*2}$  et un bruit blanc centré  $(\mathcal{E}_t)_{t \in T}$  tel que

$$\begin{cases} \forall t \in T \quad t \leq 0 \implies X_t = 0 \\ \forall t \in T \quad \Delta^d(X_t) = \mathcal{E}_t \\ \sup_{1 \leq t \leq n} \mathbb{E} \left( |\mathcal{E}_t|^{2+\delta} \right) < +\infty \end{cases}$$

alors

$$\forall h \neq 0 \quad \hat{\Xi}(h) \xrightarrow[n \rightarrow +\infty]{\mathbb{P}} 1$$

## 14.2 Identification de la nature de la tendance

Pour identifier la nature de la tendance on réalise successivement les 3 tests suivants : un test de racine unitaire ADF (respectivement PP) sur la série initiale puis sur la série différenciée et le test KPSS de stationnarité sur la série différenciée. On répondra *Non* au test si on rejette  $H_0$  et *Oui* sinon.

### 14.2.1 Méthode d'identification avec les tests ADF et KPSS

On réalise successivement les 3 tests suivants :

- test ADF sur la série initiale
- test ADF sur la série différenciée
- test KPSS sur la série différenciée

Réponses aux tests	Nature de la tendance
Non Non Oui	Tendance linéaire
Oui Non Non	Tendance quadratique
Oui Non Oui	Tendance stochastique d'ordre 1
Oui Oui Non ou Oui Oui Oui ou Non Oui Non ou Non Oui Oui	Tendance stochastique d'ordre 2

TABLE 14.1 – Tableau d'interprétation des résultats aux tests ADF et KPSS

### 14.2.2 Méthode d'identification avec les tests PP et KPSS

On réalise successivement les 3 tests suivants :

- test PP sur la série initiale
- test PP sur la série différenciée
- test KPSS sur la série différenciée

Réponses aux tests	Nature de la tendance
Non Non Oui	Tendance linéaire
Oui Non Non	Tendance quadratique
Oui Non Oui	Tendance stochastique d'ordre 1
Oui Oui Non ou Oui Oui Oui	Tendance stochastique d'ordre 2

TABLE 14.2 – Tableau d'interprétation des résultats aux tests PP et KPSS

## 14.3 Modélisation d'une tendance déterministe : processus ARMAX

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

Modélisons le processus  $(X_t)_{t \in T}$  par une tendance déterministe. Supposons donc qu'il existe une fonction déterministe (sans composante périodique)  $m \in \mathbb{R}^T$  et un processus stationnaire centré  $(\mathcal{E}_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $X_t = m(t) + B_t$ .

La modélisation se fait en deux étapes.

- Modélisation de la fonction déterministe. Dans la pratique on estime généralement par une régression paramétrique ou non-paramétrique. Le plus souvent on procède par régression (paramétrique) multiple en estimant  $m$  par une fonction polynomiale.
- Modélisation de la série résiduelle par un processus stationnaire.

On valide ensuite le modèle par test de stationnarité des résidus.

### 14.3.1 Par régression linéaire

#### 14.3.1.1 Modélisation de la fonction déterministe

Lorsqu'on modélise la tendance d'une série par une régression linéaire, une difficulté consiste à identifier le degré du polynôme qui décrit la tendance. Plusieurs méthodes sont possibles.

- Différencier successivement la série et détecter (à l'aide du chronogramme et de l'acf) le moment où la tendance a été supprimée.
- Construire les modèles correspondant aux degrés 1, 2, ... et les comparer (à l'aide du  $R^2_{\text{ajusté}}$  et de la variance résiduelle, mais pas à l'aide des critères tels que AIC car ils supposent les 4 conditions précédentes vérifiées.

#### 14.3.1.2 Modélisation de la série résiduelle

Le modèle linéaire requiert que  $(B_t)_{t \in T}$  soit un bruit blanc *i.e.* que les conditions suivantes sur  $(B_t)_{t \in T}$  soient vérifiées.

- **Erreur centrée**  $\mathbb{E}[B_t] = 0$
- **Homoscédasticité**  $\exists \sigma \in \mathbb{R} \quad \text{Var}(B_t) = \sigma^2$
- **Non autocorrélation**  $\forall (t_1, t_2) \in T^2 \quad t_1 \neq t_2 \implies \text{Cov}(t_1, t_2) = 0$

Pour des échantillons petits, il peut être nécessaire de supposer que ce bruit blanc soit gaussien *i.e.* que :

- **Normalité** Pour tout  $t \in T$ ,  $B_t$  suit une loi normale.

Dans le cadre de la régression linéaire, on n'utilise pas les tests de bruit blanc. On utilise :

- le test de Breusch-Godfrey pour la non autocorrélation,
- le test de Harrison-McCabe pour l'homoscédasticité.

Si la série résiduelle  $(X_t - \hat{m}(t))_{t \in T}$  suit les conditions usuelles du modèle linéaire, alors on utilise le modèle fourni par la régression pour réaliser les prévisions. Généralement les conditions du modèle linéaire ne sont pas vérifiées et on doit affiner le modèle en modélisant les résidus par un ARMA. On parle alors de modèle ARMAX.

**Proposition (Élimination d'une tendance déterministe polynomiale)**

Tout processus à tendance déterministe polynomiale est un processus *Differencial Stationnary* (DS).

Soit  $(X_t)_{t \in T}$  un processus à tendance déterministe polynomiale. Par définition, il existe une fonction polynomiale  $m \in \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que, pour tout  $t \in T$ ,  $X_t = m(t) + B_t$ . Notons  $d$  le degré de  $m$  et  $\alpha_d$  le coefficient dominant de  $m$ .

On a

$$\Delta^d(X_t) = \Delta^d(m(t) + B_t) = \Delta^d(m(t)) + \Delta^d(B_t) = d! \alpha_d + \Delta^d B_t$$

Le processus  $(\Delta^d(B_t))_{t \in T}$  est stationnaire centré donc le processus  $(\Delta^d(X_t))_{t \in T}$  est stationnaire. On en conclut que le processus  $(X_t)_{t \in T}$  est un processus *Differencial Stationnary* (DS).

Soit  $(d, \alpha_d) \in \mathbb{N}^* \times \mathbb{R}$ ,  $m \in \mathbb{R}^T$  une fonction polynomiale de degré  $d$  et de coefficient dominant  $\alpha_d$  et  $(B_t)_{t \in T}$  un processus stationnaire centré de sorte que  $(m(t) + b_t)_{t \in T}$  soit un processus à tendance déterministe polynomiale.

**Exemple** Soit  $(a, b, \sigma) \in \mathbb{R}^3$ ,  $(\mathcal{E}_t)_{t \in T}$  un processus stochastique tel que  $\mathcal{E}_t \simeq BB(0, \sigma^2)$ .

Le processus  $(at + b + \mathcal{E}_t)_{t \in T}$  est un processus à tendance déterministe linéaire. On a

$$\forall t \in T \quad \Delta(at + b + \mathcal{E}_t) = \Delta(at) + \Delta(b) + \Delta(\mathcal{E}_t) = a + \mathcal{E}_t - \mathcal{E}_{t-1}$$

Le processus  $(\mathcal{E}_t - \mathcal{E}_{t-1})_{t \in T}$  est un processus MA(1) centré donc stationnaire. On en déduit que  $(at + b + \mathcal{E}_t)_{t \in T}$  est un processus *Difference Stationnary* (DS).

**Remarque**  $TS \cap DS \neq \emptyset$

## 14.4 Modélisation d'une tendance stochastique : processus ARIMA

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

Modélisons le processus  $(X_t)_{t \in T}$  par une tendance stochastique. Supposons donc qu'il existe  $(d, c) \in \mathbb{N}^* \times \mathbb{R}$  et  $(B_t)_{t \in T}$  un processus stationnaire centré tel que

$$\forall t \in T \quad \Delta^d(X_t) = c + B_t$$

La modélisation se fait en deux étapes.

- Estimation du degré d'intégration  $d$ , par différenciations successives et tests de stationnarité.
- Modélisation de la série résiduelle par un processus stationnaire. Dans la pratique on modélise généralement  $(B_t)_{t \in T}$  par un processus ARMA ce qui définit les processus ARIMA.

## 14.5 Exemples de modélisation d'une tendance

Considérons le jeu de données `airquality` du logiciel **R**.

```
data(airquality) # class(airquality) "data.frame"
```

Avant d'afficher les données il convient de prendre en compte les dimension du jeu de données.

```
dim(airquality)
```

```
## [1] 153 6
```

L'affichage de l'ensemble des données nécessite plusieurs pages. Affichons donc uniquement les 6 premières et 6 dernières observations.

```
head(airquality) # ou airquality[1:6,], class "data.frame"
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```
tail(airquality) # class data.frame
```

```
##   Ozone Solar.R Wind Temp Month Day
## 148    14     20 16.6   63     9  25
## 149    30    193  6.9   70     9  26
## 150    NA    145 13.2   77     9  27
## 151    14    191 14.3   75     9  28
## 152    18    131  8.0   76     9  29
## 153    20    223 11.5   68     9  30
```

Extrayons la série temporelle `Temp`. Cette série temporelle quotidienne représente la température mesurée à New-York entre le 1<sup>er</sup> mai et le 30 septembre 1973. Affichons uniquement les 6 premières et 6 dernières observations.

```
Temp <- airquality$Temp # class "integer"
```

```
## Autres méthodes, indicées
```

```
# Temp <- airquality[4]
```

```
# Temp <- airquality[[4]]
```

```
## Autres méthodes, par nom
```

```
# Temp <- airquality[, "Temp"]
```

```
# Temp <- airquality[["Temp"]]
```

```
head(Temp) # class "integer"
```

```
## [1] 67 72 74 62 56 66
```

```
tail(Temp) # class "integer"
```

```
## [1] 63 70 77 75 76 68
```

Remarquez la différence avec les commandes suivantes.

```
Temp_df <- airquality[4] # class "data.frame"
```

```
## Autre méthode, par nom
```

```
# Temp_df <- airquality["Temp"]
```

```
head(Temp_df) # class "data.frame"
```

```
##      Temp
## 1      67
## 2      72
## 3      74
## 4      62
## 5      56
## 6      66

tail(Temp_df) # class "data.frame"

##      Temp
## 148     63
## 149     70
## 150     77
## 151     75
## 152     76
## 153     68
```

Cette ne présente aucune valeur manquante.

```
which(is.na(Temp), arr.ind=TRUE)

## integer(0)
```

Cette série est constituée de 153 observations.

```
length(Temp)

## [1] 153
```

La série n'est pas de classe `ts`.

```
class(Temp)

## [1] "integer"
```

Convertissons la en classe `ts`.

```
Temp <- as.ts(Temp)
```

Posons  $T = \{1, \dots, 153\}$  et, pour tout  $t \in T$ ,  $X_t := \text{Temp}[t]$ . On cherche donc à modéliser la série temporelle  $(X_t)_{t \in T}$ .

Traçons le chronogramme de cette série.

```
plot(Temp, ylab="Température", main="Chronogramme de la série Temp")
```

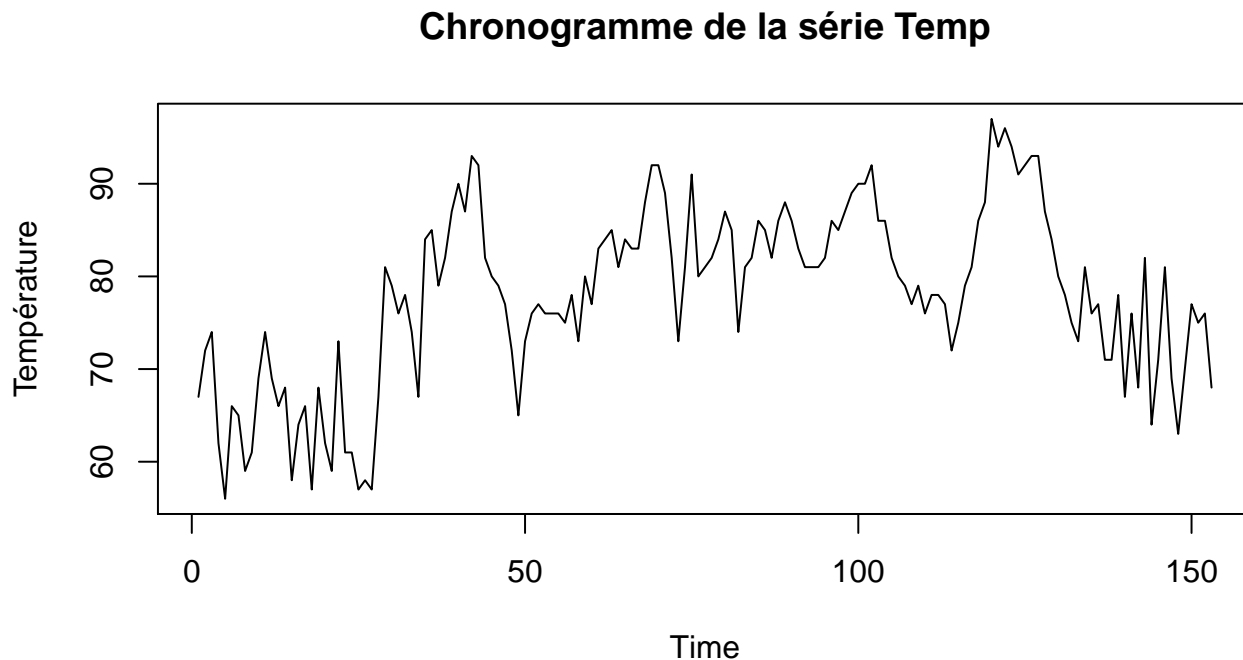


FIGURE 14.1 – Chronogramme de la série Temp

La série n'est pas périodique ou présente moins de deux périodes donc on ne peut pas tracer sa décomposition usuelle avec la commande suivante.

```
plot(decompose(Temp))
```

Effectuons un test de stationnarité.

```
library(tseries)

## Registered S3 method overwritten by 'quantmod':
## method      from
## as.zoo.data.frame zoo

kpss.test(Temp) # class "htest"

## Warning in kpss.test(Temp): p-value smaller than printed p-value

##
## KPSS Test for Level Stationarity
##
## data: Temp
## KPSS Level = 0.93715, Truncation lag parameter = 4, p-value = 0.01

# Extraction de la p-valeur
names(kpss.test(Temp))

## Warning in kpss.test(Temp): p-value smaller than printed p-value
```

```
## [1] "statistic" "parameter" "p.value" "method" "data.name"
kpss.test(Temp)$p.value
## Warning in kpss.test(Temp): p-value smaller than printed p-value
## [1] 0.01
```

On a `kpss.test(Temp)$p.value = 0.01 < 0.05` donc le test KPSS rejette la stationnarité de la série `Temp`.

### 14.5.1 Détection d'une tendance

Le chronogramme ne met pas en évidence de tendance. Traçons la fonction d'autocorrélation empirique.

```
acf(Temp)
```

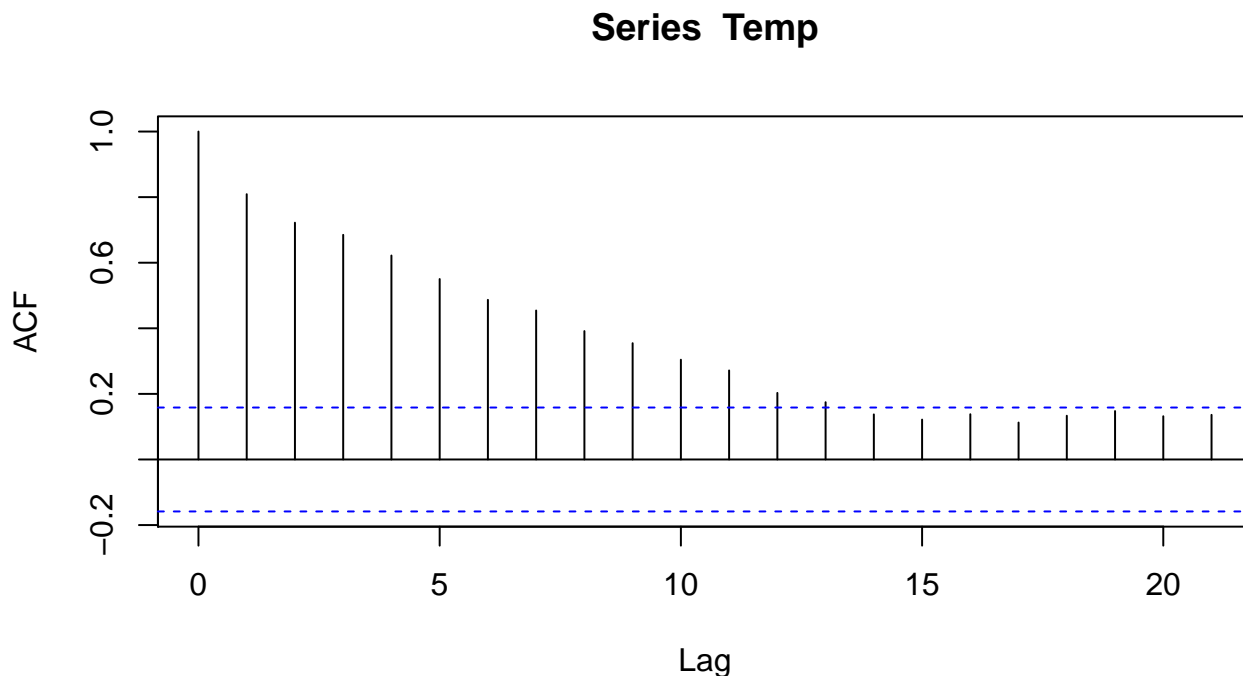


FIGURE 14.2 – Autocorrélogramme empirique de la série `Temp`

La tendance est détectée grâce à la fonction d'autocorrélation empirique.

### 14.5.2 Identification de la nature de la tendance

```
adf.test(Temp)
##
## Augmented Dickey-Fuller Test
##
## data: Temp
## Dickey-Fuller = -2.4041, Lag order = 5, p-value = 0.408
## alternative hypothesis: stationary
```

```

adf.test(diff(Temp))

## Warning in adf.test(diff(Temp)): p-value smaller than printed p-value
##
## Augmented Dickey-Fuller Test
##
## data: diff(Temp)
## Dickey-Fuller = -6.2228, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary

kpss.test(diff(Temp))

## Warning in kpss.test(diff(Temp)): p-value greater than printed p-value
##
## KPSS Test for Level Stationarity
##
## data: diff(Temp)
## KPSS Level = 0.084547, Truncation lag parameter = 4, p-value = 0.1

```

La réponse aux tests est *Oui, Non, Oui* donc une modélisation avec une tendance stochastique d'ordre 1 est préconisée.

### 14.5.3 Exemple de modélisation d'une tendance déterministe

Contrairement à la préconisation précédente et dans le but d'illustrer la méthode, modélisons la série par une tendance déterministe. Supposons donc qu'il existe une fonction déterministe (sans composante périodique)  $m \in \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad X_t = m(t) + B_t$$

#### 14.5.3.1 Modélisation de la fonction déterministe

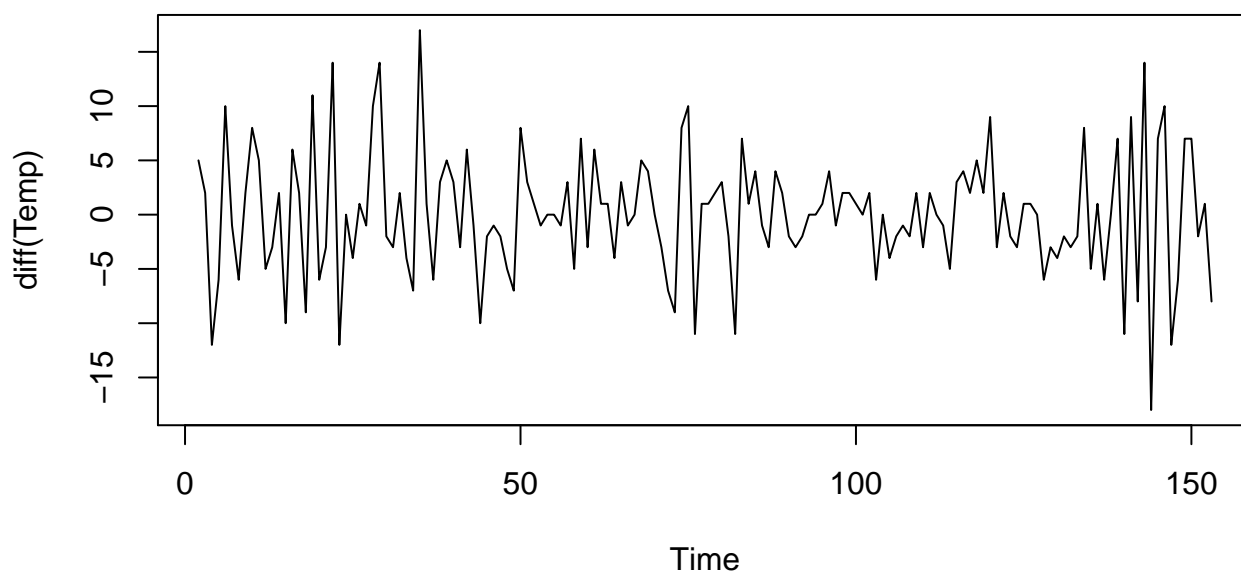
Définissons le degré du polynôme qui décrit la tendance  $m$ . Différencions successivement la série jusqu'à ce que la tendance disparaisse.

```

# Première différenciation
plot(diff(Temp), main="Chronogramme de la série diff(Temp)")
acf(diff(Temp))

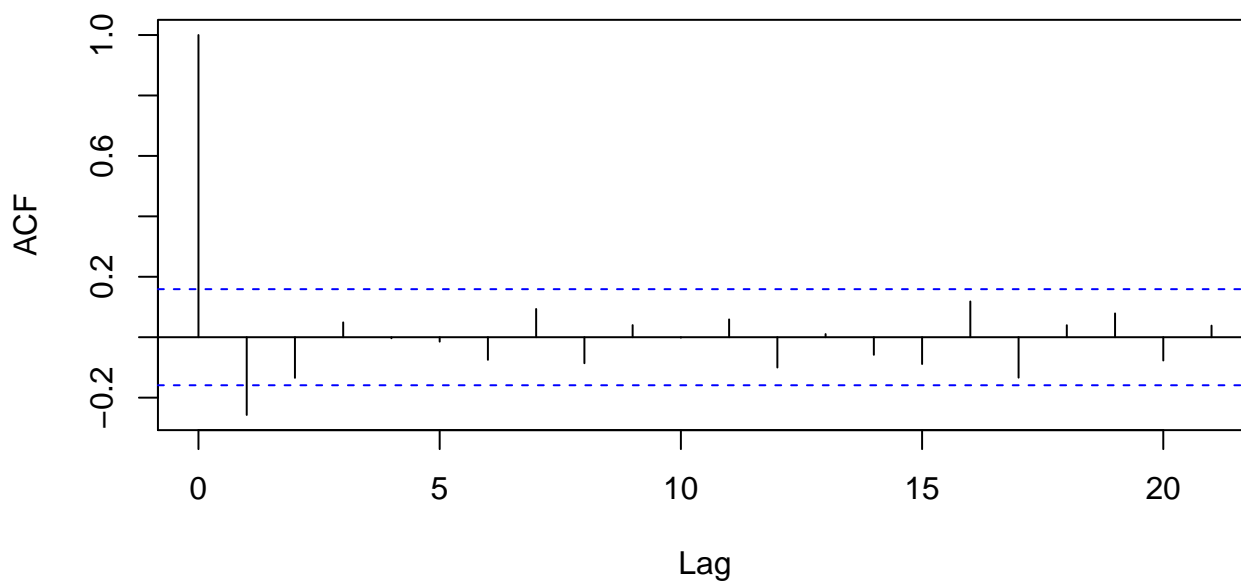
```

### Chronogramme de la série `diff(Temp)`



(a) Chronogramme de la série `diff(Temp)`

### Series `diff(Temp)`



(b) Autocorrélogramme de la série `diff(Temp)`

FIGURE 14.3 – Chronogramme et autocorrélogramme empirique de la série `diff(Temp)`

D'après la Figure 14.3, il semble ne plus rester de tendance dans la série différenciée. Un modèle de régression linéaire devrait donc suffire. Néanmoins, on calcule des modèles de régression de degré 1 et de degré 2.

#### 14.5.3.1.1 Par régression d'ordre 1

Explicitons notre modèle. On suppose qu'il existe une fonction affine  $m \in \mathbb{R}^T$  telle que :

$$\forall t \in T \quad X_t = m(t) + B_t$$

ou encore, il existe  $(\alpha_0, \alpha_1) \in \mathbb{R}^2$  tel que :

$$\forall t \in T \quad X_t = \alpha_0 + \alpha_1 t + B_t$$

```
temps <- 1:length(Temp) # class "integer"
head(temps) # class "integer"

## [1] 1 2 3 4 5 6

tail(temps) # class "integer"

## [1] 148 149 150 151 152 153
```

```
lm1 <- lm(Temp ~ temps)
lm1 # class "lm"

##
## Call:
## lm(formula = Temp ~ temps)
##
## Coefficients:
## (Intercept)      temps
##    71.53999      0.08237
```

On peut afficher uniquement les coefficients du modèle avec les commandes suivantes.

```
names(lm1)

## [1] "coefficients" "residuals" "effects" "rank"
## [5] "fitted.values" "assign" "qr" "df.residual"
## [9] "xlevels" "call" "terms" "model"

lm1$coefficients # ou plus simplement lm1$coef, class "numeric"

## (Intercept)      temps
## 71.53998968  0.08236835
```

#### 14.5.3.1.2 Par régression d'ordre 2

Explicitons notre modèle. On suppose qu'il existe une fonction du second degré  $m \in \mathbb{R}^T$  telle que :

$$\forall t \in T \quad X_t = m(t) + B_t$$

ou encore, il existe  $(\alpha_0, \alpha_1, \alpha_2) \in \mathbb{R}^3$  tel que :

$$\forall t \in T \quad X_t = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + B_t$$

**Remarque** Pour la régression d'ordre 2, il faut veiller à ne pas introduire d'opérateur de calcul (autre que le + qui représente la structure additive entre les variables explicatives) au sein de la fonction `lm()`. La commande suivante ne génère pas le modèle attendu.

```
lm(Temp ~ temps + temps^2)

##
## Call:
## lm(formula = Temp ~ temps + temps^2)
##
## Coefficients:
## (Intercept)      temps
##    71.53999      0.08237
```

Il faut procéder de la façon suivante.

```
temps2 <- temps^2 # class "numeric"
head(temps2) # class "numeric"

## [1] 1 4 9 16 25 36

tail(temps2) # class "numeric"

## [1] 21904 22201 22500 22801 23104 23409
```

```
lm2 <- lm(Temp ~ temps + temps2)
lm2 # class "lm"

##
## Call:
## lm(formula = Temp ~ temps + temps2)
##
## Coefficients:
## (Intercept)      temps      temps2
##  59.027098      0.566738     -0.003145
```

```
names(lm2)

## [1] "coefficients" "residuals"      "effects"      "rank"
## [5] "fitted.values" "assign"          "qr"           "df.residual"
## [9] "xlevels"      "call"           "terms"        "model"

lm2$coefficients # ou plus simplement lm2$coef, class "numeric"

## (Intercept)      temps      temps2
## 59.02709833  0.56673834 -0.00314526
```

Recalculons le modèle à l'aide de la fonction `Arima()` du package `forecast`, car la syntaxe est très simple pour calculer et tracer les prévisions.

```
library(forecast)
lm2b <- Arima(Temp, xreg=cbind(temps, temps2), method="CSS")
lm2b # class "forecast_ARIMA" "ARIMA" "Arima"

## Series: Temp
## Regression with ARIMA(0,0,0) errors
##
## Coefficients:
##      intercept      temps      temps2
##      59.0271      0.5667     -0.0031
```

```
## s.e.      1.6605  0.0498  0.0003
##
## sigma^2 estimated as 46.57:  part log likelihood=-509.42
```

On affiche directement les coefficients avec les commandes suivantes.

```
names(lm2b)

## [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
## [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
## [13] "nobs"      "model"     "aicc"      "bic"       "xreg"      "x"
## [19] "fitted"

lm2b$coef

## intercept      temps      temps2
## 59.02709833  0.56673834 -0.00314526
```

Les procédures `lm()` et `Arima()` produisent bien le même modèle.

#### 14.5.3.1.3 Comparaison des modèles de régression d'ordres 1 et 2

```
summary(lm1) # class "summary.lm"

##
## Call:
## lm(formula = Temp ~ temps)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.730  -6.034   1.177   6.553  18.000
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  71.53999     1.42373   50.248 < 2e-16 ***
## temps        0.08237     0.01604    5.136 8.55e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.762 on 151 degrees of freedom
## Multiple R-squared:  0.1487, Adjusted R-squared:  0.1431
## F-statistic: 26.37 on 1 and 151 DF, p-value: 8.554e-07

summary(lm2) # class "summary.lm"

##
## Call:
## lm(formula = Temp ~ temps + temps2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.0361  -4.4327  -0.3043   4.0580  15.7181
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  59.0270983  1.6770700   35.197 <2e-16 ***
## temps        0.5667383  0.0502788   11.272 <2e-16 ***
```

```
## temps2      -0.0031453  0.0003162  -9.946   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.825 on 150 degrees of freedom
## Multiple R-squared:  0.487, Adjusted R-squared:  0.4801
## F-statistic: 71.19 on 2 and 150 DF,  p-value: < 2.2e-16
```

Comparons les valeurs du coefficient de détermination linéaire de Pearson et de l'écart type.

```
names(summary(lm1))

## [1] "call"          "terms"          "residuals"      "coefficients"
## [5] "aliased"        "sigma"          "df"             "r.squared"
## [9] "adj.r.squared" "fstatistic"     "cov.unscaled"

# Affichage des valeurs
summary(lm1)$adj.r.squared

## [1] 0.1430535

summary(lm2)$adj.r.squared

## [1] 0.480146

summary(lm1)$sigma

## [1] 8.762133

summary(lm2)$sigma

## [1] 6.824545
```

Le modèle `lm2` présente un  $R^2$  supérieur et un  $\sigma$  inférieur à celui du `lm1`. Les critères favorisent donc la régression d'ordre 2. Étudions plus en détail le modèle `lm2`. Explicitons notre modèle. On suppose qu'il existe un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\begin{aligned} \forall t \in T \quad X_t &= \sum_{k=1}^3 \text{summary(lm2)\$coefficients}[k,1] t^{k-1} + B_t \\ &= 59.0271 + 0.5667t - 0.0031t^2 + B_t \end{aligned}$$

```
# Extraction des coefficients
summary(lm2)$coefficients # ou plus simplement summary(lm2)$coef, class "matrix"

##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) 59.02709833 1.677070042 35.196561 2.221057e-74
## temps       0.56673834 0.050278789 11.271917 9.849393e-22
## temps2      -0.00314526 0.000316248 -9.945548 3.254536e-18

summary(lm2)$coefficients[1,1] # class "numeric"

## [1] 59.0271
```

Remarquez les différences avec les commandes suivantes.

```
lm2$coefficients[1] # ou plus simplement lm2$coef[1], class "numeric"

## (Intercept)
##      59.0271
```

### 14.5.3.2 Analyse des résidus du modèle de régression d'ordre 2

```
residus <- lm2$residuals # ou plus simplement lm2$res
head(residus)

##           1           2           3           4           5           6
##  7.4093086 11.8520060 13.3009940  0.7562725 -5.7821585  3.6857010

tail(residus)

##          148          149          150          151          152          153
## -11.010605  -3.643202   3.730493   2.110478   3.496753  -4.110681
```

Les résidus ne sont pas de classe `ts`. Convertissons la série résiduelle en classe `ts`.

```
class(residus)

## [1] "numeric"

residus <- as.ts(residus)
```

Posons, pour tout  $t \in T$ ,  $B_t := \text{residus}[t]$ .

#### 14.5.3.2.1 Test de stationnarité des résidus

Testons si le processus obtenu est bien stationnaire.

```
kpss.test(residus) # class "htest"

##
## KPSS Test for Level Stationarity
##
## data:  residus
## KPSS Level = 0.04391, Truncation lag parameter = 4, p-value = 0.1

# Extaction de la p-valeur
names(kpss.test(residus))

## [1] "statistic" "parameter" "p.value" "method" "data.name"

kpss.test(residus)$p.value

## [1] 0.1
```

On a `kpss.test(residus)$p.value = 0.1 > 0.05` donc les résidus sont stationnaires. Par construction les résidus sont centrés. La modélisation de la série `Temp` par une tendance déterministe semble donc valide.

Explicitons notre modèle.

$$\begin{aligned} \forall t \in T \quad X_t &= \sum_{k=1}^3 \text{summary(lm2)}\$coefficients[k,1] t^{k-1} + \text{residus}[t] \\ &= 59.0271 + 0.5667t - 0.0031t^2 + B_t \end{aligned}$$

#### 14.5.3.2.2 Test de non corrélation et d'homoscédasticité

Effectuons le test de Harrison-McCabe. Rappelons que pour ce test les hypothèses sont les suivantes.

- **Hypothèse nulle**  $H_0$  : « Les résidus sont homoscédastiques. »
- **Hypothèse alternative**  $H_1$  : « Les résidus sont hétéroscédastiques. »

```
library(lmtest)

## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##   as.Date, as.Date.numeric

hmctest(lm2) # class "hctest"

##
## Harrison-McCabe test
##
## data:  lm2
## HMC = 0.57164, p-value = 0.91

# Extraction de la p-valeur
names(hmctest(lm2))

## [1] "statistic" "method"      "p.value"      "data.name"

hmctest(lm2)$p.value

## [1] 0.909
```

On a `hmctest(lm2)$p.value = 0.925 > 0.05` donc les résidus semblent avoir une variance constante.

Effectuons le test de Breusch-Godfrey. Rappelons que pour ce test les hypothèses sont les suivantes.

- **Hypothèse nulle**  $H_0$  : « Il n'existe pas d'autocorrélation dans les résidus pour tout ordre inférieur ou égal à  $p$ . »
- **Hypothèse alternative**  $H_A$  : « Il existe une autocorrélation dans les résidus pour un ordre inférieur ou égal à  $p$ . »

```
bgtest(lm2) # class "bgtest" "hctest"

##
## Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lm2
## LM test = 62.56, df = 1, p-value = 2.584e-15

names(bgtest(lm2))

## [1] "statistic"      "parameter"      "method"          "p.value"          "data.name"
## [6] "coefficients"  "vcov"

bgtest(lm2)$p.value

## [1] 2.584299e-15
```

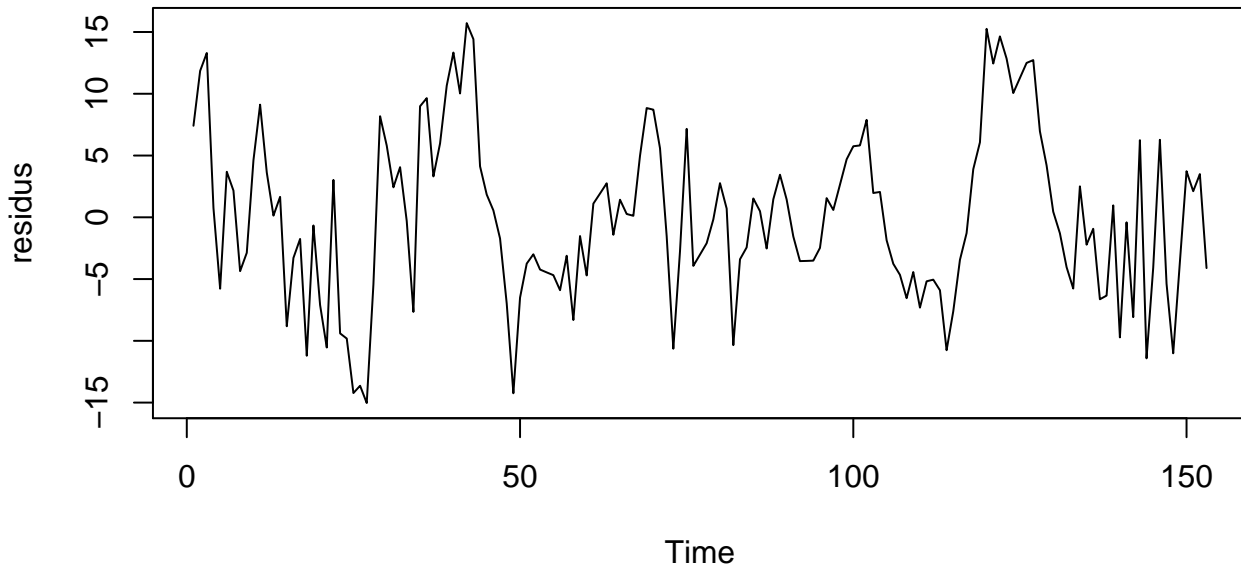
On a `bgtest(lm2)$p.value = 0 < 0.05` donc les résidus sont corrélés. On ne peut donc pas utiliser le modèle `lm2` pour effectuer des prévisions. On va affiner le modèle en modélisant les résidus.

#### 14.5.3.2.3 Modélisation des résidus

#### 14.5.3.2.3.1 Test de blancheur

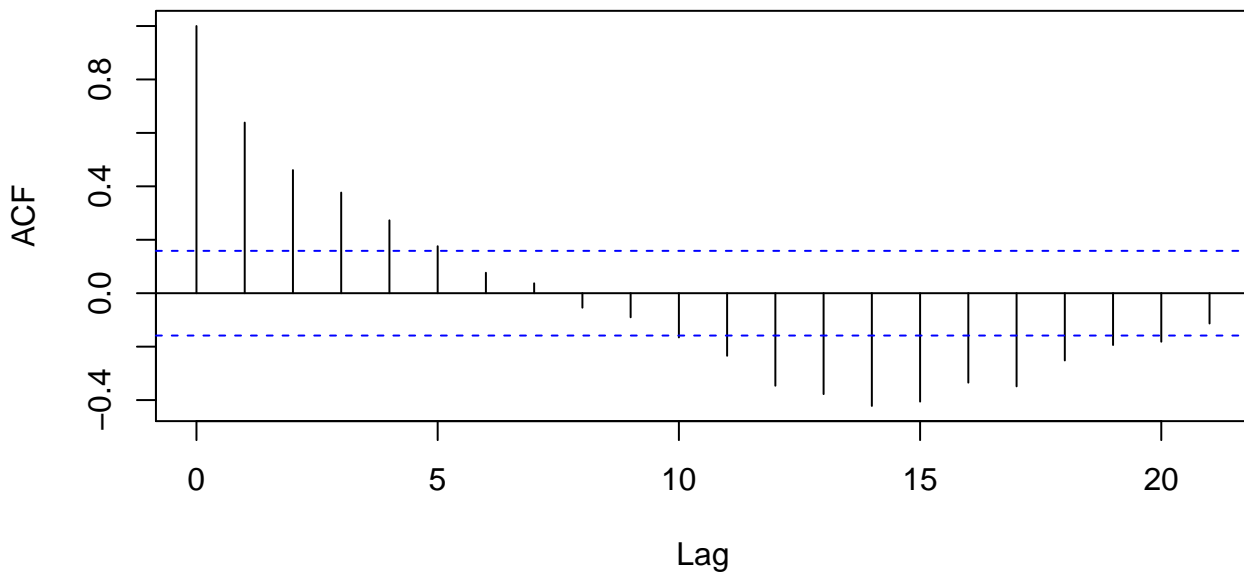
```
plot(residus, main="Chronogramme de la série residus")  
acf(residus)
```

### Chronogramme de la série résidus



(a) Chronogramme de la série `résidus`

### Series résidus



(b) Autocorrélogramme empirique de la série `résidus`

FIGURE 14.4 – Chronogramme et autocorrélogramme empirique de la série `résidus`

Les résidus ne forment pas un bruit blanc. Modélisons la série résiduelle par un processus ARMA.

### 14.5.3.2.3.2 Modélisation par processus ARMA

#### 14.5.3.2.3.2.1 Identification des degrés

Déterminons les ordres  $p$  et  $q$ . Établissons la table des fonctions d'autocorrélation étendues.

**Remarque** Le package `TSA` n'étant plus disponible, il faut extraire le code source de la fonction `eacf()`.

```
eacf(residus)

## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x o o o o o x x x x
## 1 o o o o o o o o o o o o o o
## 2 x x o o o o o o o o o o o o
## 3 x x o o o o o o o o o o o o
## 4 x o x o o o o o o o o o o o
## 5 x o o o o o o o o o o o o o
## 6 o x x o o o o o o o o o o o
## 7 o x x o o o o o o o o o o o
```

Cette table suggère un modèle ARMA(1,1) pour les résidus. (On pourrait prendre un ARMA(2,2) puisqu'il englobe l'ARMA(1,1) mais c'est inutile car cela augmentera les critères AIC et BIC. On propose tout de même cette modélisation en annexe.)

#### 14.5.3.2.3.2.2 Estimation des paramètres du modèle ARMA(1,1)

Construisons le modèle complet.

```
modeleD <- Arima(Temp, xreg=cbind(temps,temps2), order=c(1,0,1))
modeleD # class "forecast_ARIMA" "ARIMA" "Arima"

## Series: Temp
## Regression with ARIMA(1,0,1) errors
##
## Coefficients:
##          ar1          ma1  intercept      temps      temps2
##          0.7515      -0.1906      60.1454      0.5411      -3e-03
## s.e.      0.0852      0.1299       3.8137      0.1141       7e-04
##
## sigma^2 estimated as 27.4: log likelihood=-468.09
## AIC=948.18   AICc=948.76   BIC=966.37
```

#### 14.5.3.2.3.2.3 Simplification du modèle ARMA(1,1) en AR(1)

Testons si le modèle ARMA(1,1) est simplifiable.

**Remarque** Le package `caschrono` n'étant plus disponible il faut extraire le code source de la fonction `t_stat`.

```
t_stat(modeleD) # class "matrix"

##          ar1          ma1  intercept      temps      temps2
## t.stat  8.824925  -1.467682  15.77073  4.743679  -4.221192
## p.val   0.000000  0.142191   0.00000  0.000002  0.000024

t_stat(modeleD)[2,2]

## [1] 0.142191
```

On a  $t\_stat(modeleD)[2,2] = 0.1422 > 0.05$  donc le modèle ARMA(1,1) est simplifiable en un AR(1).

```

modeleD <- Arima(Temp, xreg=cbind(temps,temps2), order=c(1,0,0))
modeleD

## Series: Temp
## Regression with ARIMA(1,0,0) errors
##
## Coefficients:
##          ar1  intercept    temps   temps2
##          0.6413    59.6370    0.5545   -0.0031
## s.e.    0.0618     3.3566    0.1006    0.0006
##
## sigma^2 estimated as 27.56:  log likelihood=-469.03
## AIC=948.07   AICc=948.48   BIC=963.22

```

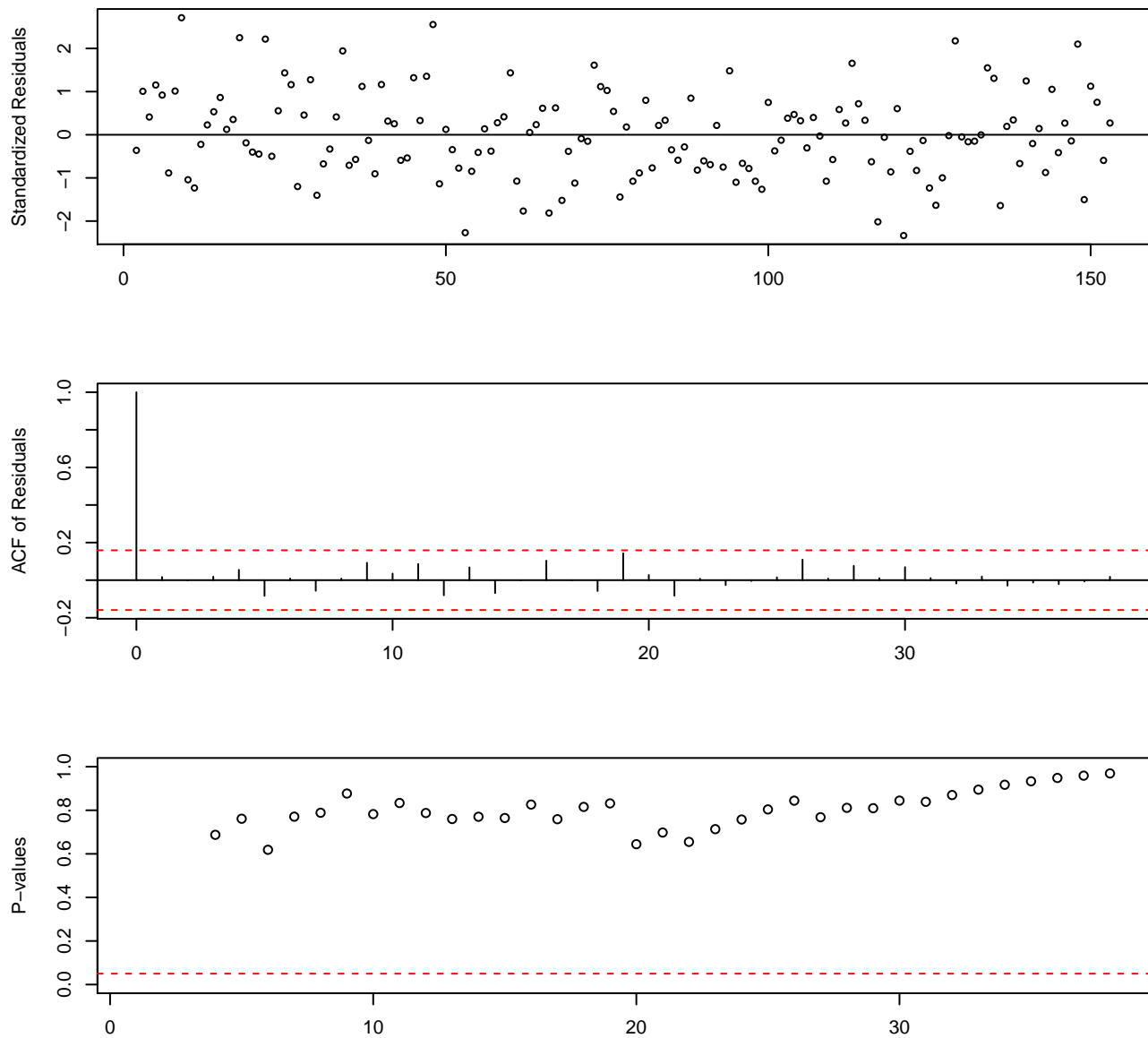
#### 14.5.3.2.3.2.4 Validation du modèle simplifié AR(1)

**Remarque** Le package TSA n'étant plus disponible il faut extraire le code source de la fonction `tsdiag.Arima()`. Le package `spgs` n'existe plus il faut extraire le code source de la fonction `lb.test()`.

```

tsdiag.Arima(modeleD, gof.lag=round(length(Temp)/4))

```



Le modèle simplifié AR(1) est valide.

#### 14.5.3.3 Explicitation du modèle final ARMAX(2,1)

Finalement, explicitons notre modèle final.

$$\forall t \in T \quad X_t = m(t) + B_t$$

avec

- $m$  la fonction du second degré définie par

$$\begin{aligned} \forall t \in T \quad m(t) &= \sum_{k=1}^3 \text{coef test}(\text{modeleD})[k+1,1] \, t^{k-1} \\ &= 59.637 + 0.5545t - 0.0031t^2 \end{aligned}$$

- $(B_t)_{t \in T}$  le processus AR(1) défini par

$$\begin{aligned} \forall t \in T \quad B_t &= \mu + \text{coefest}(\text{modeleD})[1,1](B_{t-1} - \mu) + \varepsilon_t \\ &= 59.637 + 0.6413(B_{t-1} - 59.637) + \varepsilon_t \end{aligned}$$

où  $(\varepsilon_t)_{t \in T}$  est un bruit blanc (faible).

```
# Extraction des coefficients
coefest(modeleD) # class "coefest"

##
## z test of coefficients:
##
##              Estimate Std. Error z value Pr(>|z|)
## ar1           0.64133057  0.06179014 10.3792 < 2.2e-16 ***
## intercept 59.63704450  3.35658564 17.7672 < 2.2e-16 ***
## temps       0.55446598  0.10064410  5.5092 3.605e-08 ***
## temps2     -0.00309851  0.00063568 -4.8743 1.092e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On s'attendait à trouver les mêmes coefficients que dans la régression. Par défaut, les estimations se font par maximum de vraisemblance, et non par moindres carrés, comme pour la régression.

#### 14.5.3.4 Annexe - Modélisation par processus ARMA(2,2)

Supposons que l'on ait modélisé les résidus par un processus ARMA(2,2).

```
modeleD1 <- Arima(Temp, xreg=cbind(temps,temps2), order=c(2,0,2))
modeleD1

## Series: Temp
## Regression with ARIMA(2,0,2) errors
##
## Coefficients:
##          ar1      ar2      ma1      ma2 intercept   temps   temps2
##       -0.0933  0.6669  0.6795 -0.2397   60.2725  0.5371  -3e-03
## s.e.    0.1682  0.1146  0.1820  0.1197    3.8925  0.1164   7e-04
##
## sigma^2 estimated as 27.27: log likelihood=-466.76
## AIC=949.52   AICc=950.52   BIC=973.76
```

##### 14.5.3.4.1 Simplification du modèle ARMA(2,2) en ARMA(2,1)

Testons si le modèle ARMA(2,2) est simplifiable.

```
t_stat(modeleD1)

##          ar1      ar2      ma1      ma2 intercept   temps   temps2
## t.stat -0.554605  5.817496  3.734105 -2.001800   15.48421  4.615157 -4.106399
## p.val   0.579165  0.000000  0.000188  0.045306    0.00000  0.000004  0.000040

t_stat(modeleD1)[2,4]

## [1] 0.045306
```

On a  $t\_stat(modeleD1)[2,4] = 0.0453$  donc le modèle ARMA(2,2) est simplifiable en un ARMA(2,1). Construisons le modèle simplifié.

```
modeleD2 <- Arima(Temp, xreg=cbind(temps,temps2), order=c(2,0,1))
modeleD2

## Series: Temp
## Regression with ARIMA(2,0,1) errors
##
## Coefficients:
##          ar1      ar2      ma1 intercept   temps   temps2
##       0.8937 -0.0967 -0.3270   60.2095  0.5394  -3e-03
## s.e.    0.3567  0.2450  0.3435    3.8649  0.1155   7e-04
##
## sigma^2 estimated as 27.56: log likelihood=-468.02
## AIC=950.03   AICc=950.8   BIC=971.25
```

**Remarque** Par défaut, les estimations se font par maximum de vraisemblance, et non par moindres carrés, comme pour la régression. Seul le modèle `modeleD2b` permet d'obtenir les mêmes coefficients que dans la régression linéaire (ajout de `method="CSS"`).

```
modeleD2b <- Arima(Temp, xreg=cbind(temps,temps2), order=c(2,0,1), method="CSS")
modeleD2b

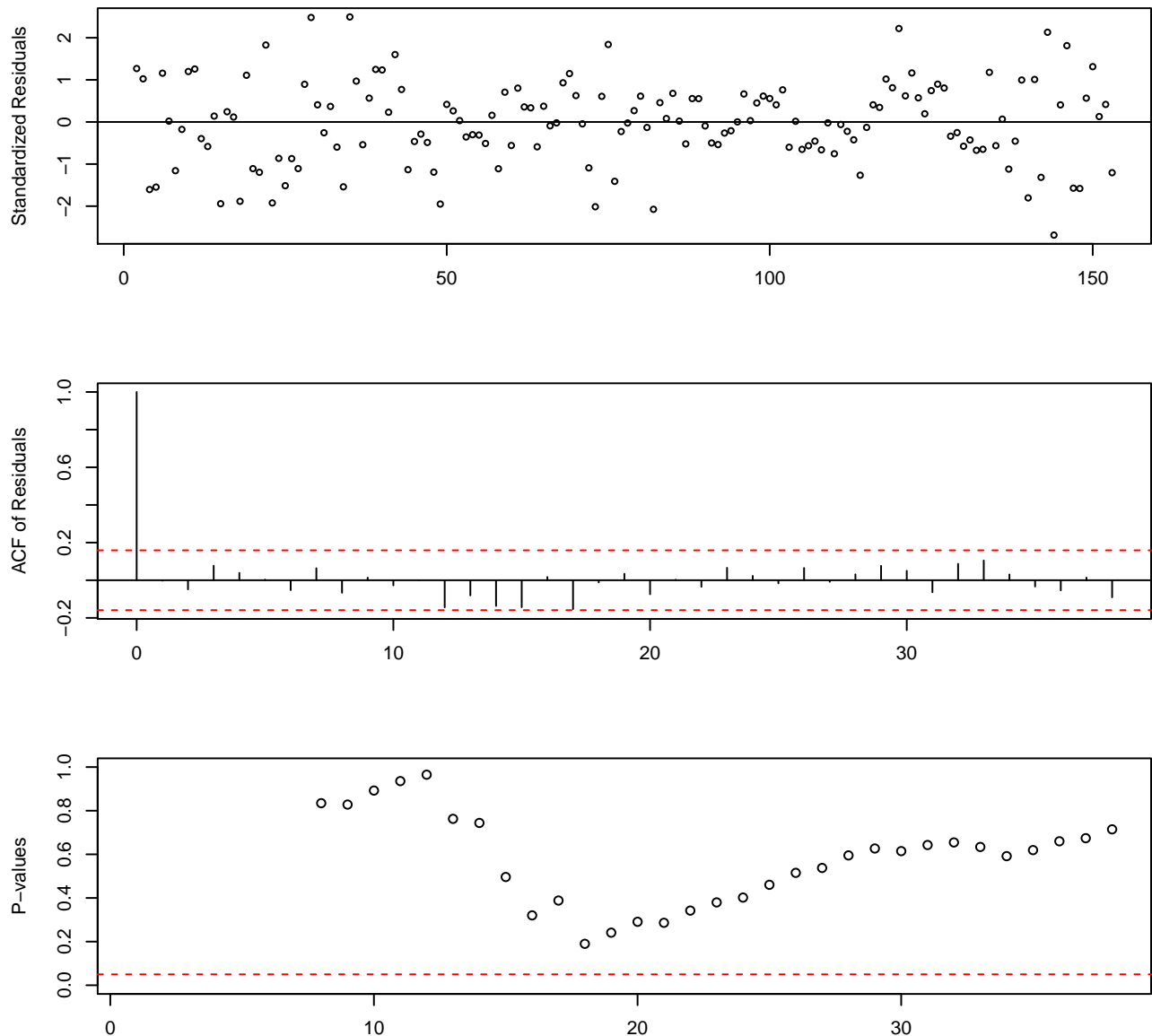
## Series: Temp
## Regression with ARIMA(2,0,1) errors
```

```
##
## Coefficients:
##      ar1      ar2      ma1  intercept  temps  temps2
##      0.8474 -0.0740 -0.2966  54.6337  0.6786 -0.0038
## s.e.  0.2505  0.1738  0.2410   4.6998  0.1312  0.0008
##
## sigma^2 estimated as 26.8:  part log likelihood=-466.6
```

#### 14.5.3.4.2 Validation du modèle simplifié ARMA(2,1)

Testons la validité du modèle simplifié.

```
tsdiag.Arima(modeleD2, gof.lag=round(length(Temp)/4))
```



Le modèle simplifié est valide.

#### 14.5.3.4.3 Comparaison des modèles ARMA(2,2) et ARMA(2,1) et sélection

Comparons `modeleD` et `modeleD2` d'après les critères d'information.

##### 14.5.3.4.3.1 Test de normalité des résidus

Vérifions qu'ils ont des résidus gaussiens.

```
shapiro.test(modeleD1$res)

##
##  Shapiro-Wilk normality test
##
## data:  modeleD1$res
## W = 0.99207, p-value = 0.5569

shapiro.test(modeleD2$res)

##
##  Shapiro-Wilk normality test
##
## data:  modeleD2$res
## W = 0.99251, p-value = 0.6077
```

On a `shapiro.test(modeleD1$res)$p.value = 0.5569 > 0.05` et `shapiro.test(modeleD2$res)$p.value = 0.6077 > 0.05` donc les résidus sont gaussiens.

##### 14.5.3.4.3.2 Critères d'information

Effectuons les tests AIC, AICc et BIC.

```
c(modeleD1$aic, modeleD1$aicc, modeleD1$bic)

## [1] 949.5202 950.5202 973.7637

c(modeleD2$aic, modeleD2$aicc, modeleD2$bic)

## [1] 950.0324 950.8049 971.2455
```

Seul le critère BIC, qui pénalise plus fortement le nombre de paramètres du modèle a diminué légèrement. L'AIC et l'AICc n'ont pas diminué.

##### 14.5.3.4.3.3 Test de rapport de vraisemblance

On réalise le test de rapport de vraisemblance pour les modèles emboîtés.

```
library(lmtest)
lrtest(modeleD2, modeleD1)

## Likelihood ratio test
##
## Model 1: Arima(y = Temp, order = c(2, 0, 1), xreg = cbind(temps, temps2))
## Model 2: Arima(y = Temp, order = c(2, 0, 2), xreg = cbind(temps, temps2))
##   #Df  LogLik Df   Chisq Pr(>Chisq)
## 1    7 -468.02
## 2    8 -466.76  1  2.5123    0.113
```

La perte de vraisemblance est faible donc on peut retenir le modèle simplifié `modeleD2`. Finalement explicitons notre modèle final.

$$\forall t \in T \quad X_t = m(t) + B_t$$

avec

- $m$  la fonction du second degré définie par

$$\begin{aligned}\forall t \in T \quad m(t) &= \sum_{k=1}^3 \text{coef test}(\text{modeleD2})[k+3,1] t^{k-1} \\ &= 60.2095 + 0.5394t - 0.003t^2\end{aligned}$$

- $(B_t)_{t \in T}$  le processus ARMA(2,1) défini par

$$\begin{aligned}\forall t \in T \quad (B_t - \mu) &= \sum_{k=1}^2 \text{coef test}(\text{modeleD2})[k,1](B_{t-k} - \mu) + \varepsilon_t + \text{coef test}(\text{modeleD2})[3,1]\varepsilon_{t-1} \\ &= 0.8937(B_{t-1} - \mu) - 0.0967(B_{t-2} - \mu) + \varepsilon_t - 0.327\varepsilon_{t-1}\end{aligned}$$

où  $(\varepsilon_t)_{t \in T}$  est un bruit blanc (faible).

```
# Extractions des paramètres et coefficients du modèle final
coef test(modeleD2)

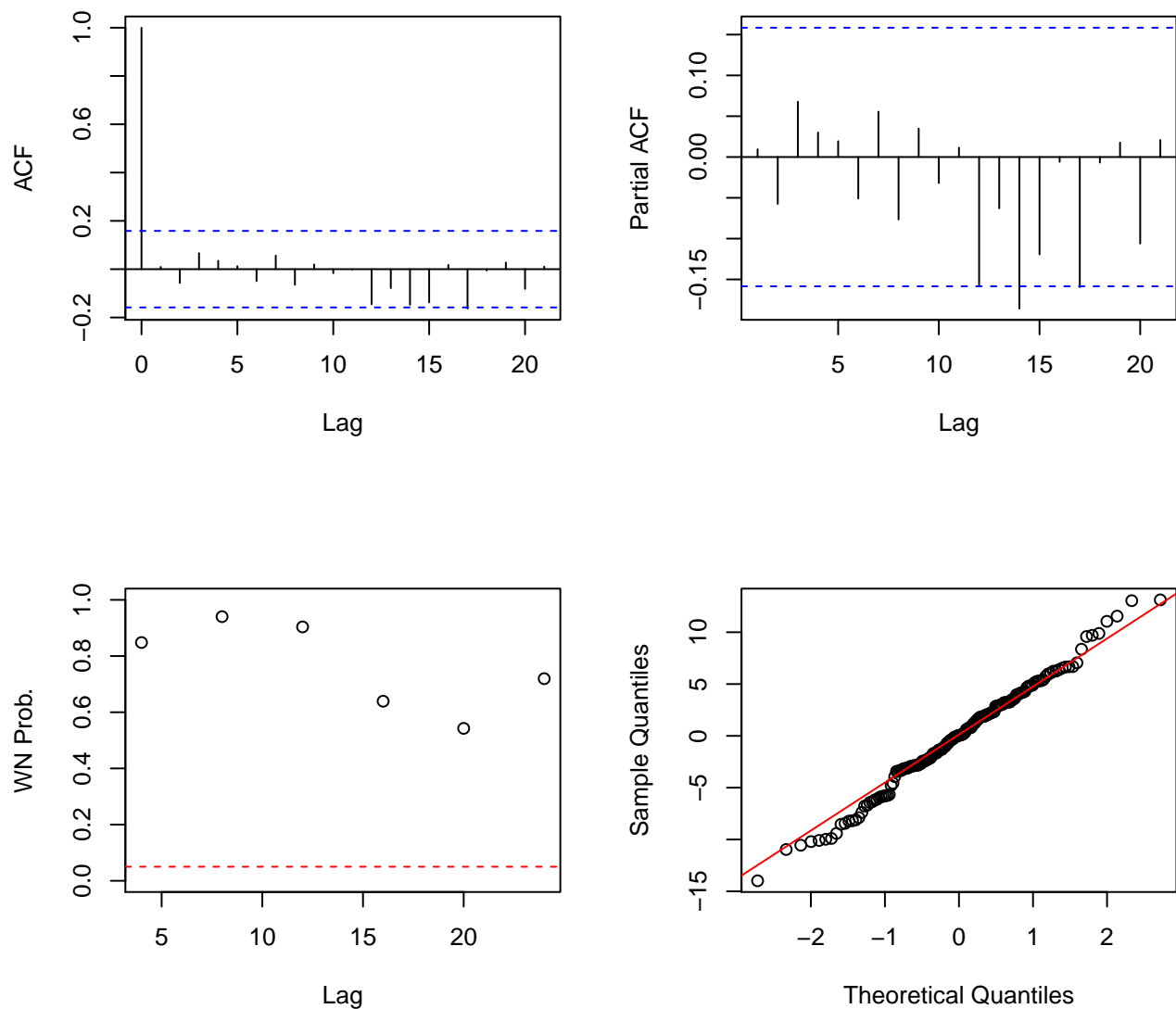
##
## z test of coefficients:
##
##           Estimate Std. Error z value Pr(>|z|)
## ar1          0.89371775 0.35674433  2.5052  0.01224 *
## ar2         -0.09673907 0.24500906 -0.3948  0.69296
## ma1         -0.32701633 0.34350219 -0.9520  0.34109
## intercept  60.20948350 3.86493995 15.5784 < 2.2e-16 ***
## temps        0.53938783 0.11554480  4.6682 3.038e-06 ***
## temps2     -0.00301730 0.00072573 -4.1576 3.216e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### 14.5.3.5 Annexe 2

```
modeleD_1 <- arima(Temp, xreg=cbind(temps,temps2), order=c(1,0,1))
```

```
ts.diag(modeleD_1)
```

#### Residual Diagnostics Plots



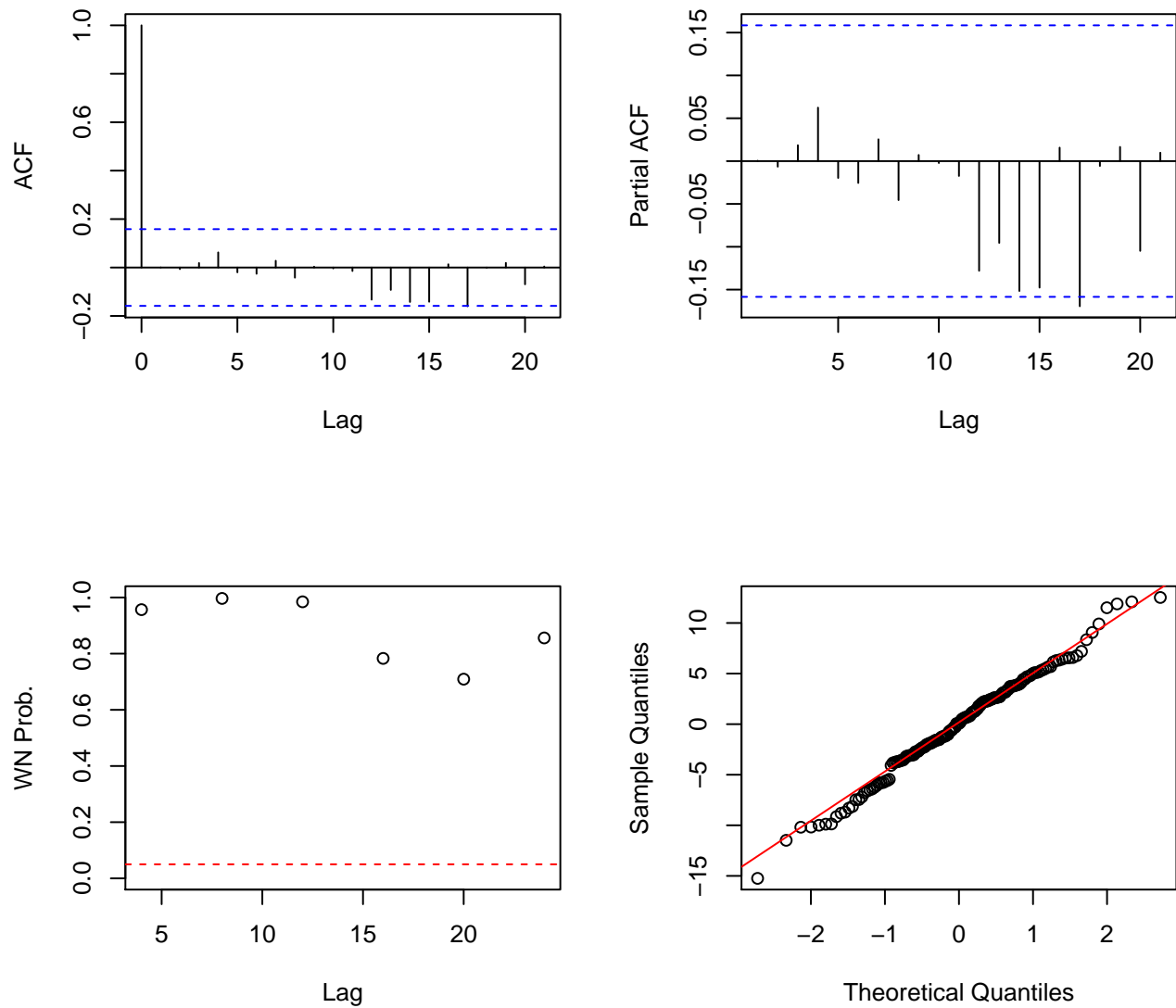
```
modeleD1_1 <- arima(Temp, xreg=cbind(temps,temps2), order=c(2,0,2))
modeleD1_1

##
## Call:
## arima(x = Temp, order = c(2, 0, 2), xreg = cbind(temps, temps2))
```

```
##
## Coefficients:
##          ar1      ar2      ma1      ma2  intercept    temps    temps2
##      -0.0933  0.6669  0.6795 -0.2397   60.2725   0.5371  -3e-03
## s.e.   0.1682  0.1146  0.1820   0.1197    3.8925   0.1164   7e-04
##
## sigma^2 estimated as 26.02:  log likelihood = -466.76,  aic = 949.52
```

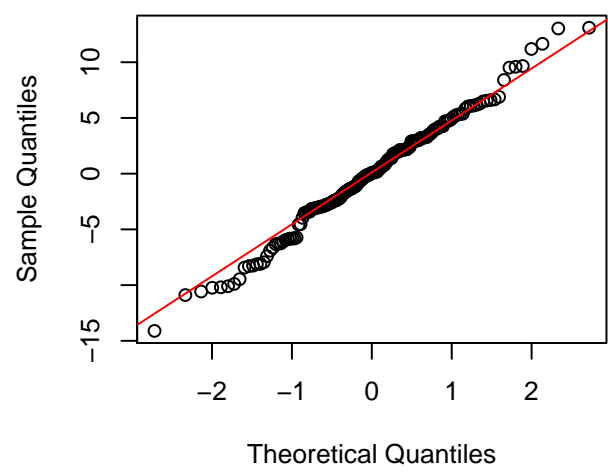
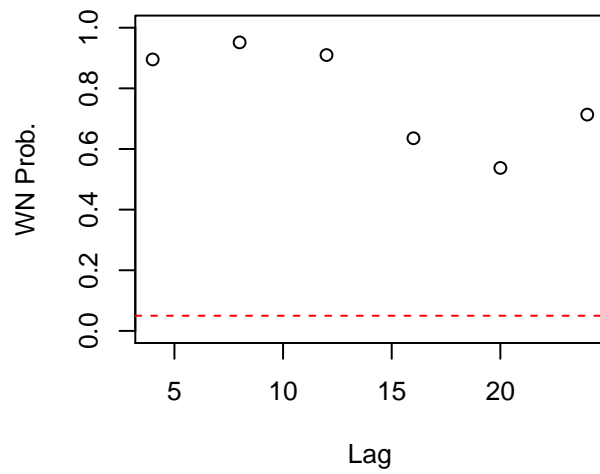
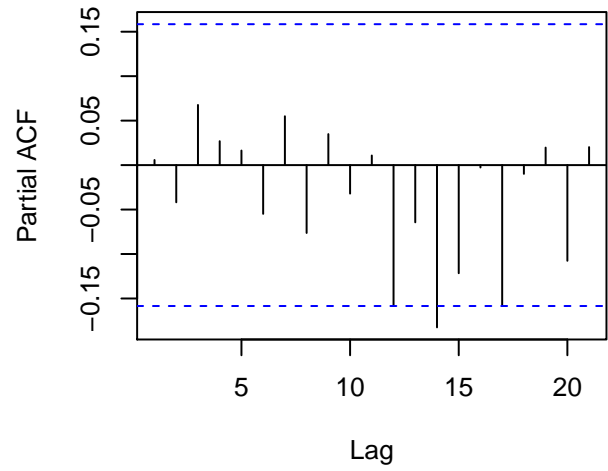
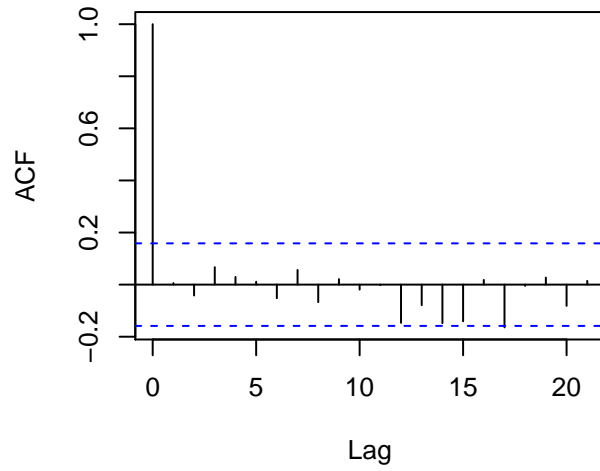
```
ts.diag(modeleD1_1)
```

### Residual Diagnostics Plots



```
modeleD2_1 <- arima(Temp, xreg=cbind(temps, temps2), order=c(2,0,1))
ts.diag(modeleD2_1)
```

## Residual Diagnostics Plots



### 14.5.4 Exemple de modélisation d'une tendance stochastique

Modélisons la série par une tendance stochastique. Supposons donc qu'il existe  $(d, c) \in \mathbb{N}^* \times \mathbb{R}$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \Delta^d(X_t) = c + B_t$$

#### 14.5.4.1 Identification du degré de différenciation

```
adf.test(Temp)

##
## Augmented Dickey-Fuller Test
##
## data: Temp
## Dickey-Fuller = -2.4041, Lag order = 5, p-value = 0.408
## alternative hypothesis: stationary

# Extraction de la p-valeur
adf.test(Temp)$p.value

## [1] 0.407972
```

On a `adf.test(Temp)$p.value = 0.408 > 0.05` donc la série `Temp` n'est pas stationnaire.

```
# Première différenciation
TempD <- diff(Temp, diff=1)
adf.test(TempD)

##
## Augmented Dickey-Fuller Test
##
## data: TempD
## Dickey-Fuller = -6.2228, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary

# Extraction de la p-valeur
adf.test(TempD)$p.value

## [1] 0.01
```

On a `adf.test(TempD)$p.value = 0.01 < 0.05` donc la série `TempD` est stationnaire. Prendre  $d = 1$  semble suffire pour rendre la série stationnaire. Confirmons avec le test de KPSS.

```
kpss.test(TempD)

##
## KPSS Test for Level Stationarity
##
## data: TempD
## KPSS Level = 0.084547, Truncation lag parameter = 4, p-value = 0.1

# Extraction de la p-valeur
kpss.test(TempD)$p.value

## [1] 0.1
```

On a `kpss.test(TempD)$p.value = 0.1 > 0.05` donc la série différenciée semble stationnaire. La modélisation de la série `Temp` par une tendance stochastique d'ordre 1 semble donc valide. Explicitons notre modèle. On suppose

qu'il existe  $c \in \mathbb{R}$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \Delta X_t = c + B_t$$

#### 14.5.4.2 Modélisation de la série résiduelle par processus ARMA

##### 14.5.4.2.1 Identification du degré

```
eacf(TempD)

## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o o o o o o o o o o o o o
## 1 x x o o o o o o o o o o o o
## 2 x o o o o o o o o o o o o o
## 3 x o o o o o o o o o o o o o
## 4 x o o x o o o o o o o o o o
## 5 o x x o o o o o o o o o o o
## 6 x x x o o o o o o o o o o o
## 7 x x x o o x o o o o o o o o
```

L'eacf suggère un modèle MA(1) pour les résidus. Construisons le modèle complet.

##### 14.5.4.2.2 Estimation des paramètres

```
modeleS <- Arima(Temp, order=c(0,1,1))
modeleS

## Series: Temp
## ARIMA(0,1,1)
##
## Coefficients:
##           ma1
##          -0.3840
## s.e.      0.0847
##
## sigma^2 estimated as 29.65:  log likelihood=-472.85
## AIC=949.7   AICc=949.78   BIC=955.75
```

##### 14.5.4.2.3 Simplification du modèle

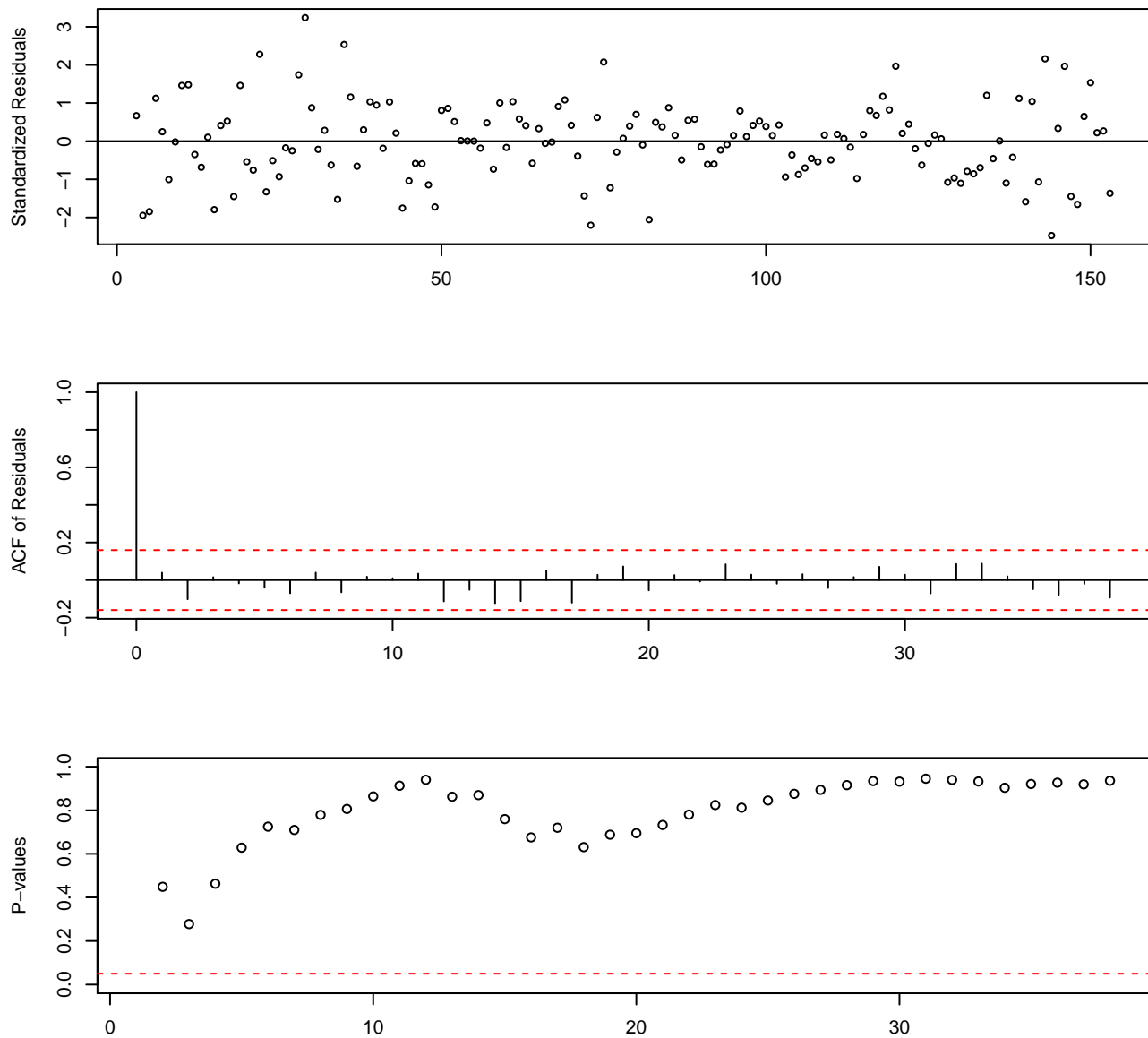
```
t_stat(modeleS)

##           ma1
## t.stat -4.533242
## p.val  0.000006
```

Le modèle n'est pas simplifiable. Validons le modèle complet pour la série initiale.

##### 14.5.4.2.4 Validation du modèle

```
tsdiag.Arima(modeleS, gof.lag=round(length(Temp)/4))
```



Le modèle est validé.

#### 14.5.4.3 Explicitation du modèle final ARIMA(0,1,1)

Finalement explicitons notre modèle final.

$$\begin{aligned} \forall t \in T \quad \Delta X_t &= c + \varepsilon_t + \text{coef test}(\text{modeleS})[1,1] \varepsilon_{t-1} \\ &= \varepsilon_t - 0.384 \varepsilon_{t-1} \end{aligned}$$

où  $(\varepsilon_t)_{t \in T}$  est un bruit blanc (faible).

```
coef test(modeleS)
##
## z test of coefficients:
```

```
##
##      Estimate Std. Error z value Pr(>|z|)
## ma1 -0.384045    0.084717 -4.5332 5.809e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### 14.5.5 Comparaison des modélisations déterministe et stochastique

Comparons nos deux modélisations.

- **Modèle ARMAX(2,1)** *Tendance déterministe du second degré et ARMA(2,1)*

$$\forall t \in T \quad X_t = m(t) + B_t$$

avec

- \*  $m$  la fonction du second degré définie par

$$\begin{aligned} \forall t \in T \quad m(t) &= \sum_{k=1}^3 \text{coef test}(\text{modeleD2})[k+3,1] t^{k-1} \\ &= 60.2095 + 0.5394t - 0.003t^2 \end{aligned}$$

- \*  $(B_t)_{t \in T}$  le processus ARMA(2,1) défini par

$$\begin{aligned} \forall t \in T \quad (B_t - \mu) &= \sum_{k=1}^2 \text{coef test}(\text{modeleD2})[k,1] (B_{t-k} - \mu) + \varepsilon_t + \text{coef test}(\text{modeleD2})[3,1] \varepsilon_{t-1} \\ &= 0.8937(B_{t-1} - \mu) - 0.0967(B_{t-2} - \mu) + \varepsilon_t - 0.327\varepsilon_{t-1} \end{aligned}$$

où  $(\varepsilon_t)_{t \in T}$  est un bruit blanc (faible).

- **Modèle ARIMA(0,1,1)** *Tendance stochastique d'ordre 1 et MA(1)*

$$\begin{aligned} \forall t \in T \quad \Delta X_t &= c + \varepsilon_t + \text{coef test}(\text{modeleS})[1,1] \varepsilon_{t-1} \\ &= \varepsilon_t - 0.384\varepsilon_{t-1} \end{aligned}$$

où  $(\varepsilon_t)_{t \in T}$  est un bruit blanc (faible).

#### 14.5.5.1 Test de normalité des résidus

On a déjà vérifié que les résidus du modèle `modeleD2` sont gaussiens.

Vérifions que les résidus du modèle stochastique sont gaussiens.

```
shapiro.test(modeleS$res)

##
##  Shapiro-Wilk normality test
##
## data:  modeleS$res
## W = 0.99336, p-value = 0.708

library(fBasics)

## Loading required package: timeDate
## Loading required package: timeSeries
##
## Attaching package: 'timeSeries'
## The following object is masked from 'package:zoo':
##
##   time<-

dagoTest(modeleS$res)$test$p.value[1]

## Omnibus Test
## 0.4217363
```

Les résidus étant gaussiens, on peut tracer les intervalles de confiance.

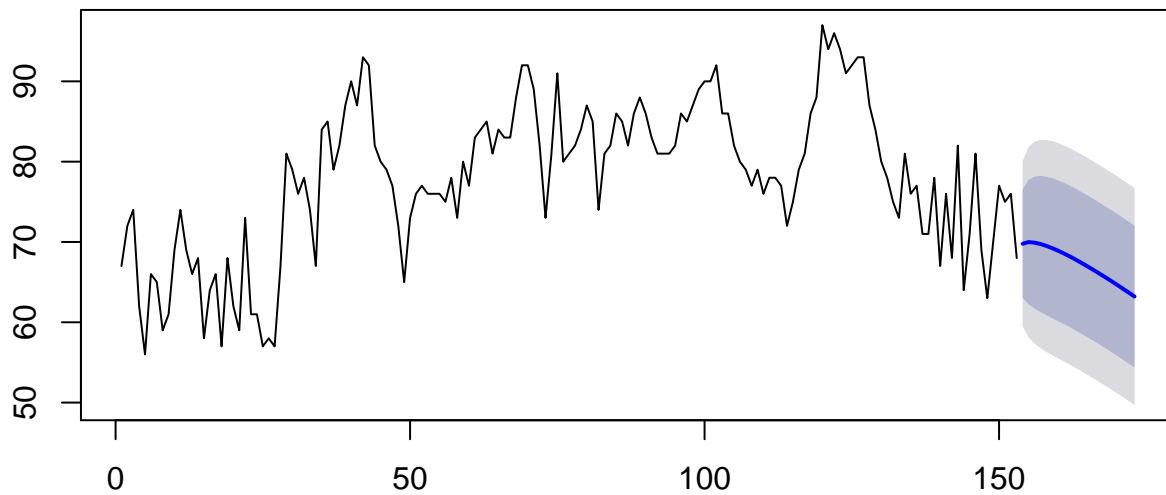
### 14.5.5.2 Prévisions des modèles

```
nbP <- 20
tempsP <- ((length(Temp)+1):(length(Temp)+nbP))
temps2P <- tempsP^2
DF <- data.frame(temps=tempsP, temps2=temps2P)
```

```
prevD <- forecast(modeleD2, xreg=data.matrix(DF[1:nbP, ], rownames.force = NA), h=nbP)
prevS <- forecast(modeleS, h=nbP)
```

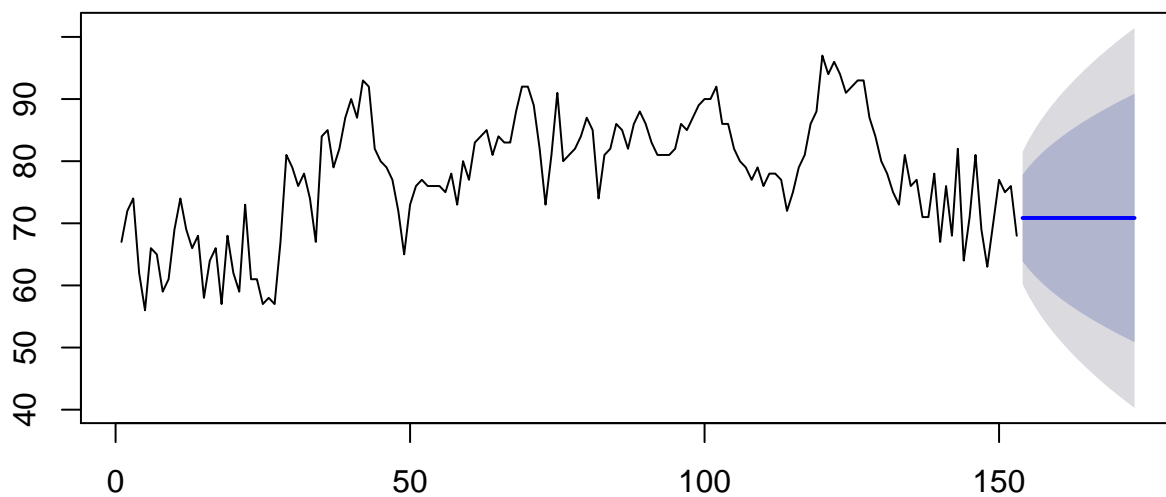
```
plot(prevD, PI=TRUE)
plot(prevS, PI=TRUE)
```

### Forecasts from Regression with ARIMA(2,0,1) errors



(a) Prévisions du modèle ARMAX(2,1) pour la série Temp

### Forecasts from ARIMA(0,1,1)



(b) Prévisions du modèle ARIMA(0,1,1) pour la série Temp

FIGURE 14.5 – Prévisions des modèles ARMAX(2,1) et ARIMA(0,1,1) pour la série Temp

La Figure 14.5 montre que les prévisions sont relativement différentes. Dans le cas de la tendance déterministe, on considère que la série va poursuivre sa tendance polynomiale, avec des variations modérées. En revanche, dans le cas de la tendance stochastique, n'importe quelle fluctuation (à la hausse ou à la baisse) est possible. D'où une prévision constante en moyenne et un intervalle de fluctuation très large.

### 14.5.5.3 Critères d'information

```
c(modeleD2$aic, modeleD2$aicc, modeleD2$bic)

## [1] 950.0324 950.8049 971.2455

c(modeleS$aic, modeleS$aicc, modeleS$bic)

## [1] 949.7010 949.7816 955.7488
```

Les valeurs des critères AIC et AICc sont proches. En revanche, le critère BIC indique que la modélisation stochastique est meilleure.

### 14.5.5.4 Indicateurs d'écart

On peut aussi avoir recours aux indicateurs d'écart en estimant les modèles sur le début de la série (95% des observations environ) et en comparant les prévisions obtenues aux valeurs réellement observées à la fin de la série. Il y a néanmoins une subtilité qui est généralement négligée : le modèle qui a été calibré sur la série complète n'est pas nécessairement celui qui aurait été retenu si on avait refait la modélisation sur la série tronquée. Et même si le processus  $(B_t)_{t \in T}$  était modélisé par un processus ARMA de même ordre, les coefficients ne seraient pas les mêmes.

```
Tronq <- round(length(Temp)*0.95)
modeleDT <- Arima(Temp[1:Tronq], xreg=cbind(temps[1:Tronq], temps2[1:Tronq]), order=c(2,0,1))
modeleST <- Arima(Temp[1:Tronq], order=c(0,1,1))
tempsPT <- ((Tronq +1):length(Temp))
temps2PT <- tempsPT^2
DFT <- data.frame(temps=tempsPT, temps2=temps2PT)
nbPT <- length(Temp) - Tronq
prevDT <- forecast(modeleDT, xreg=data.matrix(DFT[1:nbPT, ]), h=nbPT)

## Warning in forecast.forecast_ARIMA(modeleDT, xreg = data.matrix(DFT[1:nbPT, : xreg contains different
column names from the xreg used in training. Please check that the regressors are in the same order.

prevST <- forecast(modeleST, h=nbPT)
accuracy(prevDT, Temp[(Tronq +1):length(Temp)])

##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set -0.05767082 5.059257 3.998215 -0.5588642 5.358749 0.9500709
## Test set      0.59241820 5.621628 4.902299  0.2288362 6.791331 1.1649028
##              ACF1
## Training set 0.001960249
## Test set      NA

accuracy(prevST, Temp[(Tronq +1):length(Temp)])

##              ME      RMSE      MAE      MPE      MAPE      MASE
## Training set 0.03099152 5.301291 4.097524 -0.3069243 5.450184 0.973669
## Test set      2.12041443 5.872013 4.875000  2.3625602 6.615739 1.158416
##              ACF1
## Training set 0.02076651
## Test set      NA
```

Les critères indiquent que les performances des modélisations avec tendance déterministe et stochastique sont très proches, en terme de prévisions.

## 14.6 Exemple de modélisation d'une tendance déterministe

Considérons la série temporelle `LakeHuron` du package `datasets` du logiciel **R**. Cette série temporelle annuelle représente le niveau, en pieds, du lac Huron de 1875 à 1972.

```
LakeHuron # package datasets

## Time Series:
## Start = 1875
## End = 1972
## Frequency = 1
## [1] 580.38 581.86 580.97 580.80 579.79 580.39 580.42 580.82 581.40 581.32
## [11] 581.44 581.68 581.17 580.53 580.01 579.91 579.14 579.16 579.55 579.67
## [21] 578.44 578.24 579.10 579.09 579.35 578.82 579.32 579.01 579.00 579.80
## [31] 579.83 579.72 579.89 580.01 579.37 578.69 578.19 578.67 579.55 578.92
## [41] 578.09 579.37 580.13 580.14 579.51 579.24 578.66 578.86 578.05 577.79
## [51] 576.75 576.75 577.82 578.64 580.58 579.48 577.38 576.90 576.94 576.24
## [61] 576.84 576.85 576.90 577.79 578.18 577.51 577.23 578.42 579.61 579.05
## [71] 579.26 579.22 579.38 579.10 577.95 578.12 579.75 580.85 580.41 579.96
## [81] 579.61 578.76 578.18 577.21 577.13 579.10 578.25 577.91 576.89 575.96
## [91] 576.80 577.68 578.38 578.52 579.74 579.31 579.89 579.96
```

Cette série ne présente aucune valeur manquantes.

```
which(is.na(LakeHuron), arr.ind=TRUE)

## integer(0)
```

Cette série est constituée de 98 observations.

```
length(LakeHuron)

## [1] 98
```

Cette série est de classe `ts`.

```
class(LakeHuron)

## [1] "ts"
```

Traçons le chronogramme de la série `LakeHuron`.

```
plot(LakeHuron)
```

### 14.6.1 Modélisation de la fonction déterministe

```
temps <- 1:length(LakeHuron)
lm1 <- lm(LakeHuron ~ temps)
lm1

##
## Call:
## lm(formula = LakeHuron ~ temps)
##
## Coefficients:
## (Intercept)      temps
##    580.2020    -0.0242
```

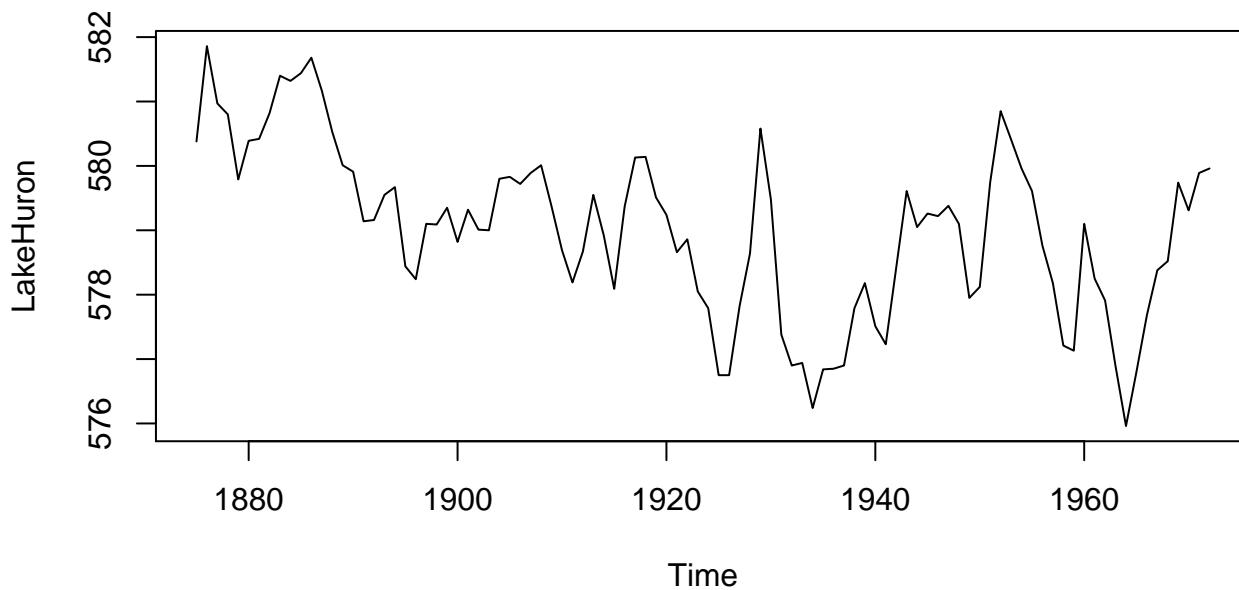


FIGURE 14.6 – Chronogramme de la série LakeHuron

```
names(lm1)

## [1] "coefficients" "residuals"      "effects"        "rank"
## [5] "fitted.values" "assign"          "qr"             "df.residual"
## [9] "xlevels"       "call"           "terms"          "model"
```

Déterminons les variances totale et résiduelle (non expliquée par la régression).

```
# Variance totale
var(LakeHuron) # package cmunorm

## [1] 1.737911

# Variance résiduelle
var(lm1$residuals) # ou plus simplement var(lm1$res)

## [1] 1.264378
```

Notons que la variance résiduelle représente 72.75% de la variance initiale.

## 14.6.2 Modélisation des résidus

# Chapitre 15

## Saisonnalité

### 15.1 Détection d'une composante saisonnière

Le chronogramme d'une série permet de détecter facilement une composante saisonnière déterministe, mais une composante saisonnière stochastique est plus difficilement identifiable, car l'amplitude des cycles peut varier. En revanche, tracer la fonction d'autocorrélation empirique permet de s'assurer qu'on est bien en présence d'une composante saisonnière.

**Théorème (Condition nécessaire à une saisonnalité déterministe pure)**

Soit  $(a, r) \in \mathbb{R} \times \mathbb{N}^*$ .

Pour tout processus  $(a \cos(\frac{2\pi t}{r}))_{t \in T}$  et  $(a \sin(\frac{2\pi t}{r}))_{t \in T}$ ,

$$\forall h \in \mathbb{Z} \quad \hat{\rho}(h) \xrightarrow{n \rightarrow +\infty} \frac{a^2}{2} \cos\left(\frac{2\pi h}{r}\right)$$

Le comportement de la fonction d'autocorrélation empirique dépend de la nature de la composante saisonnière.

### 15.2 Identification de la saisonnalité

La saisonnalité  $r$  pourrait se déduire plus ou moins directement du graphique des fonctions d'autocorrélation empiriques. Mais en fonction de la nature de bruit  $(B_t)_{t \in T}$ , la lecture peut se révéler difficile. On introduit ici un outil appelé périodogramme qui va permettre d'identifier des candidates pour la saisonnalité  $r$ .

#### 15.2.1 Périodogramme

#### 15.2.2 Validation de la saisonnalité

### 15.3 Identification de la nature de la saisonnalité

Supposons que la saisonnalité  $r$  des données saisonnières que l'on souhaite modéliser est connue. En se basant sur les tests de Canova-Hansen (CH) et de Osborn-Chui-Smith-Birchenhall (OCSB), nous proposons la méthode suivante pour distinguer entre saisonnalité déterministe et saisonnalité stochastique. Dans le cas d'un processus à saisonnalité stochastique, nous présentons un test afin d'estimer la différence saisonnière  $D$ . Ce test peut être interprété comme l'équivalent du test ADF pour les racines unitaires, mais adaptés aux données saisonnières ; il teste l'existence d'une racine unitaire saisonnière.

- 15.4 Modélisation d'une saisonnalité déterministe : processus SAR-MAX**
- 15.5 Modélisation d'une saisonnalité stochastique : processus SARIMA**
- 15.6 Exemple de modélisation d'une saisonnalité**

Considérons le jeu de données `tempdub` du package `TSA` et affichons les 6 premières et 6 dernières observations.

## Chapitre 16

# Tendance et saisonnalité : processus SARIMA(p,d,q)(P,D,Q)[r]

Considérons des séries présentant à la fois une composante de tendance et une composante de saisonnalité.

## 16.1 Exemple de modélisation d’une tendance et d’une saisonnalité

Considérons la série temporelle **AirPassengers** du logiciel **R**. Cette série temporelle mensuelle représente le nombre de voyageurs de la compagnie Airline, en milliers, de janvier 1949 à décembre 1960.

```
data(AirPassengers)
AirPassengers

##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1949 112 118 132 129 121 135 148 148 136 119 104 118
## 1950 115 126 141 135 125 149 170 170 158 133 114 140
## 1951 145 150 178 163 172 178 199 199 184 162 146 166
## 1952 171 180 193 181 183 218 230 242 209 191 172 194
## 1953 196 196 236 235 229 243 264 272 237 211 180 201
## 1954 204 188 235 227 234 264 302 293 259 229 203 229
## 1955 242 233 267 269 270 315 364 347 312 274 237 278
## 1956 284 277 317 313 318 374 413 405 355 306 271 306
## 1957 315 301 356 348 355 422 465 467 404 347 305 336
## 1958 340 318 362 348 363 435 491 505 404 359 310 337
## 1959 360 342 406 396 420 472 548 559 463 407 362 405
## 1960 417 391 419 461 472 535 622 606 508 461 390 432
```

Affichons uniquement les 6 premières et 6 dernières observations.

```
head(AirPassengers) # class "numeric" ou AirPassengers[1:5]

## [1] 112 118 132 129 121 135

tail(AirPassengers) # class "numeric"

## [1] 622 606 508 461 390 432
```

Cette série est constituée de 144 observations.

```
length(AirPassengers)

## [1] 144
```

Elle ne présente aucune valeur manquante.

```
which(is.na(AirPassengers), arr.ind=TRUE)

## integer(0)
```

Cette série est de classe **ts**.

```
class(AirPassengers)

## [1] "ts"
```

On peut obtenir le vecteur des temps de cette série.

```
time(AirPassengers) # class "ts"

##      Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 1949 1949.000 1949.083 1949.167 1949.250 1949.333 1949.417 1949.500 1949.583
## 1950 1950.000 1950.083 1950.167 1950.250 1950.333 1950.417 1950.500 1950.583
## 1951 1951.000 1951.083 1951.167 1951.250 1951.333 1951.417 1951.500 1951.583
```

```
## 1952 1952.000 1952.083 1952.167 1952.250 1952.333 1952.417 1952.500 1952.583
## 1953 1953.000 1953.083 1953.167 1953.250 1953.333 1953.417 1953.500 1953.583
## 1954 1954.000 1954.083 1954.167 1954.250 1954.333 1954.417 1954.500 1954.583
## 1955 1955.000 1955.083 1955.167 1955.250 1955.333 1955.417 1955.500 1955.583
## 1956 1956.000 1956.083 1956.167 1956.250 1956.333 1956.417 1956.500 1956.583
## 1957 1957.000 1957.083 1957.167 1957.250 1957.333 1957.417 1957.500 1957.583
## 1958 1958.000 1958.083 1958.167 1958.250 1958.333 1958.417 1958.500 1958.583
## 1959 1959.000 1959.083 1959.167 1959.250 1959.333 1959.417 1959.500 1959.583
## 1960 1960.000 1960.083 1960.167 1960.250 1960.333 1960.417 1960.500 1960.583
##           Sep      Oct      Nov      Dec
## 1949 1949.667 1949.750 1949.833 1949.917
## 1950 1950.667 1950.750 1950.833 1950.917
## 1951 1951.667 1951.750 1951.833 1951.917
## 1952 1952.667 1952.750 1952.833 1952.917
## 1953 1953.667 1953.750 1953.833 1953.917
## 1954 1954.667 1954.750 1954.833 1954.917
## 1955 1955.667 1955.750 1955.833 1955.917
## 1956 1956.667 1956.750 1956.833 1956.917
## 1957 1957.667 1957.750 1957.833 1957.917
## 1958 1958.667 1958.750 1958.833 1958.917
## 1959 1959.667 1959.750 1959.833 1959.917
## 1960 1960.667 1960.750 1960.833 1960.917
```

Posons  $T = \{1, \dots, 144\}$  et, pour tout  $t \in T$ ,  $X_t = \text{AirPassengers}[t]$ . On cherche donc à modéliser la série temporelle  $(X_t)_{t \in T}$ . Traçons le chronogramme de cette série.

```
plot(AirPassengers, main="Chronogramme de la série AirPassengers")
```

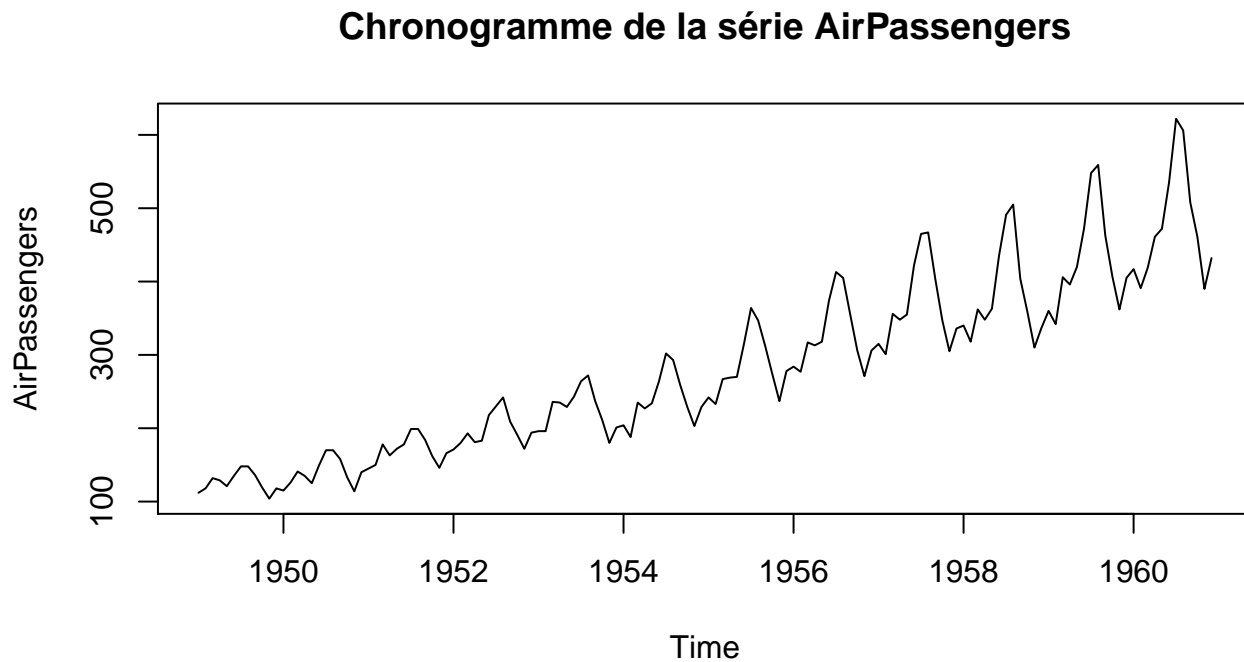


FIGURE 16.1 – Chronogramme de la série AirPassengers

Traçons la décomposition usuelle de la série AirPassengers.

```
plot(decompose(AirPassengers))
```

### Decomposition of additive time series

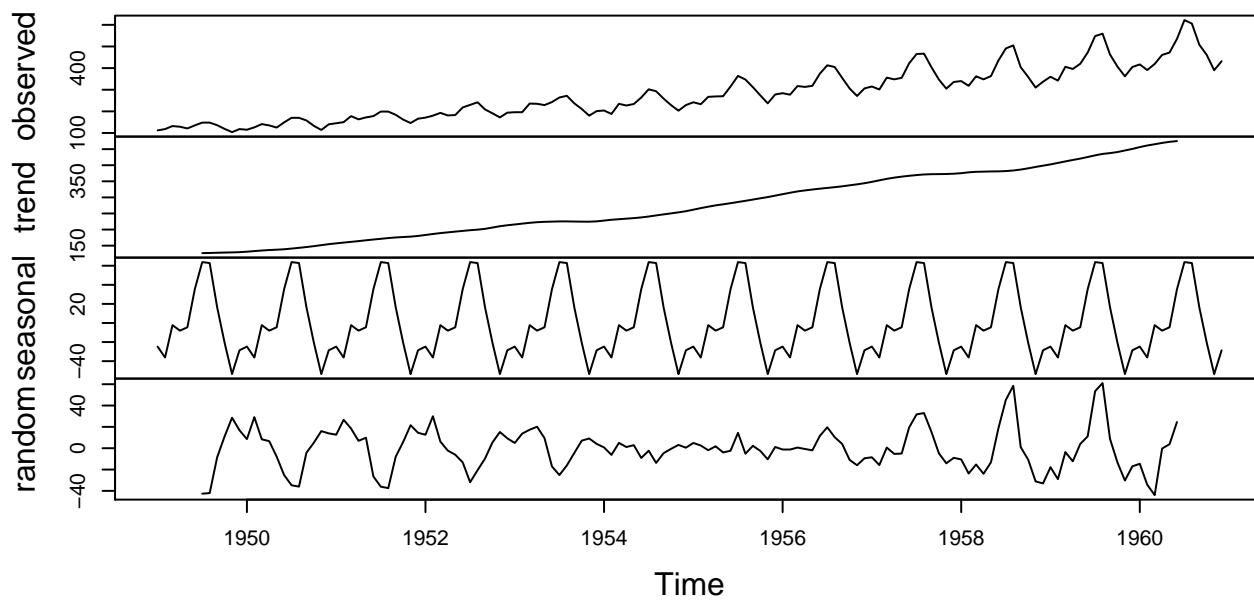


FIGURE 16.2 – Décomposition de la série AirPassengers

Le chronogramme de la série `AirPassengers` montre bien la présence d'une tendance et d'un cycle. Vérifions que la série `AirPassengers` n'est pas stationnaire.

```
library(tseries) # fonction kpss.test()

## Registered S3 method overwritten by 'quantmod':
## method      from
## as.zoo.data.frame zoo

kpss.test(AirPassengers)

## Warning in kpss.test(AirPassengers): p-value smaller than printed p-value

##
## KPSS Test for Level Stationarity
##
## data:  AirPassengers
## KPSS Level = 2.7395, Truncation lag parameter = 4, p-value = 0.01
```

On a `kpss.test(AirPassengers)$p.value = 0.01 < 0.05` donc le test (KPSS) rejette la stationnarité de la série `AirPassengers`.

La croissance du cycle avec le temps nous suggère deux méthodes.

- Procéder à une transformation de Box-Cox (pour stabiliser la variance ou variabilité qui augmente avec le temps).
- Modéliser cette série par un modèle présentant une tendance et une saisonnalité stochastiques.

Plus précisément nous proposerons les modèles suivants.

- En sous-section 15.1.1, *Tendance déterministe puis saisonnalité déterministe*.
  - \* En sous-paragraphe 15.1.1.3.4.1, *Tendance déterministe puis saisonnalité déterministe par moyennes saisonnières*.
  - \* En sous-paragraphe 15.1.1.3.4.2 et 15.1.1.3.4.3, *Tendance déterministe puis saisonnalité déterministe par régression harmonique*.
- En paragraphe 15.1.1.3.5, *Tendance déterministe puis saisonnalité stochastique*.
- En sous-section 15.1.2, *Saisonnalité déterministe puis tendance stochastique*.
  - \* En sous-paragraphe 15.1.2.2.4.1, *Saisonnalité déterministe par moyennes saisonnières puis tendance stochastique*.
  - \* En sous-paragraphe 15.1.2.2.4.2, *Saisonnalité déterministe par régression harmonique puis tendance stochastique*.
- En sous-section, 15.1.3, *Tendance stochastique et saisonnalité stochastique*.
- En sous-section 15.1.4, *Tendance stochastique et saisonnalité stochastique*.

### 16.1.1 Modélisation par transformation de Box-Cox (tendance puis saisonnalité)

Procédons à une transformation de Box-Cox. Rappelons que la fonction `boxcox()` renvoie un objet de class `list` contenant deux objets *i.e.* de longueur 2.

```
library(MASS)
temps <- time(AirPassengers)
bc <- boxcox(lm(AirPassengers~temps)) # class "list"
```

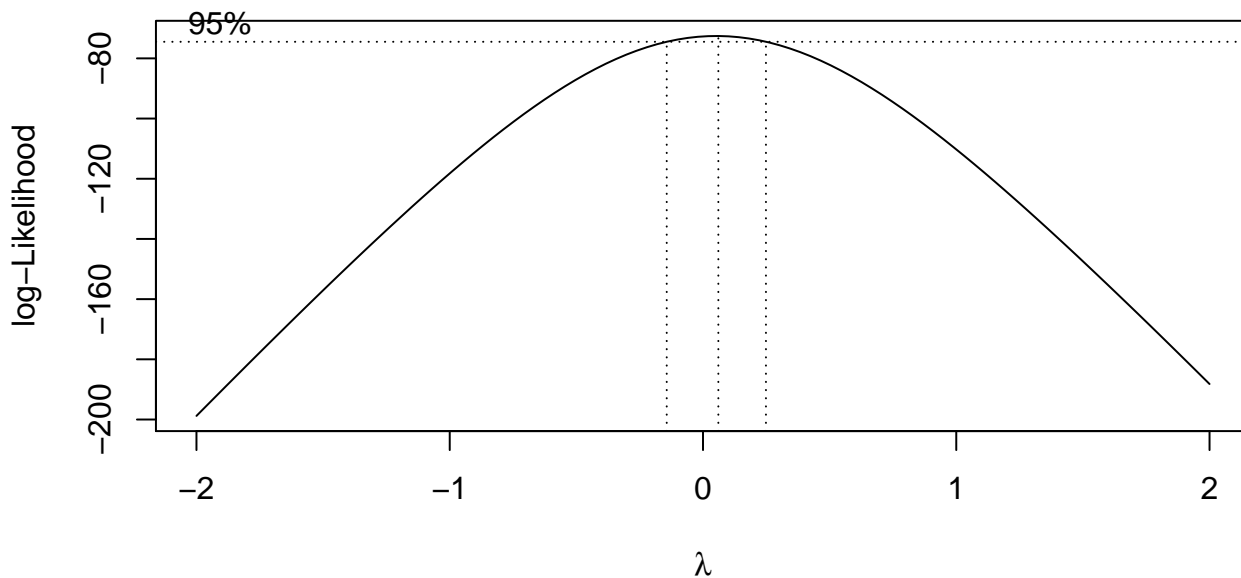


FIGURE 16.3 – Log-vraisemblance de la régression de la série `AirPassengers` transformée

Le coefficient  $\lambda$  correspondant à la vraisemblance maximale du modèle linéaire généralisé est donné par les commandes suivantes.

```
# Obtention du nom des deux composants de la liste
names(bc)

## [1] "x" "y"

bc$x[which.max(bc$y)]

## [1] 0.06060606
```

Il n'est pas nécessairement utile de réaliser la transformation de coefficient optimal  $\lambda = 0.0606061$ . Ici, le passage au logarithme *i.e.* une transformation de Box-Cox avec  $\lambda = 0$  suffira.

```
plot(log(AirPassengers),
main="Chronogramme de la série log(AirPassengers)")
```

### Chronogramme de la série $\log(\text{AirPassengers})$

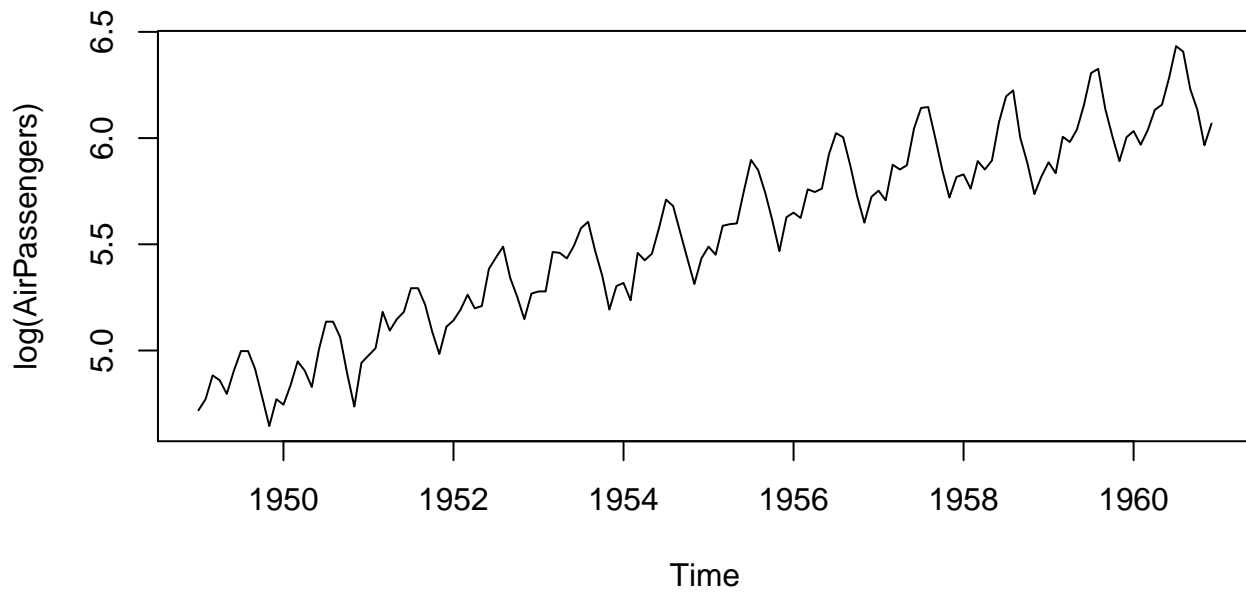
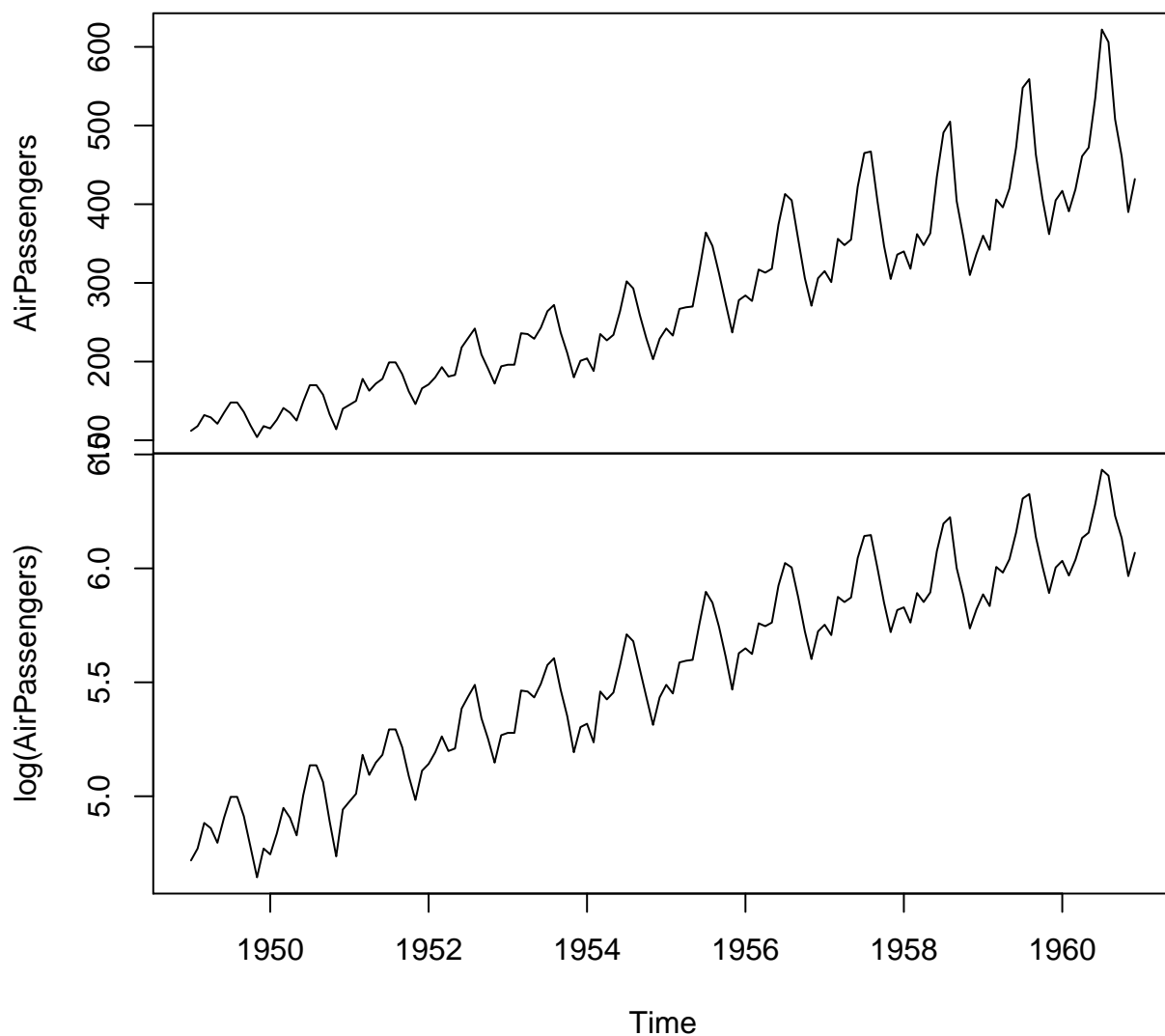


FIGURE 16.4 – Chronogramme de la série  $\log(\text{AirPassengers})$

```
x <- cbind(AirPassengers, log(AirPassengers))
# Par défaut, main="cbind(AirPassengers, log(AirPassengers))"
plot.ts(x, main="Chronogrammes des séries AirPassengers et log(AirPassengers)")
```

## Chronogrammes des séries AirPassengers et log(AirPassengers)



### 16.1.1.1 Détection d'une tendance (et d'une saisonnalité)

L'amplitude du cycle est devenue stable et on visualise à la fois la composante tendancielle et la composante saisonnière.

Nous supposons donc qu'il existe deux fonctions  $(m, s) \in \mathbb{R}^T \times \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \log(X_t) = m(t) + s(t) + B_t$$

Traçons la fonction d'autocorrélation empirique.

```
acf(log(AirPassengers)) # type="correlation" par défaut
```

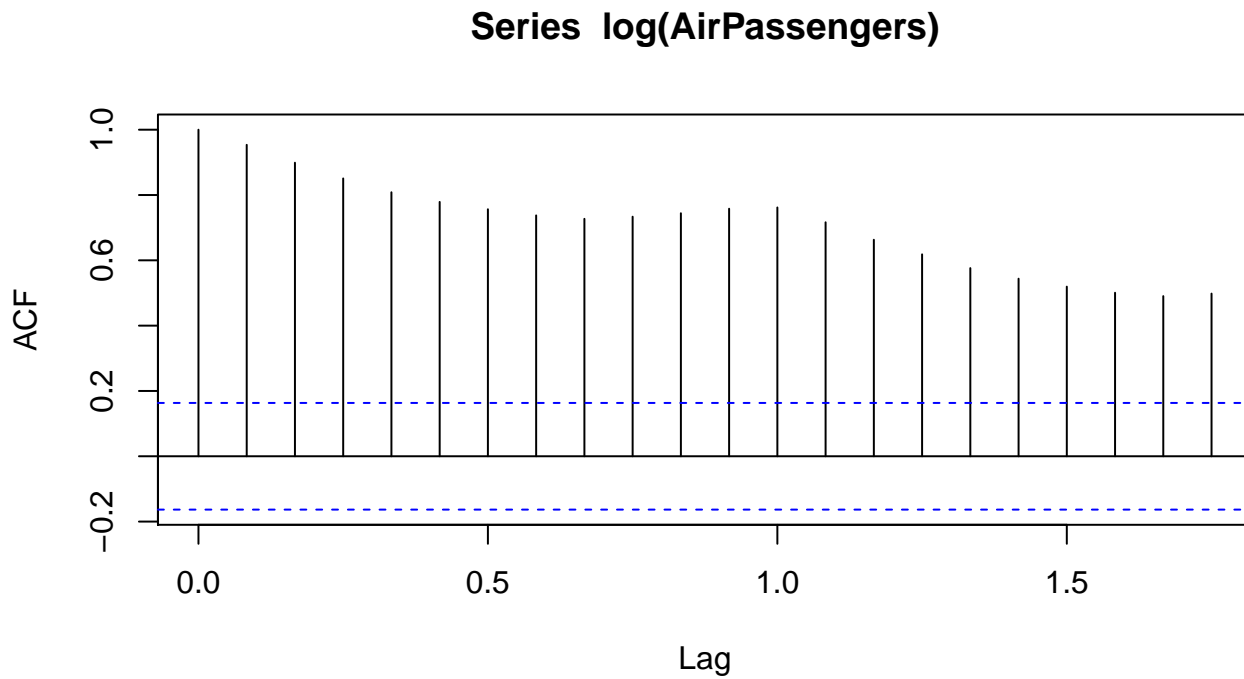


FIGURE 16.5 – Fonction d'autocorrélation empirique de la série  $\log(\text{AirPassengers})$

Le graphique de la fonction d'autocorrélation empirique met surtout en évidence la présence de la tendance.

### 16.1.1.2 Étude de la composante tendancielle

#### 16.1.1.2.1 Identification de la nature de la tendance

Rappelons que les objets suivants sont de classe `htest`.

```
library(tseries)
adf.test(log(AirPassengers))

## Warning in adf.test(log(AirPassengers)): p-value smaller than printed p-value
##
## Augmented Dickey-Fuller Test
##
## data: log(AirPassengers)
## Dickey-Fuller = -6.4215, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary

adf.test(diff(log(AirPassengers)))

## Warning in adf.test(diff(log(AirPassengers))): p-value smaller than printed p-value
##
## Augmented Dickey-Fuller Test
##
## data: diff(log(AirPassengers))
## Dickey-Fuller = -6.4313, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary

kpss.test(diff(log(AirPassengers)))

## Warning in kpss.test(diff(log(AirPassengers))): p-value greater than printed p-value
##
## KPSS Test for Level Stationarity
##
## data: diff(log(AirPassengers))
## KPSS Level = 0.028205, Truncation lag parameter = 4, p-value = 0.1
```

On accède directement aux p-valeurs avec les commandes suivantes.

```
# Obtention des noms des objets
names(adf.test(log(AirPassengers)))

## [1] "statistic" "parameter" "alternative" "p.value" "method"
## [6] "data.name"

names(kpss.test(diff(log(AirPassengers))))

## [1] "statistic" "parameter" "p.value" "method" "data.name"

# Extraction des p-valeurs
adf.test(log(AirPassengers))$p.value

## [1] 0.01

adf.test(diff(log(AirPassengers)))$p.value

## [1] 0.01

kpss.test(diff(log(AirPassengers)))$p.value

## [1] 0.1
```

La réponse aux tests est *Non, Non, Oui* donc une modélisation avec une tendance (déterministe) linéaire est préconisée. La fonction  $m$  est donc déterministe.

#### 16.1.1.2.2 Modélisation de la tendance (déterministe) linéaire

Comme attendu, si on différencie la série une fois, la fonction d'autocorrélation empirique montre que la composante de tendance semble avoir disparu.

```
acf(diff(log(AirPassengers)))
```

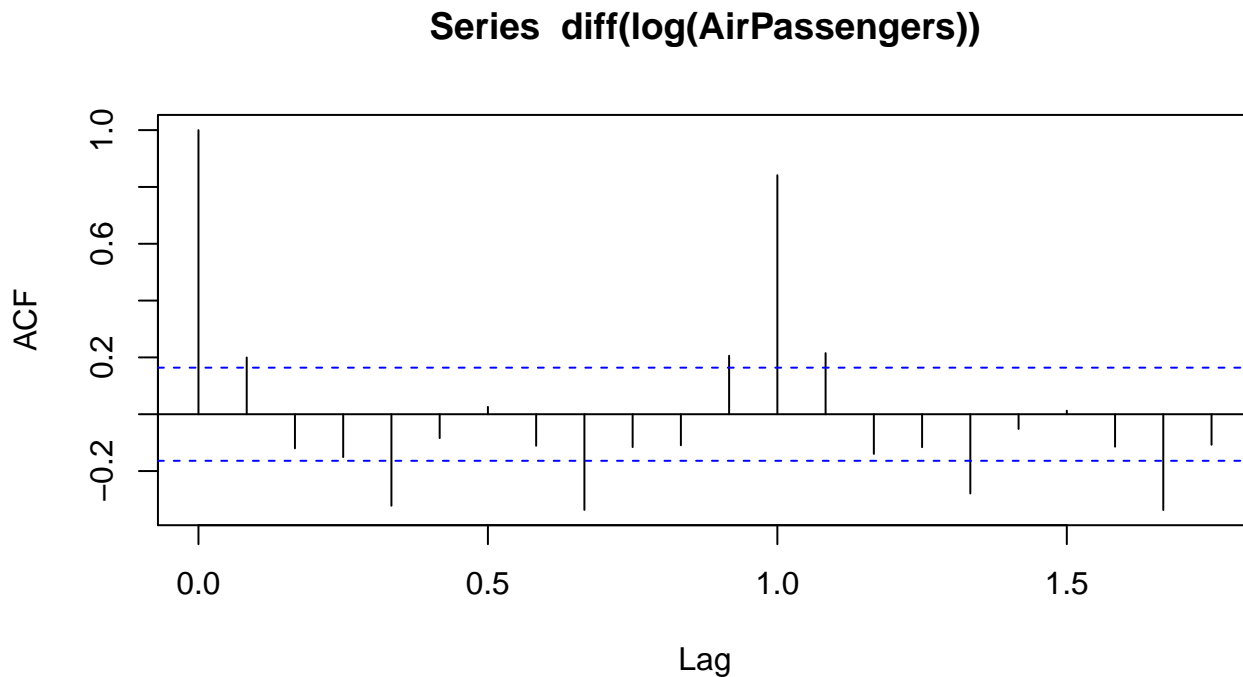


FIGURE 16.6 – Fonction d'autocorrélation empirique de la série `diff(log(AirPassengers))`

Une régression linéaire (de degré 1) semble effectivement suffire.

```
temps <- 1:length(log(AirPassengers))
lm1 <- lm(log(AirPassengers)~temps)
lm1

##
## Call:
## lm(formula = log(AirPassengers) ~ temps)
##
## Coefficients:
## (Intercept)      temps
##    4.81367      0.01005
```

On accède directement aux coefficients du modèle avec les commandes suivantes.

```
# Obtention des noms de l'objet
names(lm1)

## [1] "coefficients" "residuals" "effects" "rank"
## [5] "fitted.values" "assign" "qr" "df.residual"
## [9] "xlevels" "call" "terms" "model"

# Affichage des coefficients alpha_0=(Intercept) et alpha_1=temps
lm1$coefficients # ou plus simplement lm1$coef, class "numeric"
```

```
## (Intercept)      temps
## 4.81366828 0.01004838
```

Explicitons notre modélisation. Il existe une fonction  $s \in \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \log(X_t) = (4.8137 + 0.01t) + s(t) + B_t$$

Retenons les valeurs des coefficients  $\alpha_0$  et  $\alpha_1$ .

```
summary(lm1)

##
## Call:
## lm(formula = log(AirPassengers) ~ temps)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30858 -0.10388 -0.01796  0.09738  0.29538
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.8136683  0.0232940  206.65  <2e-16 ***
## temps       0.0100484  0.0002787   36.05  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.139 on 142 degrees of freedom
## Multiple R-squared:  0.9015, Adjusted R-squared:  0.9008
## F-statistic: 1300 on 1 and 142 DF,  p-value: < 2.2e-16

summary(lm1)$coefficients # class "matrix"

##              Estimate Std. Error  t value      Pr(>|t|)
## (Intercept) 4.81366828 0.023294000 206.64842 5.927796e-178
## temps       0.01004838 0.000278732  36.05034 2.405115e-73

summary(lm1)$coefficients[1,1] # class "numeric"

## [1] 4.813668

summary(lm1)$coefficients[2,1]

## [1] 0.01004838

alpha_0 <- summary(lm1)$coefficients[1,1]
alpha_1 <- summary(lm1)$coefficients[2,1]
```

Remarquer la différence avec la commande suivante.

```
lm1$coefficients[1] # class "numeric"

## (Intercept)
##      4.813668
```

Effectuons le test de non corrélation de Breusch-Godfrey.

```
library(lmtest)
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##   as.Date, as.Date.numeric

bgtest(lm1) # class "bgtest" "htest"

##
## Breusch-Godfrey test for serial correlation of order up to 1
##
## data:   lm1
## LM test = 71.054, df = 1, p-value < 2.2e-16

names(bgtest(lm1))

## [1] "statistic"      "parameter"      "method"          "p.value"         "data.name"
## [6] "coefficients"   "vcov"

bgtest(lm1)$p.value

## [1] 3.476104e-17
```

On a  $p\text{-value} < 0,05$  donc les résidus sont corrélés ou encore il reste de l'autocorrélation dans les résidus. On ne peut donc pas utiliser le modèle `lm1` pour effectuer des prévisions. Affinons le modèle en modélisant la série résiduelle  $(s(t) + B_t)_{t \in T}$ .

### 16.1.1.3 Étude de la composante saisonnière

```
residus <- lm1$residuals # ou plus simplement lm1$res, class "numeric"  
residus <- as.ts(residus)
```

```
split.screen(2:1)  
  
## [1] 1 2  
  
screen(1) ; plot(residus, main="Chronogramme de la série residus")  
screen(2) ; acf(residus)
```

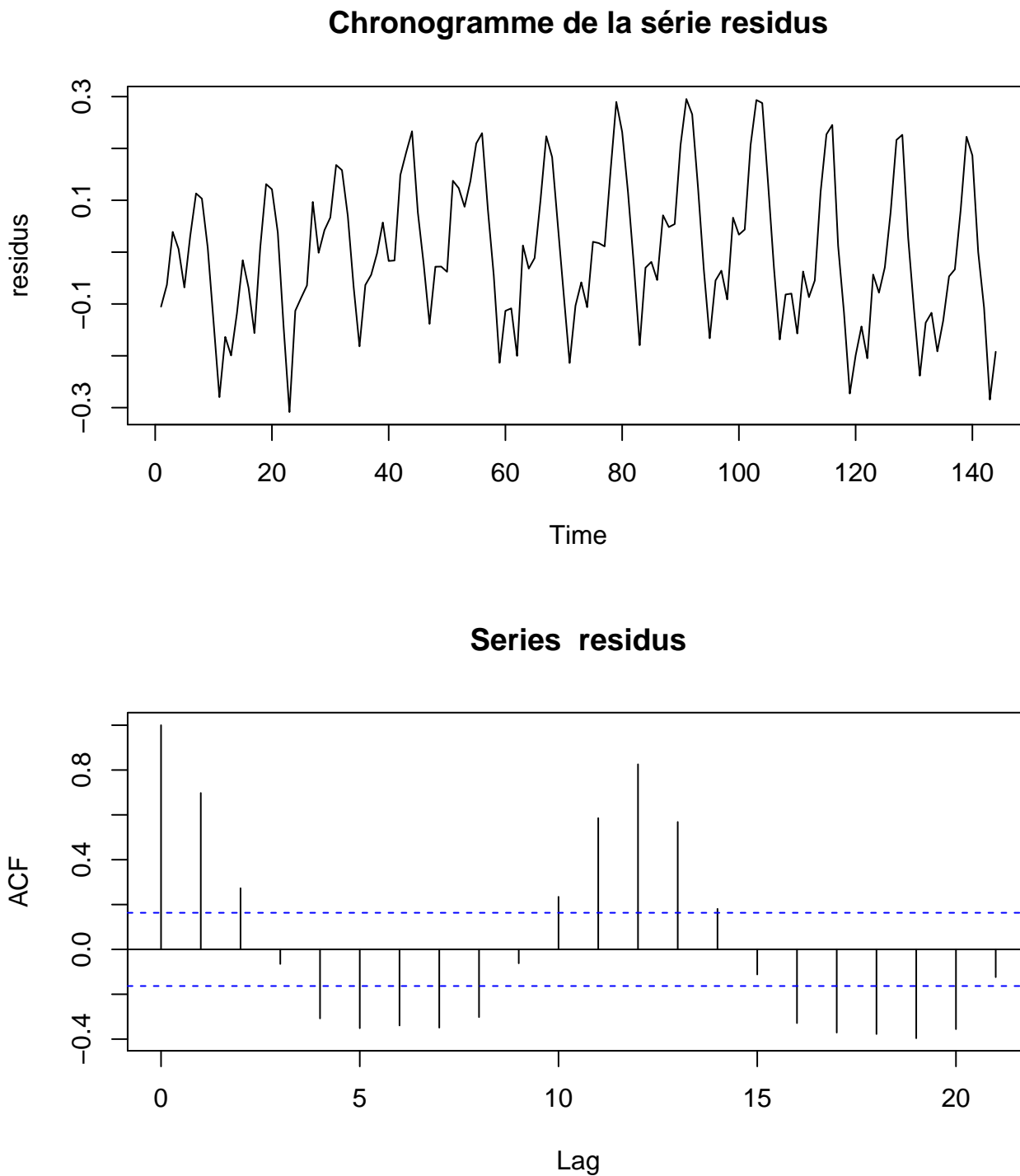


FIGURE 16.7 – Chronogramme et fonction d'autocorrélation empirique de la série **residus**

D'après la figure 16.7, il faut maintenant étudier la composante saisonnière.

#### 16.1.1.3.1 Identification de la saisonnalité

```
periodogram(residus)
abline(v=1/12, col="red", lty=2, lwd=2)
```

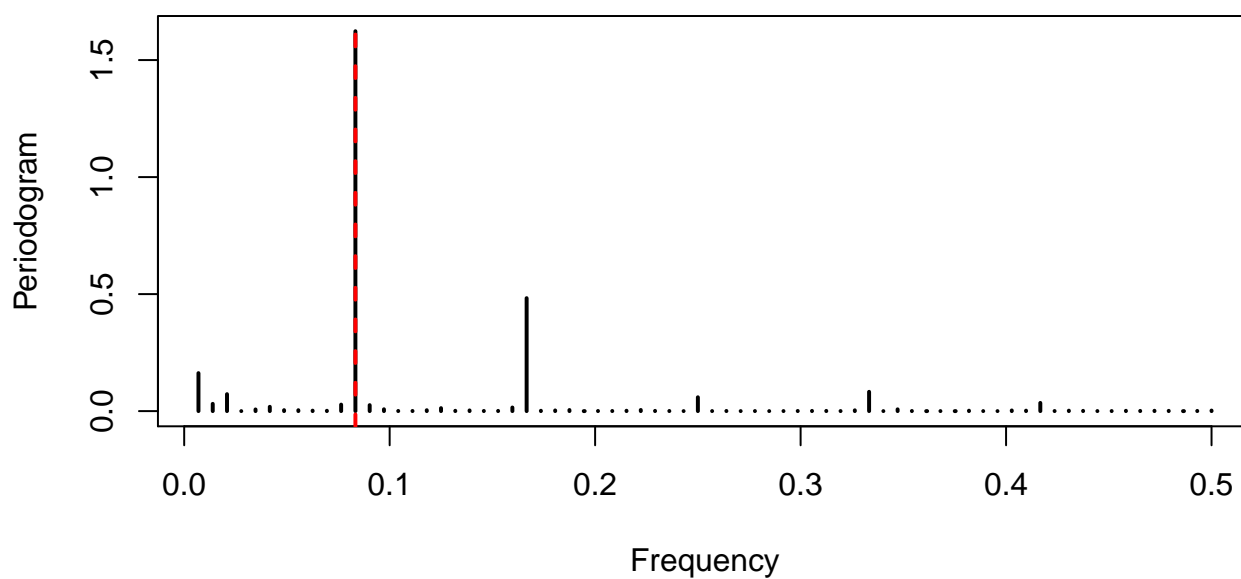


FIGURE 16.8 – Périodogramme de la série `residus`

Le périodogramme de la figure 16.8 détecte la période  $r = 12$ .

#### 16.1.1.3.2 Validation de la saisonnalité

Validons cette période en utilisant le test issu du *Friedman Rank Sum Test*.

```
Saison.test(residus,12)
```

##	Période	d.f.	Tobs	p-valeur
##	12.0000	11.0000	119.6026	0.0000

On a  $p\text{-value} < 0,05$  donc la saisonnalité  $r = 12$  est validée.

### 16.1.1.3.3 Identification de la nature de la saisonnalité

Appliquons les tests de racine unitaire saisonnière pour savoir si la modélisation doit se faire avec une saisonnalité déterministe ou stochastique.

```
residus <- ts(residus, frequency=12)
library(forecast) # fonction nsdiffs()
# Le test de Canova-Hasen (CH) nécessite le package uroot
# install.packages("uroot")
# nsdiffs(residus, test="ch", m=12)
nsdiffs(residus, test="ocsb", m=12)

## Warning: argument m is deprecated; please set the frequency in the ts object.
## [1] 0
```

La saisonnalité n'est donc pas stochastique mais plutôt déterministe. La fonction  $s$  est donc déterministe. Explicitons notre modèle. Il existe  $(s_1, \dots, s_{12}) \in \mathbb{R}^{12}$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \log(X_t) = (4,812 + 0,01t) + \sum_{i=1}^{12} s_i \chi_{i+12\mathbb{Z}}(t) + B_t$$

### 16.1.1.3.4 Modélisation de la saisonnalité déterministe

#### 16.1.1.3.4.1 Par moyennes saisonnières

Modélisons avec un processus des moyennes saisonnières.

```
mois. <- season(residus)
lmMS <- lm(residus~mois.-1) # class "lm"
lmMS

##
## Call:
## lm(formula = residus ~ mois. - 1)
##
## Coefficients:
##   mois.January   mois.February   mois.March   mois.April   mois.May
##   -0.085520    -0.107554      0.022694   -0.008555   -0.010907
##   mois.June     mois.July     mois.August  mois.September  mois.October
##    0.111260     0.215222     0.205948     0.061334    -0.076804
##   mois.November  mois.December
##   -0.220501     -0.106616
```

On accède directement aux coefficients du modèle avec les commandes suivantes.

```
names(lmMS)

## [1] "coefficients" "residuals" "effects" "rank"
## [5] "fitted.values" "assign" "qr" "df.residual"
## [9] "contrasts" "xlevels" "call" "terms"
## [13] "model"

lmMS$coefficients # ou plus simplement lmMS$coef, class "numeric"

##   mois.January   mois.February   mois.March   mois.April   mois.May
##   -0.08551957   -0.10755397   0.02269357   -0.00855486   -0.01090708
##   mois.June     mois.July     mois.August  mois.September  mois.October
##    0.11125955    0.21522230    0.20594788    0.06133370    -0.07680412
##   mois.November  mois.December
##   -0.22050140   -0.10661599
```

#### 16.1.1.3.4.1.1 Analyse des résidus

Valisons le modèle par stationnarité des résidus.

```
kpss.test(lmMS$res) # class "htest"

##
## KPSS Test for Level Stationarity
##
## data: lmMS$res
## KPSS Level = 0.39068, Truncation lag parameter = 4, p-value = 0.08117

names(kpss.test(lmMS$res))

## [1] "statistic" "parameter" "p.value" "method" "data.name"

kpss.test(lmMS$res)$p.value

## [1] 0.08117365
```

Les résidus sont tout justes stationnaires. Le modèle issu des moyennes saisonnières est donc valide.

Vérifions que les conditions du test de régression sont valides à l'aide des tests de Breusch-Godfrey et de Harrison-McCabe.

```
hmctest(lmMS) # class "htest"

##
## Harrison-McCabe test
##
## data: lmMS
## HMC = 0.51691, p-value = 0.61

names(hmctest(lmMS))

## [1] "statistic" "method" "p.value" "data.name"

hmctest(lmMS)$p.value

## [1] 0.607
```

On a  $p\text{-value} > 0,05$  donc les résidus semblent avoir une variance constante.

```
bgtest(lmMS) # class "htest" "bgtest"

##
## Breusch-Godfrey test for serial correlation of order up to 1
##
## data: lmMS
## LM test = 88.944, df = 1, p-value < 2.2e-16

names(bgtest(lmMS))

## [1] "statistic" "parameter" "method" "p.value" "data.name"
## [6] "coefficients" "vcov"

bgtest(lmMS)$p.value

## [1] 4.061024e-21
```

On a  $p\text{-value} < 0,05$  donc les résidus semblent corrélés. On doit donc modéliser la série résiduelle  $(B_t)_{t \in T}$ .  
Modélisons les résidus.

Les résidus ne forment pas un bruit blanc (test de blancheur).  
Cherchons une modélisation avec un ARMA(p,q).

```
eacf(lmMS$res)

## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x x x x x x
## 1 x o o x o o o o o o o x o o
## 2 x o o x o o o o o o o o o o
## 3 o o o o o o o o o o o o o o
## 4 o x x o o o o o o o o o o o
## 5 x x x x o o o o o o o o o o
## 6 x o x x x o o x o o o o o o
## 7 x o x x o o o x o o o o o o
```

L'eacf suggère un modèle ARMA(1,4) pour les résidus.

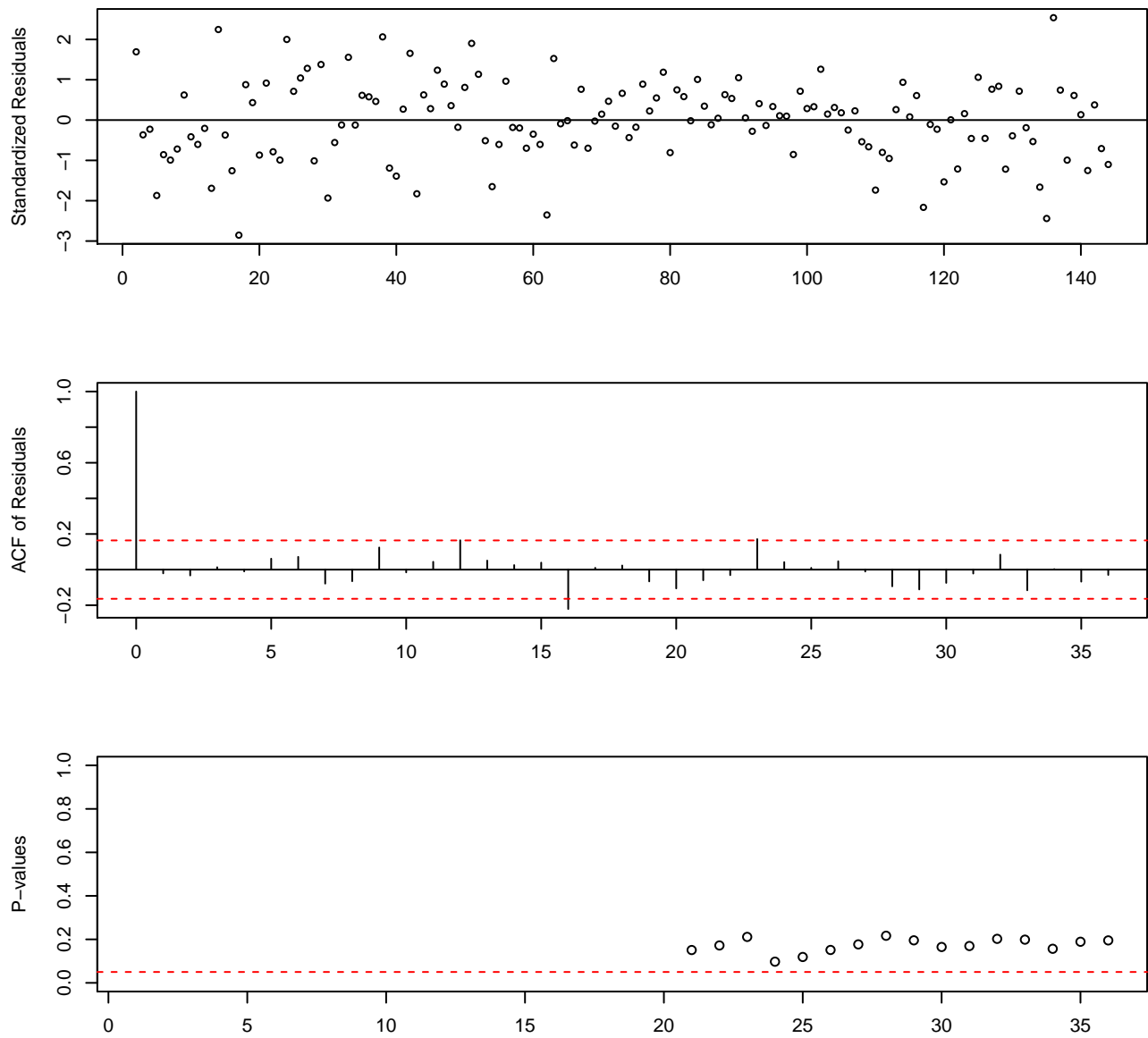
```
modMS_auto <- auto.arima(lmMS$res, d=0, max.p=7, max.q=13, ic="aic")
modMS_auto

## Series: lmMS$res
## ARIMA(2,0,0) with zero mean
##
## Coefficients:
##           ar1      ar2
##          0.6756  0.1411
## s.e.    0.0830  0.0834
##
## sigma^2 estimated as 0.001217:  log likelihood=279.38
## AIC=-552.75   AICc=-552.58   BIC=-543.84
```

```
# class "forecast_ARIMA" "ARIMA" "Arima"
modMS <- Arima(lmMS$res, order=c(1,0,4), include.mean=F)
```

Validons ce modèle.

```
tsdiag.Arima(modMS, gof.lag=round(length(log(AirPassengers))/4))
```



Voyons si le modèle est simplifiable.

```
t_stat(modMS) # class "matrix"

##          ar1          ma1          ma2          ma3          ma4
## t.stat 21.93896 -2.476006 0.606356 -1.856813 -2.566067
## p.val   0.00000  0.013286 0.544278  0.063338  0.010286
```

Le modèle n'est pas simplifiable.

Explicitons notre modèle. Il existe  $(s_1, \dots, s_{12}) \in \mathbb{R}^{12}$  tel que,

$$\forall t \in T \quad \log(X_t) = (4.8137 + 0.01t) + \sum_{i=1}^{12} s_i \chi_{i+12\mathbb{Z}}(t) + B_t$$

avec  $(B_t)_{t \in T}$  un ARMA(1,4).

#### 16.1.1.3.4.1.2 Modèle complet SARIMA(,1,)(1,0,4)[12] avec fonction Arima()

Pour construire le modèle complet, il faut préciser dans l'argument `xreg` de la fonction `Arima()` que l'estimation de la composante saisonnière déterministe se fait par moyennes saisonnières. Du coup, la constante est contenue dans la saison, d'où l'argument `include.mean=F`.

```
MatInd <- diag(1,12)
MatMS <- MatInd
colnames(MatMS) <- paste("s'", 1:12, sep="")
nbA <- length(log(AirPassengers))/12
nbA

## [1] 12

for (i in 1:(nbA-1)) {
  MatMS <- rbind(MatMS, MatInd)
}
Mat1a <- cbind(temps, MatMS)

modeleM1a <- Arima(AirPassengers, xreg=Mat1a, order=c(1,0,4), include.mean=F, lambda=0)
modeleM1a

## Series: AirPassengers
## Regression with ARIMA(1,0,4) errors
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1          ma1          ma2          ma3          ma4 temps      s'1      s'2      s'3
##          0.9464 -0.2395  0.0582 -0.1692 -0.2288 1e-02  4.7243  4.7025  4.8340
## s.e.    0.0424  0.0952  0.0948  0.0897  0.0895 4e-04  0.0371  0.0372  0.0373
##          s'4      s'5      s'6      s'7      s'8      s'9      s'10     s'11     s'12
##          4.8038  4.8017  4.9240  5.0282  5.0191  4.8746  4.7357  4.5921  4.7071
## s.e.    0.0373  0.0374  0.0376  0.0377  0.0378  0.0379  0.0379  0.0377  0.0376
##
## sigma^2 estimated as 0.001294: log likelihood=283.42
## AIC=-528.84 AICc=-522.72 BIC=-472.42
```

On accède directement aux coefficients et à la variance avec les commandes suivantes.

```
names(modeleM1a)

## [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
## [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
## [13] "nobs"      "model"     "aicc"      "bic"       "xreg"      "lambda"
## [19] "x"         "fitted"

modeleM1a$coef # class "numeric"

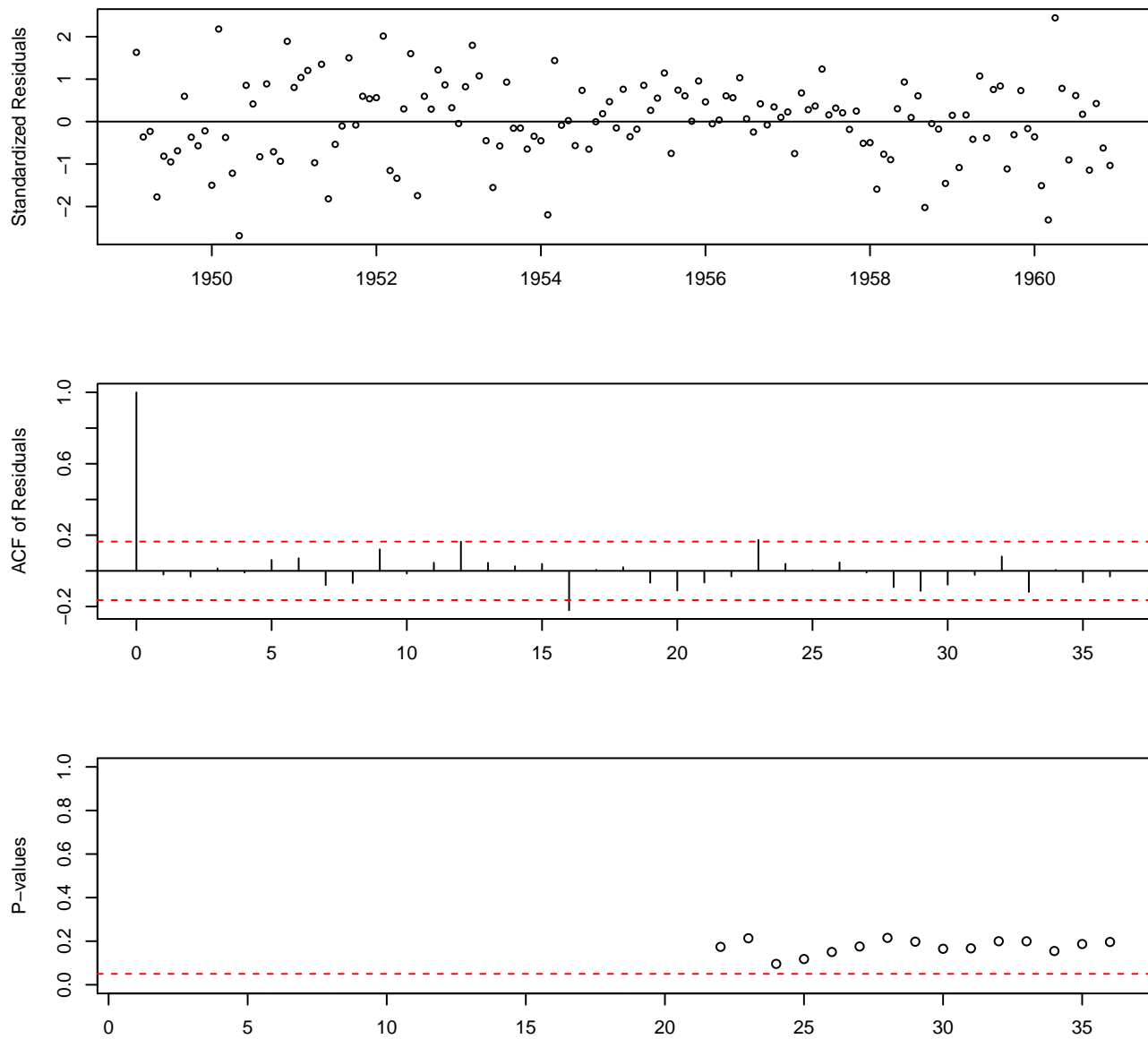
##          ar1          ma1          ma2          ma3          ma4          temps
## 0.946355765 -0.239520048  0.058206774 -0.169176500 -0.228792813  0.009963636
##          s'1          s'2          s'3          s'4          s'5          s'6
## 4.724279244 4.702477925  4.834043212  4.803787988  4.801658114  4.924027007
##          s'7          s'8          s'9          s'10         s'11         s'12
## 5.028172997 5.019061583  4.874588055  4.735714642  4.592128835  4.707086116

modeleM1a$sigma2

## [1] 0.001293907
```

Rappelons que, pour tout  $i \in \{1, \dots, 12\}$ ,  $s_i = s'_i - \alpha_0$ .  
Vérifions la validité du modèle.

```
tsdiag.Arima(modeleM1a, gof.lag=round(length(log(AirPassengers))/4))
```



Vérifions que le modèle n'est pas simplifiable.

```
t_stat(modeleM1a) # class "matrix"

##          ar1          ma1          ma2          ma3          ma4      temps      s'1
## t.stat 22.29841 -2.516658 0.614317 -1.885739 -2.557508 23.8427 127.2243
## p.val  0.00000 0.011847 0.539006 0.059330 0.010543 0.0000 0.0000
##          s'2          s'3          s'4          s'5          s'6          s'7          s'8          s'9
## t.stat 126.3862 129.7387 128.6889 128.2303 131.1054 133.4994 132.9024 128.7568
## p.val  0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
```

```
##          s'10      s'11      s'12
## t.stat 125.0698 121.6887 125.1261
## p.val   0.0000   0.0000   0.0000
```

Analysons les résidus. Testons la normalité des résidus pour expliciter le modèle complet.

```
shapiro.test(modeleM1a$res) # class "htest"

##
## Shapiro-Wilk normality test
##
## data:  modeleM1a$res
## W = 0.99458, p-value = 0.8687

names(shapiro.test(modeleM1a$res))

## [1] "statistic" "p.value" "method" "data.name"

shapiro.test(modeleM1a$res)$p.value

## [1] 0.8686973


library(fBasics)

## Loading required package: timeDate
## Loading required package: timeSeries
##
## Attaching package: 'timeSeries'
## The following object is masked from 'package:zoo':
##
## time<-

dagoTest(modeleM1a$res) # class "fHTEST" attr(,"package") "fBasics"

##
## Title:
## D'Agostino Normality Test
##
## Test Results:
## STATISTIC:
## Chi2 | Omnibus: 1.4323
## Z3 | Skewness: -1.0182
## Z4 | Kurtosis: 0.629
## P VALUE:
## Omnibus Test: 0.4886
## Skewness Test: 0.3086
## Kurtosis Test: 0.5293
##
## Description:
## Sun May 8 12:33:06 2022 by user:

dagoTest(modeleM1a$res)$test # class "list"

## $statistic
## Chi2 | Omnibus Z3 | Skewness Z4 | Kurtosis
## 1.4323435 -1.0181705 0.6290248
##
```

```
## $method
## [1] "D'Agostino Omnibus Normality Test"
##
## $p.value
## Omnibus Test Skewness Test Kurtosis Test
##      0.4886192      0.3085969      0.5293328
##
## $data.name
## [1] "modeleM1a$res"

dagoTest(modeleM1a$res)$test$p.value # class "numeric"

## Omnibus Test Skewness Test Kurtosis Test
##      0.4886192      0.3085969      0.5293328

dagoTest(modeleM1a$res)$test$p.value[1]

## Omnibus Test
##      0.4886192
```

Les tests de Shapiro-Wilk et d'Agostino valident la normalité des résidus.  
 Explicitons notre modèle final.

$$\forall t \in T \quad \log(X_t) = m(t) + s(t) + B_t$$

avec

- $m(t) = 4.8137 + 0.01t$
- $s(t) = \sum_{i=1}^r s_i \chi_{i+r\mathbb{Z}} = \sum_{i=1}^r (s'_i - \alpha_0) \chi_{i+r\mathbb{Z}} = \sum_{i=1}^{12} (\text{coef test}(\text{modeleM1a})[6+i, 1] - \alpha_0) \chi_{i+12\mathbb{Z}}$   
 où
 

$s_1 = -0.0894$	$s_2 = -0.1112$	$s_3 = 0.0204$	$s_4 = -0.0099$
$s_5 = -0.012$	$s_6 = 0.1104$	$s_7 = 0.2145$	$s_8 = 0.2054$
$s_9 = 0.0609$	$s_{10} = -0.078$	$s_{11} = -0.2215$	$s_{12} = -0.1066$
- $(B_t)_{t \in T}$  un ARMA(1,4) défini par

$$\begin{aligned} B_t &= \sum_{k=1}^p \varphi_k B_{t-k} + \sum_{k=0}^q \theta_k \varepsilon_{t-k} = \varphi_1 B_{t-1} + \varepsilon_t + \sum_{k=1}^4 \theta_k \varepsilon_{t-k} \\ &= \text{coef test}(\text{modeleM1a})[1, 1] B_{t-1} + \varepsilon_t + \sum_{k=1}^4 \text{coef test}(\text{modeleM1a})[k+1, 1] \varepsilon_{t-k} \end{aligned}$$

où

$$\varphi_1 = 0.9464 \quad \theta_1 = -0.2395 \quad \theta_2 = 0.0582 \quad \theta_3 = -0.1692 \quad \theta_4 = -0.2288$$

et

$$\varepsilon_t \sim \mathcal{N}(0, 0.0013)$$

```
coef test(modeleM1a)

##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1    0.94635576 0.04244051 22.2984 < 2e-16 ***
## ma1   -0.23952005 0.09517387 -2.5167 0.01185 *
## ma2    0.05820677 0.09475043  0.6143 0.53901
## ma3   -0.16917650 0.08971363 -1.8857 0.05933 .
## ma4   -0.22879281 0.08945928 -2.5575 0.01054 *
## temps  0.00996364 0.00041789 23.8427 < 2e-16 ***
```

```
## s'1      4.72427924  0.03713348 127.2243 < 2e-16 ***
## s'2      4.70247793  0.03720722 126.3862 < 2e-16 ***
## s'3      4.83404321  0.03725985 129.7386 < 2e-16 ***
## s'4      4.80378799  0.03732869 128.6889 < 2e-16 ***
## s'5      4.80165811  0.03744558 128.2303 < 2e-16 ***
## s'6      4.92402701  0.03755777 131.1054 < 2e-16 ***
## s'7      5.02817300  0.03766438 133.4994 < 2e-16 ***
## s'8      5.01906158  0.03776502 132.9024 < 2e-16 ***
## s'9      4.87458806  0.03785888 128.7568 < 2e-16 ***
## s'10     4.73571464  0.03786458 125.0698 < 2e-16 ***
## s'11     4.59212884  0.03773669 121.6887 < 2e-16 ***
## s'12     4.70708612  0.03761874 125.1261 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### 16.1.1.3.4.1.3 Prévisions

```
# time(log(AirPassengers))=time(AirPassengers), class "ts"
end(time(log(AirPassengers)))

## [1] 1960    12
```

Remarquer la différence avec la commande suivante.

```
time(log(AirPassengers))[length(time(log(AirPassengers)))]

## [1] 1960.917

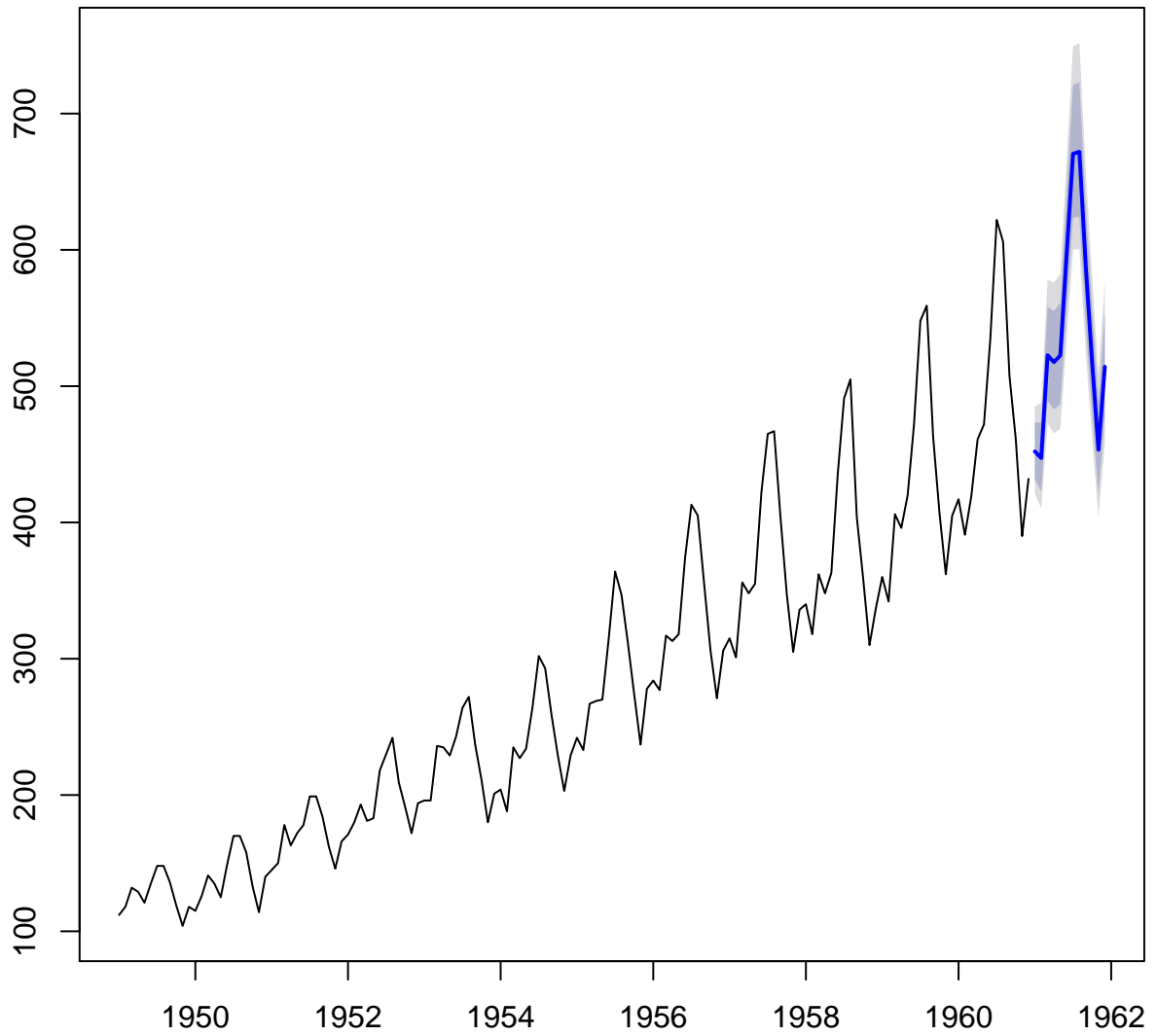
tempsP <- (length(log(AirPassengers))+1):(length(log(AirPassengers))+12)
Mat1aP <- cbind(tempsP, MatInd)
prevM1a <- forecast(modeleM1a, xreg=Mat1aP, h=12)

## Warning in forecast.forecast_ARIMA(modeleM1a, xreg = Mat1aP, h = 12): xreg contains different
column names from the xreg used in training. Please check that the regressors are in the same order.
```

La commande suivante doit être insérée dans une cellule à part.

```
plot(prevM1a)
```

### Forecasts from Regression with ARIMA(1,0,4) errors



#### 16.1.1.3.4.2 Par régression harmonique 1

**Remarque** Le package TSA n'est plus disponible, il faut extraire le code source de la fonction `harmonic()`.

```
har. <- harmonic(residus,1) # class "matrix", dim 144 2 cos(2*pi*t) sin(2*pi*t)
lmRH <- lm(residus~har.)
lmRH

##
## Call:
## lm(formula = residus ~ har.)
##
## Coefficients:
##      (Intercept)  har.cos(2*pi*t)  har.sin(2*pi*t)
##      -3.074e-17      -1.475e-01      2.808e-02
```

##### 16.1.1.3.4.2.1 Analyse des résidus

Assurons-nous que les conditions du test de régression sont vérifiées.

```
kpss.test(lmRH$res)

##
## KPSS Test for Level Stationarity
##
## data:  lmRH$res
## KPSS Level = 0.35844, Truncation lag parameter = 4, p-value = 0.09507

names(kpss.test(lmRH$res))

## [1] "statistic" "parameter" "p.value" "method" "data.name"

kpss.test(lmRH$res)$p.value

## [1] 0.09506764
```

Les résidus sont stationnaires.

```
bgtest(lmRH)

##
## Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lmRH
## LM test = 32.386, df = 1, p-value = 1.264e-08

names(bgtest(lmRH))

## [1] "statistic" "parameter" "method" "p.value" "data.name"
## [6] "coefficients" "vcov"

bgtest(lmRH)$p.value

## [1] 1.264102e-08
```

On a  $p\text{-value} < 0,05$  donc les résidus sont corrélés. La série résiduelle ne forme donc pas un bruit blanc. Cherchons une modélisation avec un ARMA(p,q).

```
eacf(lmRH$res, ar.max=10, ma.max=15)

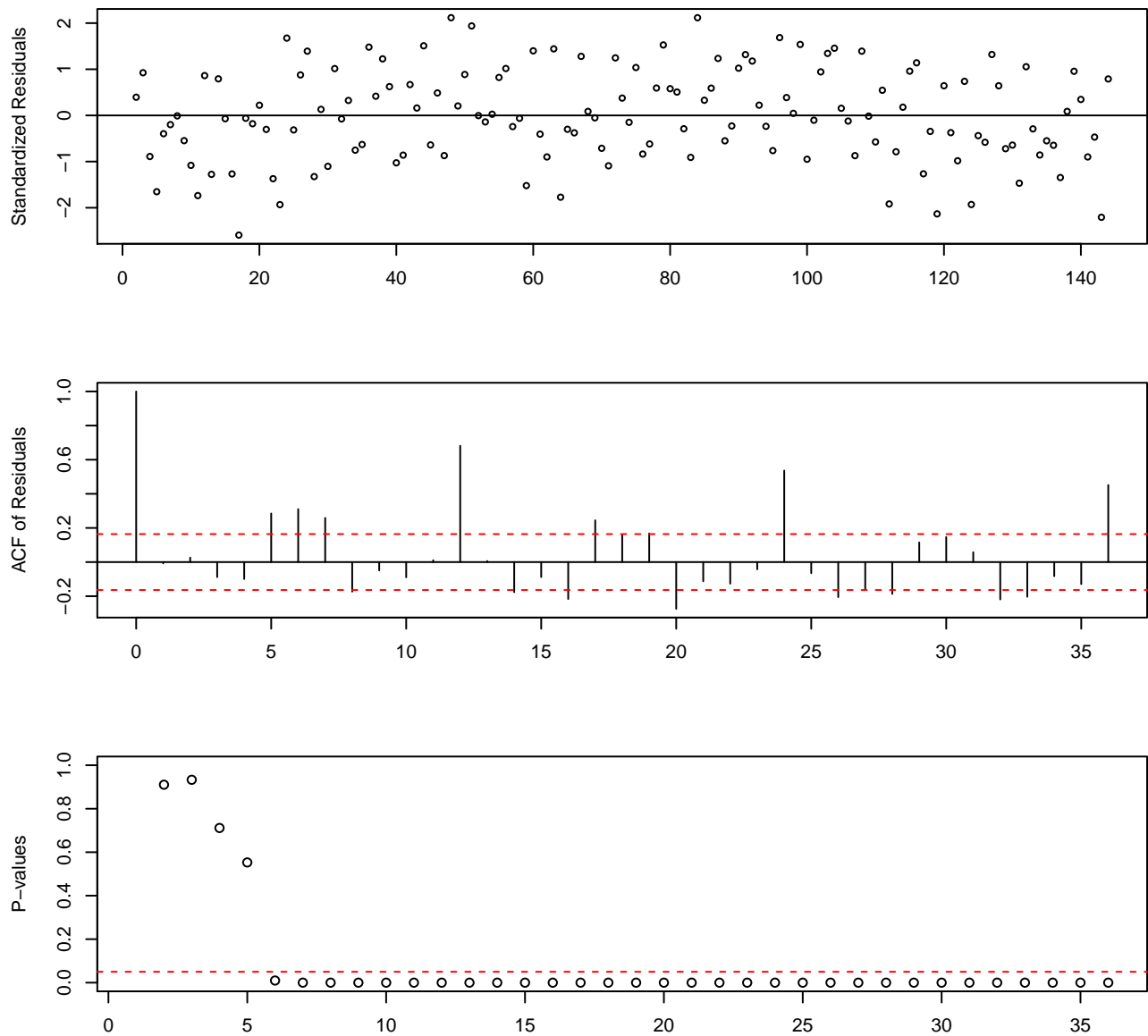
## AR/MA
##    0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
## 0  x o o o x x x o x o x x x x x o
## 1  x o o o x x x x o o o x x x x o
## 2  o x o o o o x x o o o x o x x o
## 3  x o x o o o x o x o o x x o x o
## 4  o o x x o o o o o o o x x o x o
## 5  x o x x o o o o o o o x o o x x
## 6  x o x o o x o o o o o x x o x o
## 7  x x x o o x o o o o o x o o x o
## 8  o x x o o x o o o o o x x x x o
## 9  o x x x o x x o o o o x o x o o
## 10 x x x x x x x o o o x o x o o o
```

L'eacf est difficilement interprétable. Parmi les modèles qui pourraient être suggérés, c'est-à-dire ARMA(2,2), ARMA(4,4) ou ARMA(6,6), aucun n'est valide.

```
modRH_auto <- auto.arima(lmRH$res)
modRH_auto

## Series: lmRH$res
## ARIMA(0,0,1) with zero mean
##
## Coefficients:
##          ma1
##          0.6668
## s.e.    0.0745
##
## sigma^2 estimated as 0.005468:  log likelihood=170.92
## AIC=-337.84   AICc=-337.75   BIC=-331.9

tsdiag.Arima(modRH_auto, gof.lag=round(length(log(AirPassengers))/4))
```

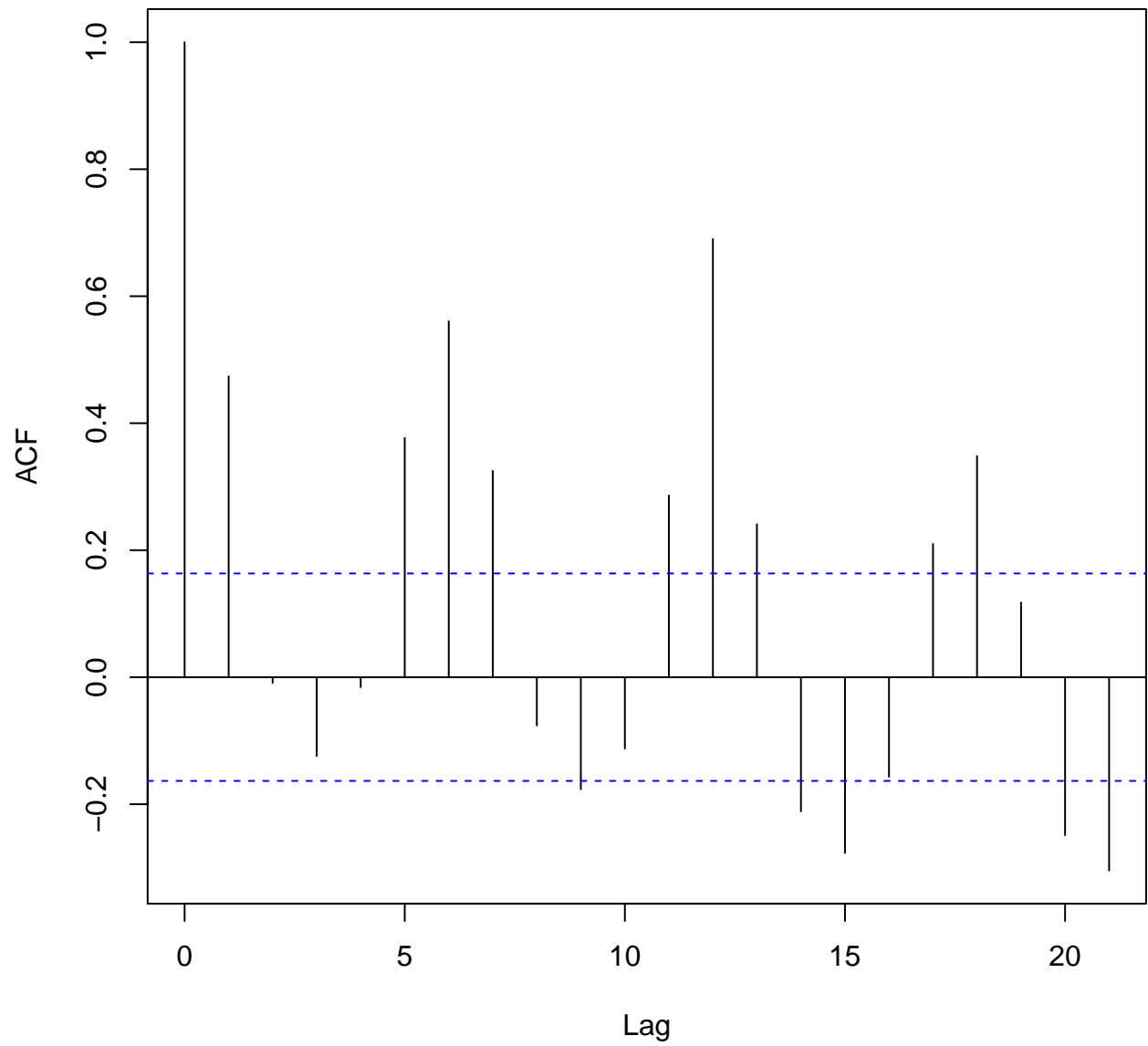


Même le modèle MA(1) suggéré par la procédure `auto.arima()` n'est pas valide.

Même les modèles suggérés par la procédure MINIC ne sont pas valides. En réalité, il reste de la saisonnalité qui n'a pas été captée par la régression harmonique.

```
acf(lmRH$res)
```

### Series ImRH\$res



Par défaut, la procédure `harmonic()` du package `TSA` ne considère comme variables explicatives que les fonctions trigonométriques  $\cos\left(\frac{2k\pi}{r}\right)$  et  $\sin\left(\frac{2k\pi}{r}\right)$  qui correspondent ici à la période  $r = 12$ .

#### 16.1.1.3.4.3 Par régression harmonique 2

On recalcule la régression harmonique mais en autorisant les fonctions trigonométriques associées aux périodes 6 et 4.

```
har. <- harmonic(residus, m=3) # class "matrix", dim 144 2 cos(2*pi*t) sin(2*pi*t)
lmRH <- lm(residus~har.)
lmRH

##
## Call:
## lm(formula = residus ~ har.)
##
## Coefficients:
##      (Intercept)  har.cos(2*pi*t)  har.cos(4*pi*t)  har.cos(6*pi*t)
##      -2.075e-17      -1.475e-01      5.671e-02      -8.751e-03
## har.sin(2*pi*t)  har.sin(4*pi*t)  har.sin(6*pi*t)
##      2.808e-02      5.906e-02      -2.731e-02
```

```
summary(lmRH)

##
## Call:
## lm(formula = residus ~ har.)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.165994 -0.043319  0.002469  0.045136  0.155587
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.075e-17  5.421e-03   0.000 1.000000
## har.cos(2*pi*t) -1.475e-01  7.666e-03 -19.241 < 2e-16 ***
## har.cos(4*pi*t)  5.671e-02  7.666e-03   7.398 1.26e-11 ***
## har.cos(6*pi*t) -8.751e-03  7.666e-03  -1.142 0.255634
## har.sin(2*pi*t)  2.808e-02  7.666e-03   3.663 0.000355 ***
## har.sin(4*pi*t)  5.906e-02  7.666e-03   7.704 2.39e-12 ***
## har.sin(6*pi*t) -2.731e-02  7.666e-03  -3.563 0.000505 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06505 on 137 degrees of freedom
## Multiple R-squared:  0.7888, Adjusted R-squared:  0.7796
## F-statistic: 85.29 on 6 and 137 DF, p-value: < 2.2e-16
```

```
summary(lmRH)$coefficients

##              Estimate Std. Error      t value      Pr(>|t|)
## (Intercept) -2.074573e-17  0.005420764 -3.827086e-15 1.000000e+00
## har.cos(2*pi*t) -1.475076e-01  0.007666118 -1.924150e+01 9.089138e-41
## har.cos(4*pi*t)  5.671184e-02  0.007666118  7.397726e+00 1.256558e-11
## har.cos(6*pi*t) -8.751236e-03  0.007666118 -1.141547e+00 2.556336e-01
## har.sin(2*pi*t)  2.808113e-02  0.007666118  3.663018e+00 3.551817e-04
## har.sin(4*pi*t)  5.906093e-02  0.007666118  7.704151e+00 2.386649e-12
## har.sin(6*pi*t) -2.731260e-02  0.007666118 -3.562768e+00 5.054746e-04
```

#### 16.1.1.3.4.3.1 Analyse des résidus

Assurons-nous que les conditions du test de régression sont vérifiées.

```
kpss.test(lmRH$res)

##
## KPSS Test for Level Stationarity
##
## data:  lmRH$res
## KPSS Level = 0.38722, Truncation lag parameter = 4, p-value = 0.08266

names(kpss.test(lmRH$res))

## [1] "statistic" "parameter" "p.value" "method" "data.name"

kpss.test(lmRH$res)$p.value

## [1] 0.08266356
```

Les résidus sont stationnaires.

```
bgtest(lmRH)

##
## Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lmRH
## LM test = 35.615, df = 1, p-value = 2.405e-09

names(bgtest(lmRH))

## [1] "statistic" "parameter" "method" "p.value" "data.name"
## [6] "coefficients" "vcov"

bgtest(lmRH)$p.value

## [1] 2.40485e-09
```

On a  $p\text{-value} < 0,05$  donc les résidus sont corrélés. La série résiduelle ne forme donc pas un bruit blanc. Cherchons une modélisation avec un ARMA(p,q).

```
eacf(lmRH$res, ar.max=10, ma.max=15)

## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
## 0  x x x x x x x o x x o  x  o  o  x  o
## 1  x o x x o x o o x o o  x  x  o  x  o
## 2  x o x x o x o x x o x  x  x  o  x  o
## 3  x x o x o o o x o o o  x  x  x  o  o
## 4  o x x x o o o x o o o  x  o  o  o  o
## 5  o x o x x o o x o o o  x  o  o  o  x
## 6  x x o x o o o x o o o  x  o  o  o  o
## 7  x x x x o o o o o o o  o  o  o  o  o
## 8  x x x x o o x o o o o  o  x  o  o  o
## 9  x x x x o x x o o o o  o  o  o  o  o
## 10 x o x x o o x o o o o  o  o  o  o  o
```

L'eacf est toujours difficilement interprétable et ne permet pas de dégager de modèle candidat. Le modèle ARIMA(3,1,3) suggéré par la procédure automatique `auto.arima()` n'est pas valide non plus.

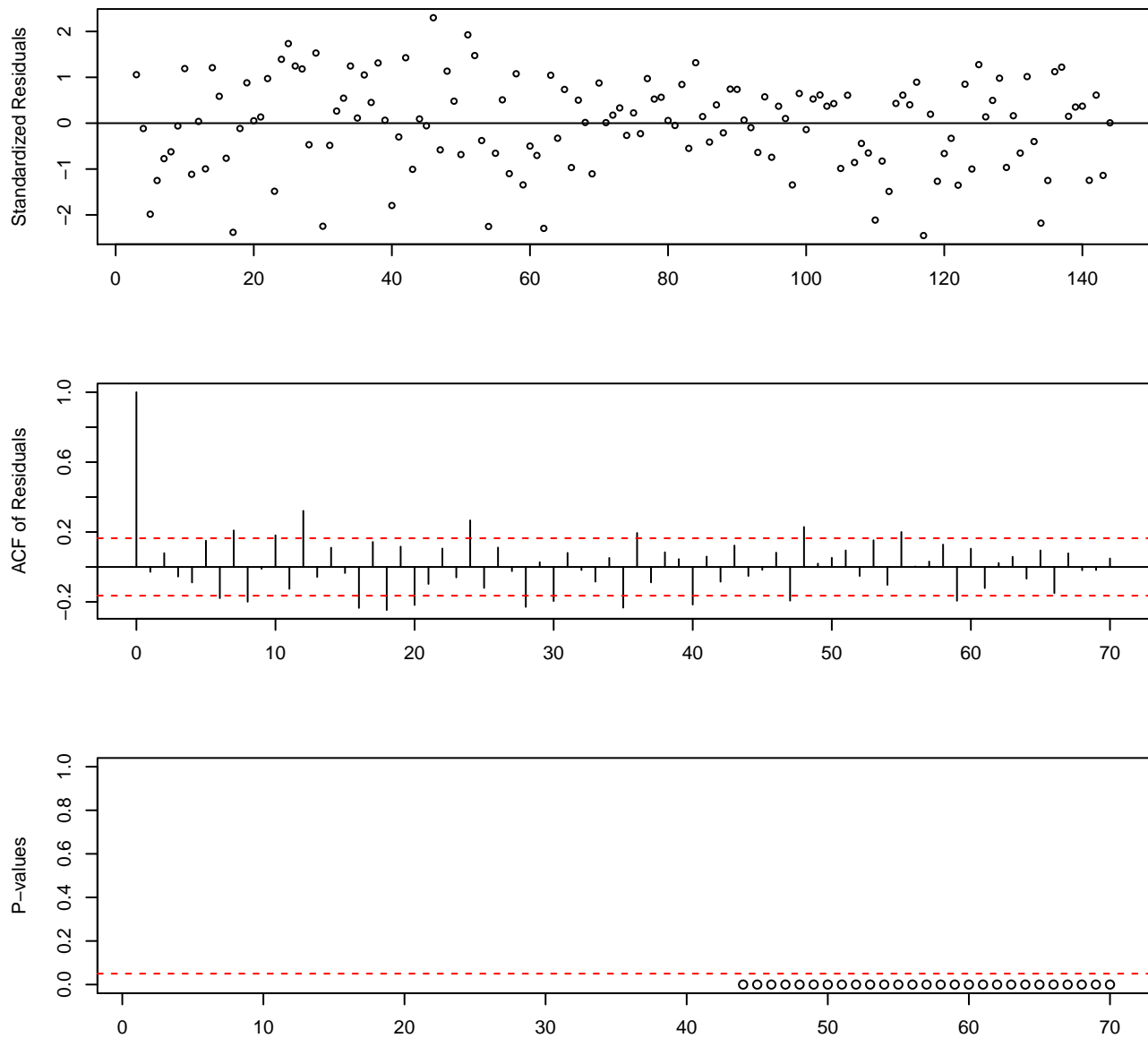
```

modRH_auto <- auto.arima(lmRH$res)
modRH_auto

## Series: lmRH$res
## ARIMA(3,1,3)
##
## Coefficients:
##          ar1      ar2      ar3      ma1      ma2      ma3
##      -0.3467 -0.3674  0.5676 -0.2298  0.3173 -0.7825
## s.e.   0.1393   0.1353  0.1296   0.1174  0.0883   0.0806
##
## sigma^2 estimated as 0.001778:  log likelihood=251.14
## AIC=-488.28   AICc=-487.45   BIC=-467.54

tsdiag.Arima(modRH_auto, gof.lag=70)

```



On fait appel à la procédure MINIC.

```
library(quantmod) # fonction Lag() utilisée dans armaselect()

## Loading required package: xts
## Loading required package: TTR
##
## Attaching package: 'TTR'
## The following object is masked from 'package:fBasics':
##
##   volatility
## Version 0.4-0 included new data defaults. See ?getSymbols.

armaselect(lmRH$res, nbmod=5) # package caschrono

##      p q      sbc
```

```
## [1,] 14 2 -910.0730
## [2,] 14 0 -908.6276
## [3,] 14 1 -906.7453
## [4,] 14 3 -906.4536
## [5,] 14 4 -906.2166
```

```
modRH <- Arima(lmRH$res, order=c(14,0,2), include.mean=F)
modRH
```

```
## Series: lmRH$res
## ARIMA(14,0,2) with zero mean
##
## Coefficients:
##      ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8      ar9
##      0.2299  0.921  -0.2264 -0.3674  0.2512  0.2552  -0.2256  -0.2795  0.3859
## s.e.  0.1761  0.131   0.1557  0.1126  0.1173  0.1049   0.1180   0.0977  0.1184
##      ar10     ar11     ar12     ar13     ar14     ma1      ma2
##      0.1553  -0.4228  0.4209  0.1993  -0.4691  0.3074  -0.4261
## s.e.  0.1121   0.1250  0.1106  0.0878   0.0813  0.1975   0.1782
##
## sigma^2 estimated as 0.001229:  log likelihood=282.58
## AIC=-531.15   AICc=-526.29   BIC=-480.66
```

```
coeftest(modRH)
```

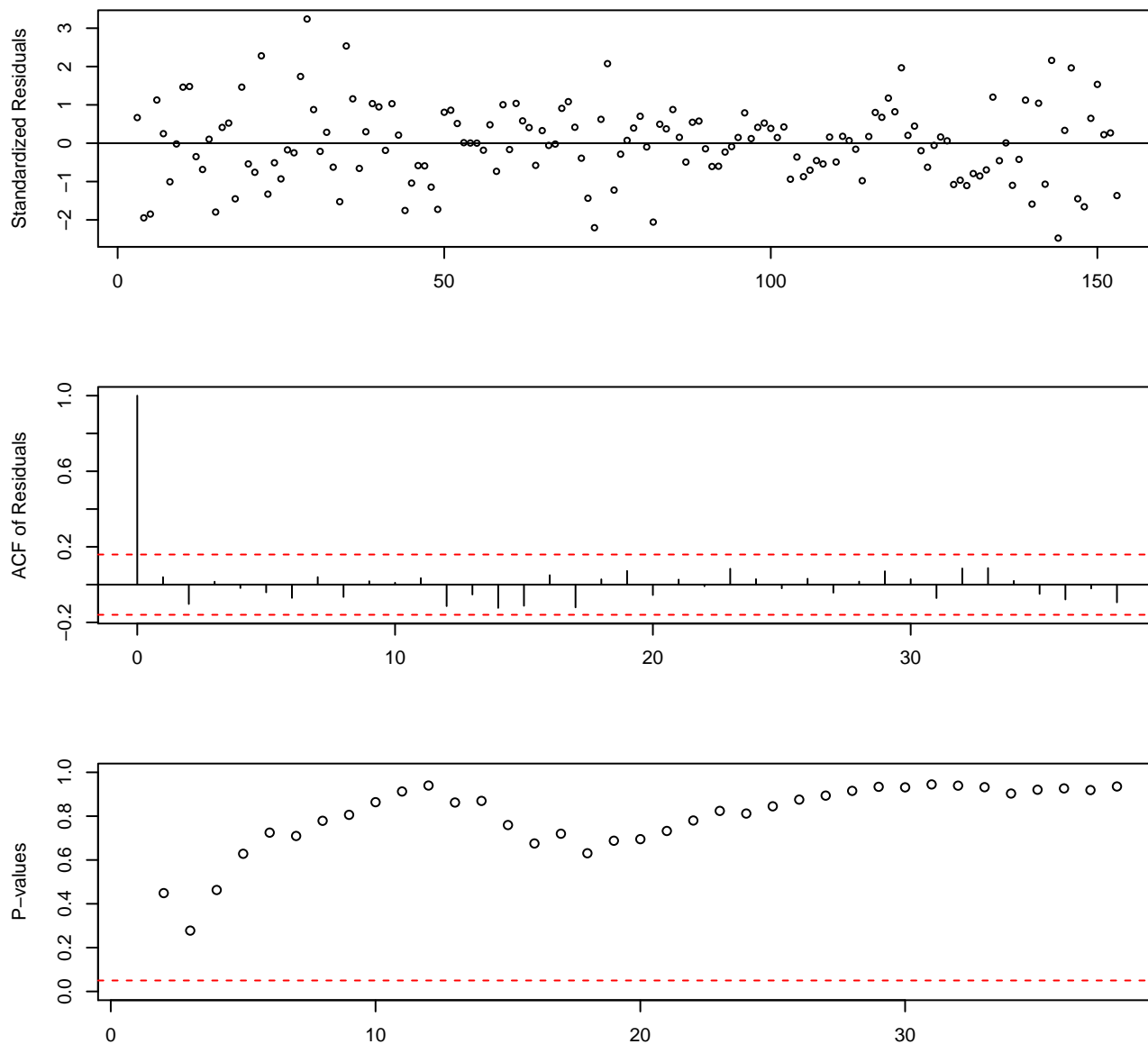
```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1   0.229902   0.176141   1.3052 0.1918200
## ar2   0.921004   0.130959   7.0328 2.025e-12 ***
## ar3  -0.226420   0.155723  -1.4540 0.1459488
## ar4  -0.367447   0.112563  -3.2644 0.0010971 **
## ar5   0.251188   0.117309   2.1413 0.0322536 *
## ar6   0.255210   0.104853   2.4340 0.0149341 *
## ar7  -0.225560   0.118000  -1.9115 0.0559380 .
## ar8  -0.279454   0.097707  -2.8601 0.0042350 **
## ar9   0.385943   0.118390   3.2599 0.0011144 **
## ar10  0.155270   0.112096   1.3851 0.1660079
## ar11 -0.422820   0.125014  -3.3822 0.0007191 ***
## ar12  0.420888   0.110579   3.8062 0.0001411 ***
## ar13  0.199289   0.087773   2.2705 0.0231765 *
## ar14 -0.469057   0.081278  -5.7710 7.880e-09 ***
## ma1   0.307361   0.197542   1.5559 0.1197256
## ma2  -0.426070   0.178195  -2.3910 0.0168009 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(modRH)[1,1]
```

```
## [1] 0.2299023
```

Pour valider le modèle ARMA(14,2) suggéré par la procédure MINIC, il faut calculer les tests de Ljung-Box à un ordre relativement important, car le modèle ARMA(p,q) proposé comporte beaucoup de paramètres.

```
tsdiag.Arima(modRH, gof.lag=70)
```



Le modèle ARMA(14,2) est valide. Voyons s'il est simplifiable.

```
t_stat(modRH)
```

```
##          ar1      ar2      ar3      ar4      ar5      ar6      ar7
## t.stat 1.305214 7.032779 -1.453991 -3.264376 2.141253 2.433972 -1.911518
## p.val  0.191820 0.000000 0.145949 0.001097 0.032254 0.014934 0.055938
##          ar8      ar9      ar10     ar11     ar12     ar13     ar14
## t.stat -2.860108 3.259917 1.385146 -3.382187 3.806221 2.270512 -5.771012
## p.val  0.004235 0.001114 0.166008 0.000719 0.000141 0.023177 0.000000
##          ma1      ma2
## t.stat 1.555926 -2.391035
## p.val  0.119726 0.016801
```

Le modèle est non simplifiable. Vérifions que les résidus du modèle sont gaussiens.

```
shapiro.test(modRH$res)

##
##  Shapiro-Wilk normality test
##
## data:  modRH$res
## W = 0.98318, p-value = 0.07493

shapiro.test(modRH$res)$p.value

## [1] 0.07492575
```

```
dagoTest(modRH$res)

##
## Title:
## D'Agostino Normality Test
##
## Test Results:
## STATISTIC:
## Chi2 | Omnibus: 2.5268
## Z3 | Skewness: -1.4762
## Z4 | Kurtosis: 0.5895
## P VALUE:
## Omnibus Test: 0.2827
## Skewness Test: 0.1399
## Kurtosis Test: 0.5555
##
## Description:
## Sun May 8 12:33:08 2022 by user:

dagoTest(modRH$res)@test

## $statistic
## Chi2 | Omnibus Z3 | Skewness Z4 | Kurtosis
## 2.5267956 -1.4762458 0.5894863
##
## $method
## [1] "D'Agostino Omnibus Normality Test"
##
## $p.value
## Omnibus Test Skewness Test Kurtosis Test
## 0.2826919 0.1398779 0.5555351
##
## $data.name
## [1] "modRH$res"

dagoTest(modRH$res)@test$p.value[1]

## Omnibus Test
## 0.2826919
```

#### 16.1.1.3.4.3.2 Modèle complet SARIMA(,1,)(14,0,2)[12] avec fonction Arima()

```

r <- 12
t <- temps
Cos1 <- cos(2*pi*(t-1)/r)
Sin1 <- sin(2*pi*(t-1)/r)
Cos2 <- cos(4*pi*(t-1)/r)
Sin2 <- sin(4*pi*(t-1)/r)
Cos3 <- cos(6*pi*(t-1)/r)
Sin3 <- sin(6*pi*(t-1)/r)
MatRH3 <- cbind(Cos1=Cos1, Sin1=Sin1, Cos2=Cos2, Sin2=Sin2, Cos3=Cos3, Sin3=Sin3)
Mat1b <- cbind(temps, MatRH3)
modeleM1b <- Arima(AirPassengers, xreg=Mat1b, order=c(14,0,2), lambda=0)
modeleM1b

```

```

## Series: AirPassengers
## Regression with ARIMA(14,0,2) errors
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##      0.2332  0.9227 -0.2297 -0.3692  0.2530  0.2576 -0.2260 -0.2804
## s.e.  0.1783  0.1317  0.1578  0.1130  0.1186  0.1055  0.1191  0.0987
##          ar9      ar10      ar11      ar12      ar13      ar14      ma1      ma2
##      0.3866  0.1560 -0.4235  0.4213  0.1982 -0.4707  0.3028 -0.4302
## s.e.  0.1185  0.1143  0.1260  0.1129  0.0889  0.0822  0.1995  0.1800
##      intercept  temps      Cos1      Sin1      Cos2      Sin2      Cos3      Sin3
##      4.8222  0.0099 -0.1488  0.0292  0.0571  0.0594 -0.0084 -0.0276
## s.e.  0.0282  0.0003  0.0119  0.0127  0.0065  0.0066  0.0020  0.0020
##
## sigma^2 estimated as 0.001309:  log likelihood=282.67
## AIC=-515.34  AICc=-504.32  BIC=-441.09

```

```

modeleM1b$coef
##          ar1          ar2          ar3          ar4          ar5          ar6
## 0.233217133 0.922739566 -0.229710061 -0.369243247 0.252979009 0.257554297
##          ar7          ar8          ar9          ar10          ar11          ar12
## -0.225954483 -0.280415103 0.386639812 0.155964666 -0.423534466 0.421253837
##          ar13          ar14          ma1          ma2          intercept          temps
## 0.198163685 -0.470698189 0.302793912 -0.430189335 4.822178414 0.009940844
##          Cos1          Sin1          Cos2          Sin2          Cos3          Sin3
## -0.148770854 0.029198210 0.057060119 0.059409531 -0.008429316 -0.027627495

```

```

t_stat(modeleM1b)
##          ar1          ar2          ar3          ar4          ar5          ar6          ar7
## t.stat 1.307821 7.006982 -1.455950 -3.266614 2.133212 2.442204 -1.896837
## p.val  0.190934 0.000000 0.145406 0.001088 0.032907 0.014598 0.057849
##          ar8          ar9          ar10          ar11          ar12          ar13          ar14
## t.stat -2.840818 3.263504 1.364237 -3.360725 3.730198 2.228468 -5.729634
## p.val  0.004500 0.001100 0.172493 0.000777 0.000191 0.025849 0.000000
##          ma1          ma2 intercept          temps          Cos1          Sin1          Cos2
## t.stat 1.517396 -2.390154 171.1821 28.83834 -12.46701 2.297088 8.720365
## p.val  0.129167 0.016841 0.0000 0.00000 0.00000 0.021614 0.000000
##          Sin2          Cos3          Sin3
## t.stat 8.973876 -4.167036 -13.61406
## p.val  0.000000 0.000031 0.00000

```

Le modèle est non simplifiable.  
 Explicitons notre modèle final.

```
coeftest(modeleM1b)

##
## z test of coefficients:
##
##           Estimate Std. Error z value Pr(>|z|)
## ar1         0.23321713 0.17832491  1.3078 0.1909339
## ar2         0.92273957 0.13168859  7.0070 2.435e-12 ***
## ar3        -0.22971006 0.15777328 -1.4560 0.1454063
## ar4        -0.36924325 0.11303548 -3.2666 0.0010884 **
## ar5         0.25297901 0.11859064  2.1332 0.0329073 *
## ar6         0.25755430 0.10545978  2.4422 0.0145979 *
## ar7        -0.22595448 0.11912173 -1.8968 0.0578495 .
## ar8        -0.28041510 0.09870930 -2.8408 0.0044998 **
## ar9         0.38663981 0.11847383  3.2635 0.0011004 **
## ar10        0.15596467 0.11432377  1.3642 0.1724931
## ar11       -0.42353447 0.12602471 -3.3607 0.0007774 ***
## ar12        0.42125384 0.11293069  3.7302 0.0001913 ***
## ar13        0.19816369 0.08892373  2.2285 0.0258493 *
## ar14       -0.47069819 0.08215152 -5.7296 1.006e-08 ***
## ma1         0.30279391 0.19954837  1.5174 0.1291667
## ma2        -0.43018933 0.17998390 -2.3902 0.0168413 *
## intercept   4.82217841 0.02816988 171.1821 < 2.2e-16 ***
## temps       0.00994084 0.00034471  28.8383 < 2.2e-16 ***
## Cos1       -0.14877085 0.01193316 -12.4670 < 2.2e-16 ***
## Sin1        0.02919821 0.01271096  2.2971 0.0216137 *
## Cos2        0.05706012 0.00654332  8.7204 < 2.2e-16 ***
## Sin2        0.05940953 0.00662028  8.9739 < 2.2e-16 ***
## Cos3       -0.00842932 0.00202286 -4.1670 3.086e-05 ***
## Sin3       -0.02762749 0.00202934 -13.6141 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

names(modeleM1b)

## [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
## [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
## [13] "nobs"      "model"     "aicc"      "bic"       "xreg"      "lambda"
## [19] "x"         "fitted"
```

$$\forall t \in T \quad \log(X_t) = m(t) + s(t) + B_t$$

avec

- $m(t) = 4.8137 + 0.01t$
  - $s(t) = \sum_{k=1}^3 c_k \cos\left(\frac{2k\pi(t-1)}{12}\right) + d_k \sin\left(\frac{2k\pi(t-1)}{12}\right)$
- où

$$\forall k \in \{1, 2, 3\} \quad c_k = \text{coeftest}(\text{modeleM1b})[18+2i-1, 1] \quad \text{et} \quad d_k = \text{coeftest}(\text{modeleM1b})[19+2i-1, 1]$$

$$\begin{aligned} c_1 &= -0.1488 & c_2 &= 0.0571 & c_3 &= -0.0084 \\ d_1 &= 0.0292 & d_2 &= 0.0594 & d_3 &= -0.0276 \end{aligned}$$

- $(B_t)_{t \in T}$  un ARMA(14,2) défini par

$$B_t = \sum_{k=1}^{14} \varphi_k B_{t-k} + \sum_{k=0}^2 \theta_k \varepsilon_{t-k}$$

où

$$\begin{array}{llll} \varphi_1 = 0.2332 & \varphi_2 = 0.9227 & \varphi_3 = -0.2297 & \varphi_4 = -0.3692 \\ \varphi_5 = 0.253 & \varphi_6 = 0.2576 & \varphi_7 = -0.226 & \varphi_8 = -0.2804 \\ \varphi_9 = 0.3866 & \varphi_{10} = 0.156 & \varphi_{11} = -0.4235 & \varphi_{12} = 0.4213 \\ \varphi_{13} = 0.1982 & \varphi_{14} = -0.4707 & \theta_1 = 0.3028 & \theta_2 = -0.4302 \end{array}$$

et

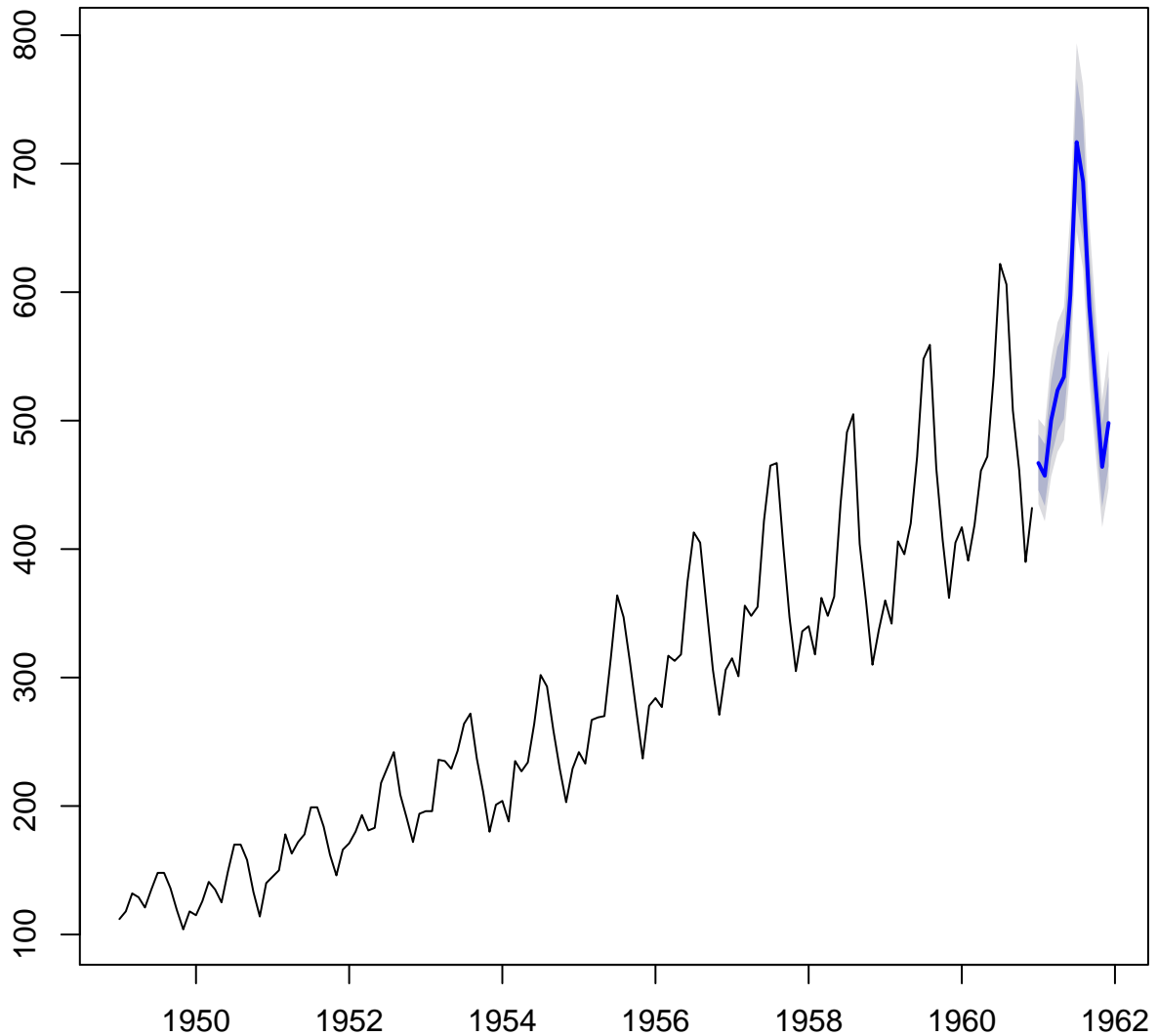
$$\varepsilon_t \sim \mathcal{N}(0, 0.0013)$$

```
tempsP <- (length(log(AirPassengers))+1):(length(log(AirPassengers))+12)
t <- tempsP
Cos1 <- cos(2*pi*(t-1)/r)
Sin1 <- sin(2*pi*(t-1)/r)
Cos2 <- cos(4*pi*(t-1)/r)
Sin2 <- sin(4*pi*(t-1)/r)
Cos3 <- cos(6*pi*(t-1)/r)
Sin3 <- sin(6*pi*(t-1)/r)
MatRH3P <- cbind(Cos1=Cos1, Sin1=Sin1, Cos2=Cos2, Sin2=Sin2, Cos3=Cos3, Sin3=Sin3)
Mat1bP <- cbind(tempsP, MatRH3P)
prevM1b <- forecast(modeleM1b, xreg=Mat1bP, h=12)

## Warning in forecast.forecast_ARIMA(modeleM1b, xreg = Mat1bP, h = 12): xreg contains different
## column names from the xreg used in training. Please check that the regressors are in the same order.

plot(prevM1b)
```

### Forecasts from Regression with ARIMA(14,0,2) errors



#### 16.1.1.3.5 Annexe : Modélisation par saisonnalité stochastique

Les tests de Canova-Hansen (CH) et de (OCSB) indiquent qu'une modélisation des résidus par une tendance saisonnière stochastique n'est pas adaptée. Nous allons néanmoins tenter de construire ce modèle pour le comparer à tous les autres.

Supposons donc qu'il existe  $D \in \mathbb{N}^*$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \Delta_{12}^D(\log(X_t) - 0.01t) = B_t$$

Il faudrait déterminer l'ordre  $D$  de la différenciation saisonnière. Cette méthode est forcément inapplicable ici.

```
nsdiffs(residus, test="ocsb", m=12, max.D=3)
```

```
## Warning: argument m is deprecated; please set the frequency in the ts object.
```

```
## [1] 0
```

Considérons arbitrairement que  $D = 1$ . Explicitons notre modèle. Il existe un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \Delta_{12}^1(\log(X_t) - 0.01) = B_t$$

Tentons de modéliser la série différenciée  $(B_t)_{t \in T}$  avec un ARMA(p,q).

```
eacf(diff(residus, lag=12))

## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x x x x x x x x o o x o o
## 1 x o x o o o o o o o x o o
## 2 x o x o o o o o o o x o o
## 3 x x x o o o o o o o x o o
## 4 x x o o o o o o o o x o o
## 5 x x o o o o o o o o x o o
## 6 x x o o o o o o o o x o o
## 7 o x o o o o o o o o x o o
```

L'eacf pourrait suggérer les modèles ARMA(1,3), ARMA(2,3), ARMA(3,3) ou encore un ARMA(4,2). Mais ces modèles ne sont pas valides. Utilisons la procédure MINIC pour obtenir d'autres modèles candidats.

```
armaselect(diff(residus, lag=12), nbmod=3)

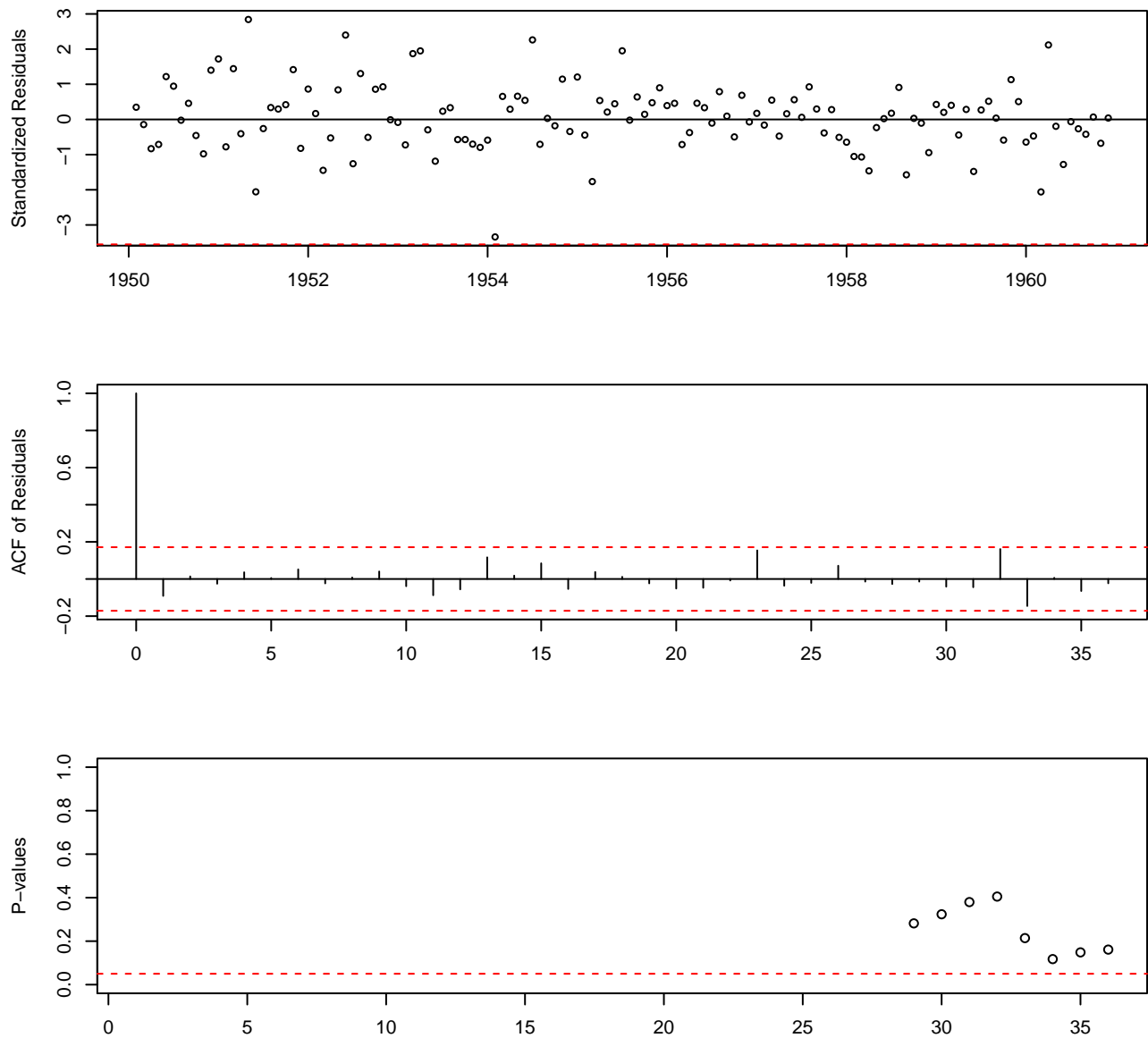
##      p  q      sbc
## [1,] 1  2 -843.8906
## [2,] 1  1 -841.4932
## [3,] 1 12 -840.9538
```

Le premier modèle à être valide est le modèle ARMA(1,12). On considère donc pour la série initiale le modèle SARIMA(1,0,12)(0,1,0)[12] avec tendance linéaire.

```
modeleM2 <- Arima(AirPassengers, include.drift=T, order=c(1,0,12),
seasonal=list(order=c(0,1,0), period =12), lambda=0)
modeleM2

## Series: AirPassengers
## ARIMA(1,0,12)(0,1,0)[12] with drift
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1          ma1          ma2          ma3          ma4          ma5          ma6          ma7
##      0.9337 -0.2816  0.0334 -0.1814 -0.2164  0.1583 -0.0199  0.0151
## s.e.  0.0394  0.1046  0.1268  0.1101  0.1219  0.1163  0.1318  0.1313
##          ma8          ma9          ma10          ma11          ma12          drift
##     -0.0722  0.0902 -0.0041  0.1322 -0.6537  0.0099
## s.e.   0.1394  0.1389  0.1052  0.1283  0.1192  0.0005
##
## sigma^2 estimated as 0.001267:  log likelihood=254.95
## AIC=-479.91  AICc=-475.77  BIC=-436.66
```

```
tsdiag.Arima(modeleM2, gof.lag=round(length(log(AirPassengers))/4))
```



Le modèle est valide.

```
t_stat(modeleM2)

##          ar1          ma1          ma2          ma3          ma4          ma5          ma6
## t.stat 23.70481 -2.691309 0.263529 -1.648364 -1.775001 1.361459 -0.150971
## p.val  0.00000 0.007117 0.792143 0.099278 0.075898 0.173369 0.879999
##          ma7          ma8          ma9          ma10          ma11          ma12          drift
## t.stat 0.115121 -0.517803 0.649649 -0.038663 1.030421 -5.482114 20.79467
## p.val  0.908349 0.604595 0.515919 0.969159 0.302812 0.000000 0.00000
```

Le modèle n'est pas simplifiable.

#### 16.1.1.3.5.1 Explicitation du modèle SARIMA(1,0,12)(0,1,0)[12]

Explicitons notre modèle.

```
coeftest(modeleM2) # class coeftest

##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1    0.93367113 0.03938741 23.7048 < 2.2e-16 ***
## ma1   -0.28155741 0.10461726 -2.6913 0.007117 **
## ma2    0.03342529 0.12683711 0.2635 0.792143 .
## ma3   -0.18143174 0.11006775 -1.6484 0.099278 .
## ma4   -0.21640523 0.12191834 -1.7750 0.075898 .
## ma5    0.15831012 0.11627972 1.3615 0.173369
## ma6   -0.01990519 0.13184814 -0.1510 0.879999
## ma7    0.01511713 0.13131551 0.1151 0.908349
## ma8   -0.07217912 0.13939483 -0.5178 0.604595
## ma9    0.09022037 0.13887551 0.6496 0.515919
## ma10  -0.00406585 0.10516054 -0.0387 0.969159
## ma11   0.13224817 0.12834382 1.0304 0.302812
## ma12  -0.65370393 0.11924305 -5.4821 4.203e-08 ***
## drift  0.00993715 0.00047787 20.7947 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

$$\forall t \in T \quad \Delta_{12}^1(\log(X_t) - m_t) = B_t$$

avec

- $m(t) = 0.01t$
- $(B_t)_{t \in T}$  un ARMA(1,12) donné par

$$\begin{aligned} B_t &= \sum_{k=1}^1 \varphi_k B_{t-k} + \sum_{q=0}^{12} \theta_q \varepsilon_{t-q} \\ &= \varphi_1 B_{t-1} + \varepsilon_t + \sum_{q=1}^{12} \text{coeftest(modeleM2)[k+1,1]} \varepsilon_{t-q} \end{aligned}$$

où

$$\begin{aligned} \theta_1 &= -0.2816 & \theta_2 &= 0.0334 & \theta_3 &= -0.1814 & \theta_4 &= -0.2164 \\ \theta_5 &= 0.1583 & \theta_6 &= -0.0199 & \theta_7 &= 0.0151 & \theta_8 &= -0.0722 \\ \theta_9 &= 0.0902 & \theta_{10} &= -0.0041 & \theta_{11} &= 0.1322 & \theta_{12} &= -0.6537 \\ \varphi_1 &= 0.9337 \end{aligned}$$

et  $(\varepsilon_t)_{t \in T}$  est un bruit blanc de variance 0.0013.

#### 16.1.1.3.5.2 Analyse des résidus

Testons la normalité des résidus du modèle.

```
shapiro.test(modeleM2$res)

##
## Shapiro-Wilk normality test
##
## data:  modeleM2$res
## W = 0.97452, p-value = 0.008662

dagoTest(modeleM2$res)$test$p.value[1]

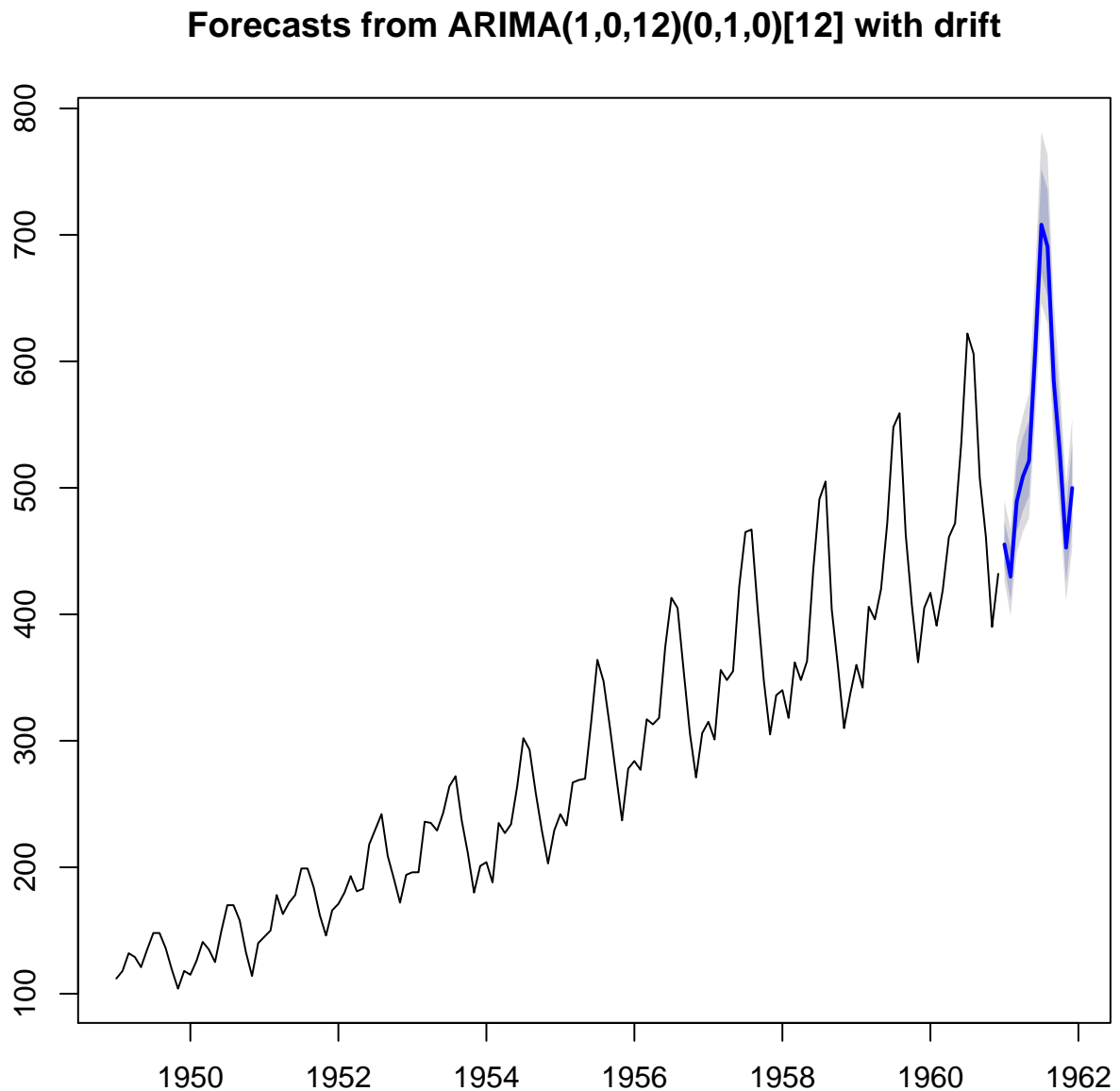
## Omnibus Test
## 0.02447534
```

Les résidus du modèle ne sont pas gaussiens.

#### 16.1.1.3.5.3 Prévisions

Le processus  $(\varepsilon_t)_{t \in T}$  est un bruit blanc non gaussien donc on calcule les prévisions en utilisant le bootstrap.

```
prevM2 <- forecast(modeleM2, h=12, bootstrap=T)
plot(prevM2)
```



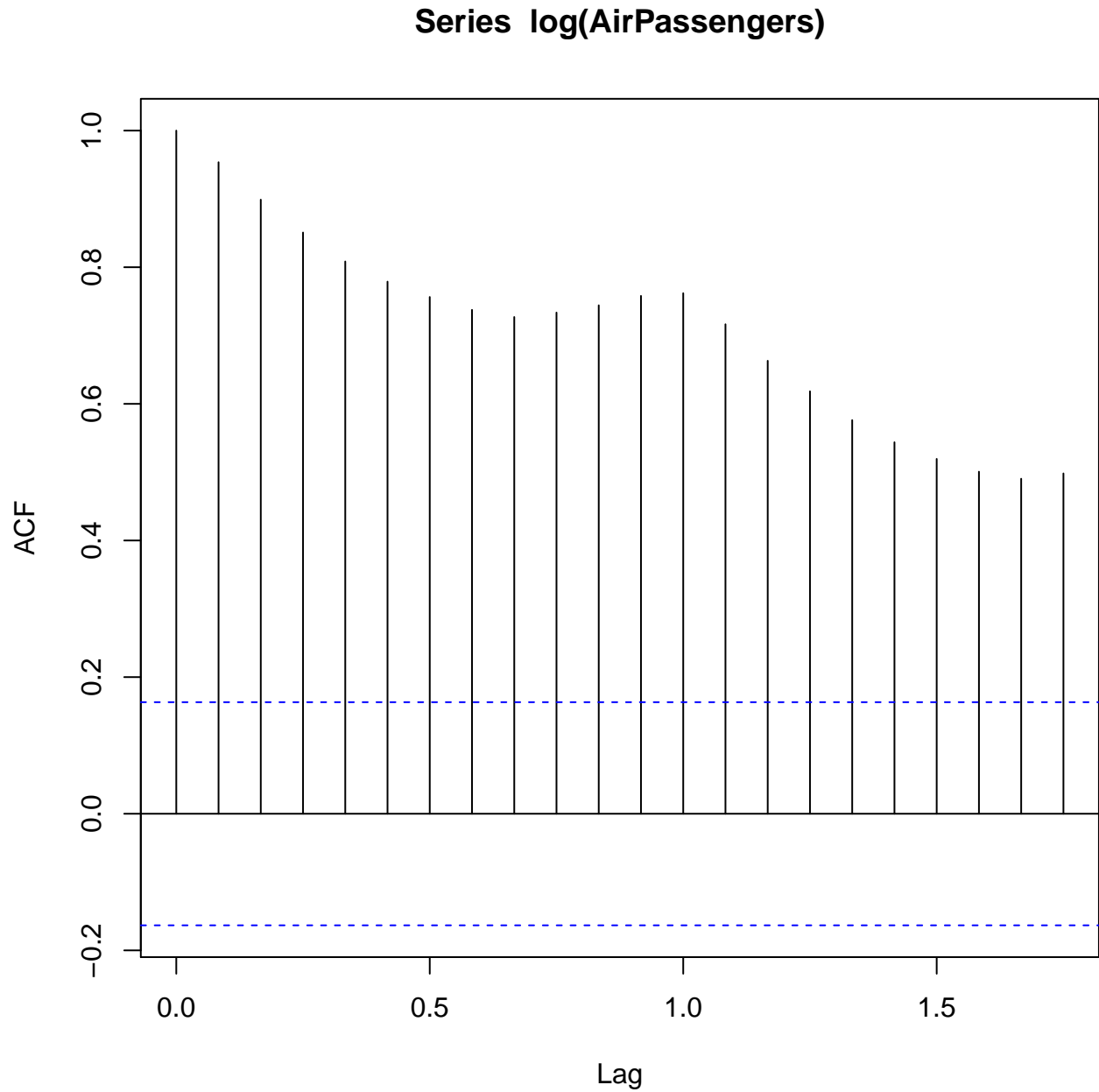
### 16.1.2 Modélisation par transformation de Box-Cox (saisonnalité puis tendance)

Nous ne répétons pas la transformation de Box-Cox.

### 16.1.2.1 Détection d'une saisonnalité (et d'une tendance)

La fonction d'autocorrélation empirique justifie la présence d'une composante saisonnière (et d'une composante tendancielle).

```
acf(log(AirPassengers))
```



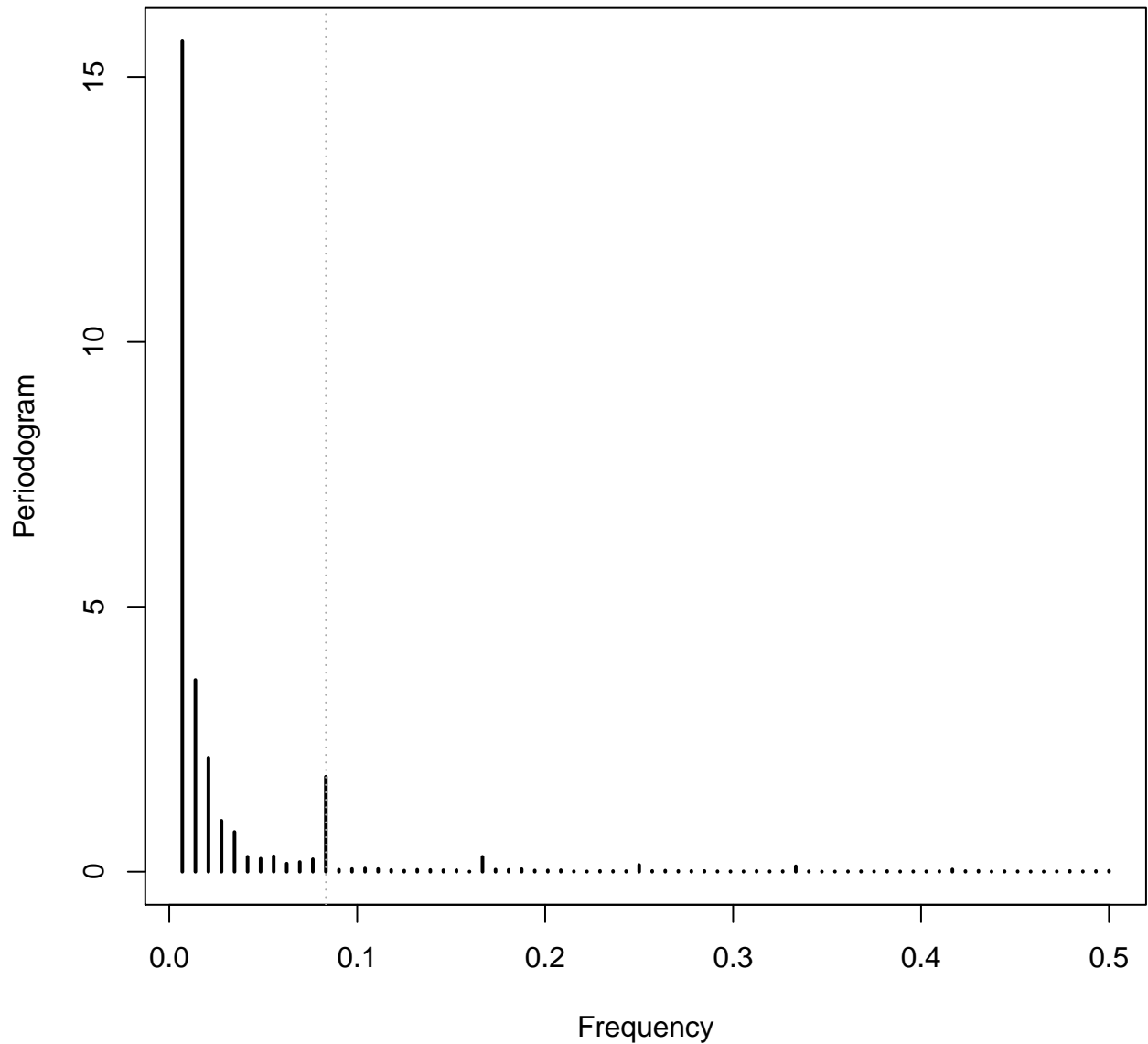
Nous supposons qu'il existe deux fonctions  $(m, s) \in \mathbb{R}^T \times \mathbb{R}^T$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \log(X_t) = m(t) + s(t) + B_t$$

### 16.1.2.2 Étude de la composante saisonnière

#### 16.1.2.2.1 Identification de la saisonnalité

```
periodogram(log(AirPassengers))
abline(v=1/12, lty=3, col="darkgrey")
```



Le périodogramme détecte la période  $r = 12$ , mais le périodogramme privilégierait plutôt une saisonnalité beaucoup plus grande, ce qui remettrait en cause la légitimité du modèle.

#### 16.1.2.2.2 Validation de la saisonnalité

Néanmoins, la saisonnalité  $r = 12$  est validée avec le test de saisonnalité de Friedman.

```
Saison.test(log(AirPassengers), 12)

## Période      d.f.      Tobs p-valeur
## 12.0000    11.0000  121.0803   0.0000
```

### 16.1.2.2.3 Identification de la nature de la saisonnalité

Appliquons les tests de racine unitaire saisonnière pour identifier la nature de la composante saisonnière.

```
# nsdiffs(log(AirPassengers), test="ch", m=12)
nsdiffs(log(AirPassengers), test="ocsb", m=12)

## Warning: argument m is deprecated; please set the frequency in the ts object.

## [1] 0
```

D'après les tests de Canova-Hansen (CH) et de (OCSB), il est préférable de modéliser la série  $\log(\text{AirPassengers})$  avec une composante saisonnière déterministe. La fonction  $s$  est donc déterministe. Explicitons notre modèle.

Il existe  $m \in \mathbb{R}^T$ ,  $(s_1, \dots, s_{12}) \in \mathbb{R}^{12}$  et  $(B_t)_{t \in T}$  un processus stationnaire centré tel que

$$\forall t \in T \quad \log(X_t) = m(t) + \sum_{i=1}^{12} s_j \chi_{i+12\mathbb{Z}} + B_t$$

### 16.1.2.2.4 Modélisation de la saisonnalité déterministe

#### 16.1.2.2.4.1 Par moyennes saisonnières

```
mois. <- season(log(AirPassengers))
lmMS <- lm(log(AirPassengers)~mois. -1)
lmMS

##
## Call:
## lm(formula = log(AirPassengers) ~ mois. - 1)
##
## Coefficients:
##   mois.January   mois.February   mois.March   mois.April   mois.May
##         5.401         5.389         5.530         5.509         5.516
##   mois.June   mois.July   mois.August   mois.September   mois.October
##         5.648         5.762         5.763         5.629         5.501
##   mois.November   mois.December
##         5.367         5.491
```

On accède directement aux coefficients avec les commandes suivantes.

```
names(lmMS)

## [1] "coefficients" "residuals" "effects" "rank"
## [5] "fitted.values" "assign" "qr" "df.residual"
## [9] "contrasts" "xlevels" "call" "terms"
## [13] "model"

lmMS$coefficients # ou plus simplement lmMS$coef

##   mois.January   mois.February   mois.March   mois.April   mois.May
##         5.401390         5.389404         5.529700         5.508500         5.516196
##   mois.June   mois.July   mois.August   mois.September   mois.October
##         5.648411         5.762422         5.763196         5.628631         5.500541
##   mois.November   mois.December
##         5.366892         5.490826
```

#### 16.1.2.2.4.1.1 Analyse des résidus

Assurons nous que les conditions du test de régression sont vérifiées.

```
bgtest(lmMS)

##
## Breusch-Godfrey test for serial correlation of order up to 1
##
## data: lmMS
## LM test = 139.62, df = 1, p-value < 2.2e-16
```

On a  $p\text{-value} < 0,05$  donc les résidus  $(m(t) + B_t)_{t \in T}$  ne sont pas issus d'un bruit blanc.

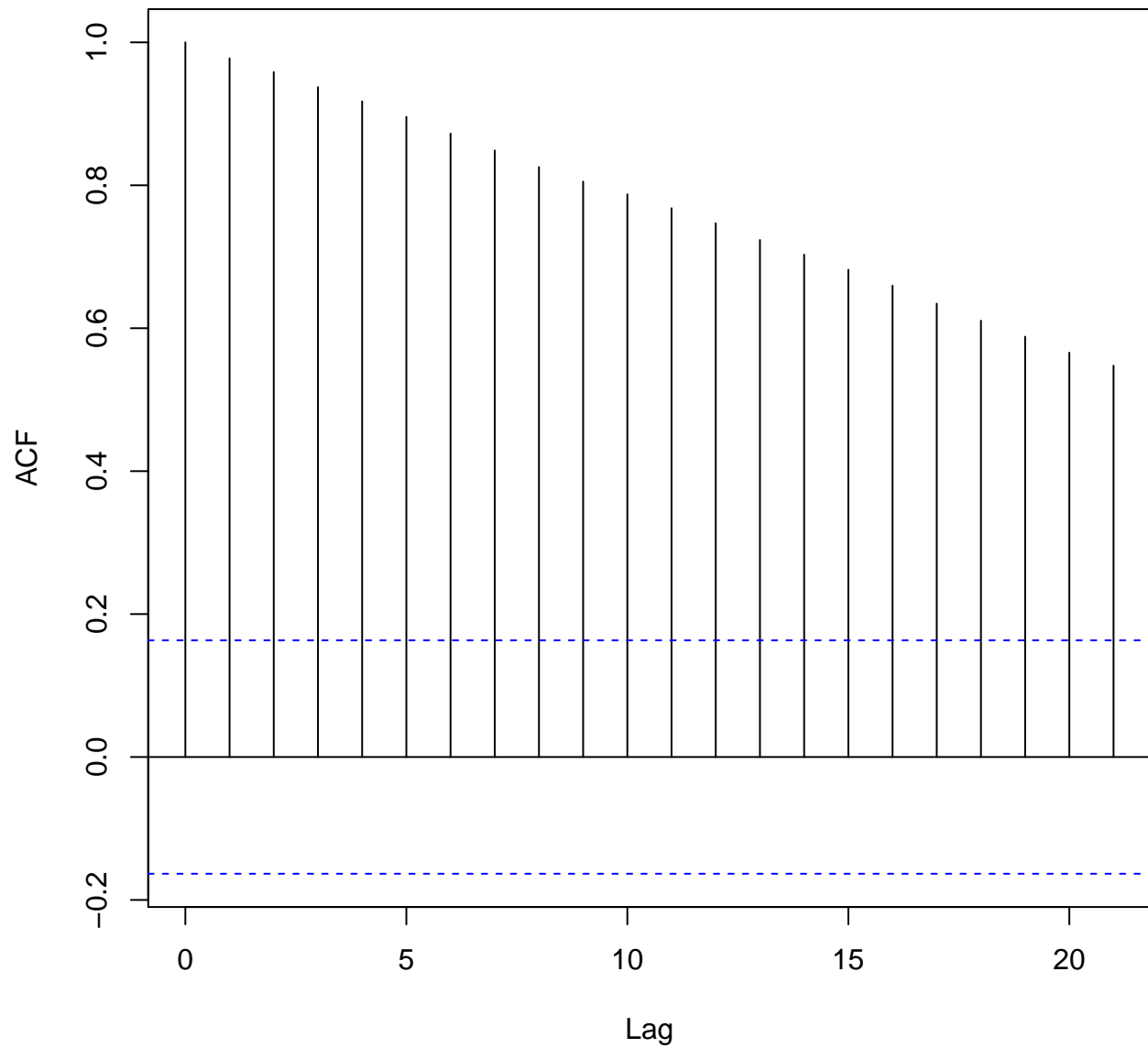
```
kpss.test(lmMS$res)

## Warning in kpss.test(lmMS$res): p-value smaller than printed p-value
##
## KPSS Test for Level Stationarity
##
## data: lmMS$res
## KPSS Level = 2.9233, Truncation lag parameter = 4, p-value = 0.01
```

Les résidus ne sont même pas stationnaires ; il reste en effet une composante de tendance, que l'on visualise sur le tracé de l'acf.

```
acf(lmMS$res)
```

### Series lmMS\$res



#### 16.1.2.2.4.1.2 Étude de la composante tendancielle

##### 16.1.2.2.4.1.2.1 Identification de la nature de la tendance

```
adf.test(lmMS$res)
```

```
##
```

```
## Augmented Dickey-Fuller Test
```

```
##
```

```
## data: lmMS$res
```

```
## Dickey-Fuller = -2.8924, Lag order = 5, p-value = 0.2049
```

```
## alternative hypothesis: stationary
```

```
adf.test(diff(lmMS$res))
```

```
## Warning in adf.test(diff(lmMS$res)): p-value smaller than printed p-value

##
## Augmented Dickey-Fuller Test
##
## data: diff(lmMS$res)
## Dickey-Fuller = -6.2419, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary

kpss.test(diff(lmMS$res))

## Warning in kpss.test(diff(lmMS$res)): p-value greater than printed p-value

##
## KPSS Test for Level Stationarity
##
## data: diff(lmMS$res)
## KPSS Level = 0.049521, Truncation lag parameter = 4, p-value = 0.1
```

Les réponses aux tests sont *Oui*, *Non*, *Oui* donc une modélisation par tendance stochastique d'ordre 1 est préconisée.

Explicitons notre modèle. Il existe  $(d, c, s_1, \dots, s_{12}) \in \mathbb{N}^* \times \mathbb{R}^{13}$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \Delta^d \left( \log(X_t) - c - \sum_{i=1}^{12} s_j \chi_{i+12\mathbb{Z}} \right) = B_t$$

**Remarque** Une autre méthode orienterait vers un autre modèle :

```
pp.test(lmMS$res)

## Warning in pp.test(lmMS$res): p-value smaller than printed p-value

##
## Phillips-Perron Unit Root Test
##
## data: lmMS$res
## Dickey-Fuller Z(alpha) = -35.819, Truncation lag parameter = 4, p-value
## = 0.01
## alternative hypothesis: stationary

pp.test(diff(lmMS$res))

## Warning in pp.test(diff(lmMS$res)): p-value smaller than printed p-value

##
## Phillips-Perron Unit Root Test
##
## data: diff(lmMS$res)
## Dickey-Fuller Z(alpha) = -155.24, Truncation lag parameter = 4, p-value
## = 0.01
## alternative hypothesis: stationary

kpss.test(diff(lmMS$res))

## Warning in kpss.test(diff(lmMS$res)): p-value greater than printed p-value

##
## KPSS Test for Level Stationarity
##
## data: diff(lmMS$res)
## KPSS Level = 0.049521, Truncation lag parameter = 4, p-value = 0.1
```

Les réponses sont *Non, Non, Oui* donc une modélisation par tendance linéaire est préconisée. Nous avons déjà proposé cette modélisation.

#### 16.1.2.2.4.1.2.2 Identification du degré de différenciation

Différencions la série.

```
kpss.test(diff(lmMS$res))

## Warning in kpss.test(diff(lmMS$res)): p-value greater than printed p-value

##
## KPSS Test for Level Stationarity
##
## data: diff(lmMS$res)
## KPSS Level = 0.049521, Truncation lag parameter = 4, p-value = 0.1
```

La série différenciée est modélisable par un ARMA(p,q). Explicitons notre modèle. Il existe  $(c, s_1, \dots, s_{12}) \in \mathbb{R}^{13}$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \Delta \left( \log(X_t) - c - \sum_{i=1}^{12} s_j \chi_{i+12\mathbb{Z}} \right) = B_t$$

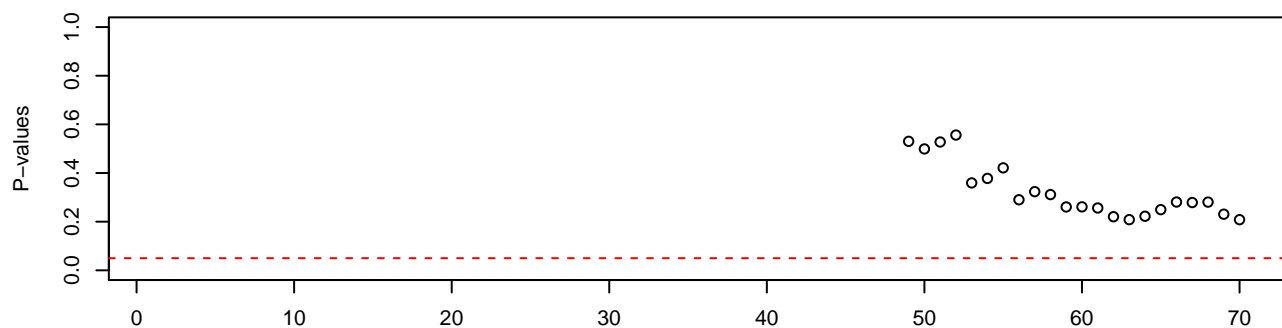
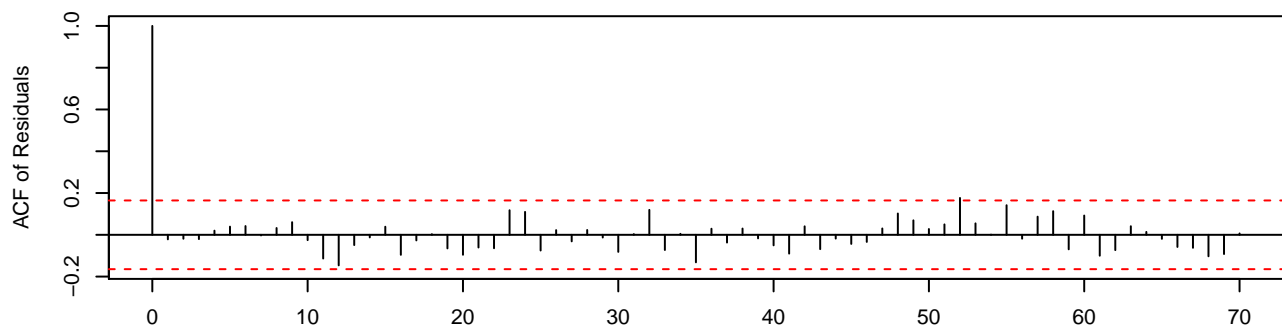
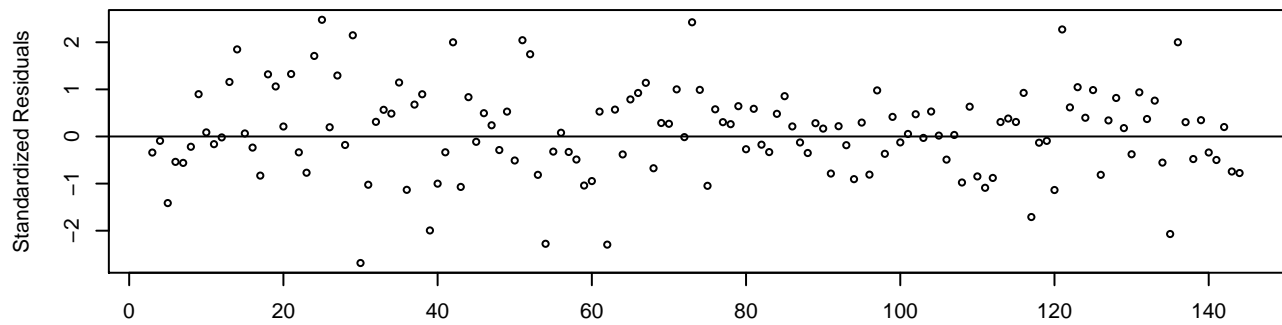
La procédure `auto.arima()` ainsi que `l'eacf()` suggèrent des modèles qui ne sont pas valides. Faisons appel à la procédure MINIC.

```
armaselect(diff(lmMS$res), nbmod=3)

##      p q      sbc
## [1,] 12 3 -887.2528
## [2,] 12 1 -883.0479
## [3,] 13 3 -882.7862
```

```
modMS <- Arima(lmMS$res, order=c(12,1,3), include.mean=F)
```

```
tsdiag.Arima(modMS, gof.lag=70)
```



Le modèle est valide.

```
t_stat(modMS)

##          ar1          ar2          ar3          ar4          ar5          ar6          ar7
## t.stat 0.570950 -0.268790 1.024704 -0.176293 0.842037 0.280819 -1.062665
## p.val  0.568034 0.788091 0.305503 0.860064 0.399767 0.778849 0.287934
##          ar8          ar9          ar10          ar11          ar12          ma1          ma2
## t.stat -0.670294 0.848442 -0.450470 1.627719 8.43835 -2.591590 0.613140
## p.val  0.502670 0.396192 0.652372 0.103584 0.00000 0.009553 0.539784
##          ma3
## t.stat -1.815301
## p.val  0.069478
```

Le modèle ARMA(12,3) est simplifiable en un ARMA(12,1).

### 16.1.2.2.4.1.2.3 Simplification du modèle

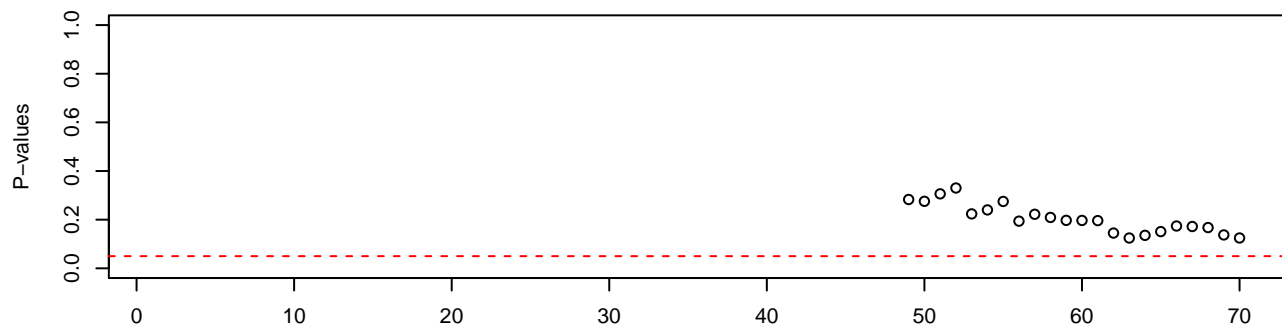
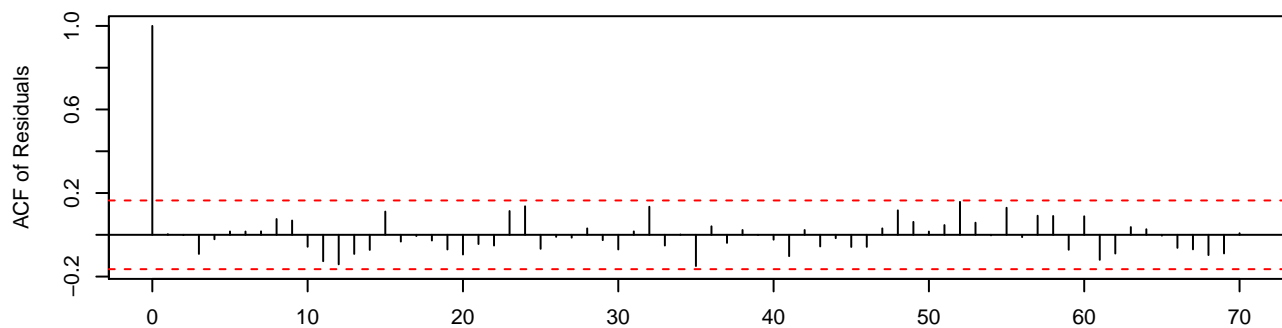
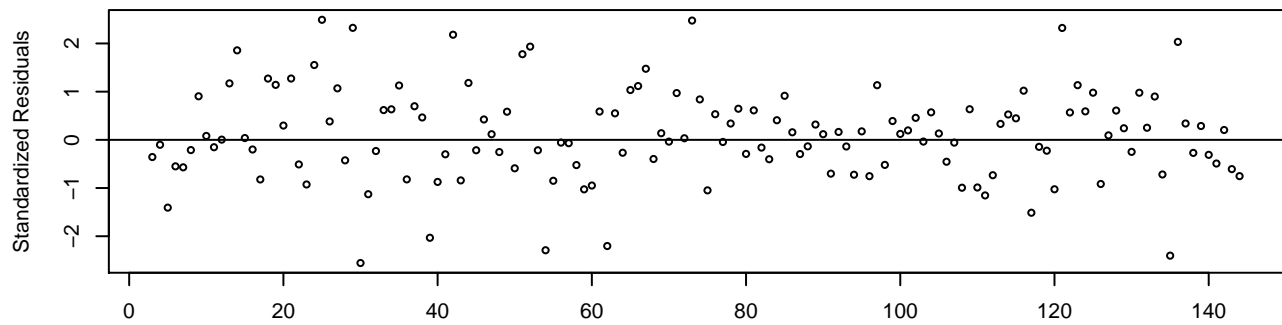
```
modMS2 <- Arima(lmMS$res, order=c(12,1,1), include.mean=F)
modMS2

## Series: lmMS$res
## ARIMA(12,1,1)
##
## Coefficients:
##      ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##      0.1275  0.0569 -0.0466 -0.0375  0.0683  0.0201 -0.0900 -0.0421
## s.e.  0.1095  0.0691  0.0677  0.0682  0.0684  0.0653  0.0678  0.0680
##      ar9      ar10     ar11     ar12     ma1
##      0.0756 -0.0371  0.0907  0.5794 -0.4573
## s.e.  0.0691  0.0695  0.0691  0.0736  0.1378
##
## sigma^2 estimated as 0.001542:  log likelihood=263.75
## AIC=-499.5   AICc=-496.22   BIC=-458.02
```

```
coeftest(modMS2)

##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1   0.127527   0.109497  1.1647 0.2441561
## ar2   0.056857   0.069071  0.8232 0.4104146
## ar3  -0.046603   0.067699 -0.6884 0.4912047
## ar4  -0.037501   0.068196 -0.5499 0.5823902
## ar5   0.068281   0.068444  0.9976 0.3184662
## ar6   0.020110   0.065278  0.3081 0.7580249
## ar7  -0.089960   0.067821 -1.3264 0.1846915
## ar8  -0.042097   0.068046 -0.6187 0.5361417
## ar9   0.075633   0.069067  1.0951 0.2734848
## ar10 -0.037096   0.069472 -0.5340 0.5933643
## ar11  0.090717   0.069083  1.3132 0.1891304
## ar12  0.579414   0.073553  7.8775 3.339e-15 ***
## ma1  -0.457319   0.137832 -3.3179 0.0009068 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
tsdiag.Arima(modMS2, gof.lag=70)
```



Le modèle simplifié est valide.

#### 16.1.2.2.4.1.2.4 Comparaison des modèles

Utilisons les critères d'information pour comparer les modèles. Commençons par vérifier que les résidus sont gaussiens.

```
shapiro.test(modMS$res)

##
##  Shapiro-Wilk normality test
##
## data:  modMS$res
## W = 0.9855, p-value = 0.1347

dagoTest(modMS$res)$test$p.value[1]
```

```
## Omnibus Test
## 0.3471136
```

```
shapiro.test(modMS2$res)

##
## Shapiro-Wilk normality test
##
## data: modMS2$res
## W = 0.98261, p-value = 0.06487

dagoTest(modMS2$res)

##
## Title:
## D'Agostino Normality Test
##
## Test Results:
## STATISTIC:
## Chi2 | Omnibus: 2.2824
## Z3 | Skewness: 0.0986
## Z4 | Kurtosis: 1.5075
## P VALUE:
## Omnibus Test: 0.3194
## Skewness Test: 0.9214
## Kurtosis Test: 0.1317
##
## Description:
## Sun May 8 12:33:20 2022 by user:
```

Réalisons un test de vraisemblance.

```
lrtest(modMS2,modMS)

## Likelihood ratio test
##
## Model 1: Arima(y = lmMS$res, order = c(12, 1, 1), include.mean = F)
## Model 2: Arima(y = lmMS$res, order = c(12, 1, 3), include.mean = F)
## #Df LogLik Df Chisq Pr(>Chisq)
## 1 14 263.75
## 2 16 265.27 2 3.0432 0.2184
```

On accepte  $H_0$  donc on conserve le modèle simplifié car il est valide.

```
#modeleM3a <- Arima(AirPassengers, xreg=MatMS, order=c(12,1,1), lambda=0, include.mean=F)
```

La procédure `Arima()` qui repose sur l'algorithme d'optimisation numérique `optim()` ne parvient pas à converger (cela peut changer avec les mises à jour régulières de **R**). Fixons les coefficients calculés précédemment.

```
modeleM3a <- Arima(AirPassengers, xreg=MatMS, order=c(12,1,1), fixed=c(modMS2$coef, lmMS$coef),
lambda=0, include.mean=F)
modeleM3a

## Series: AirPassengers
## Regression with ARIMA(12,1,1) errors
## Box Cox transformation: lambda= 0
```

```
##
## Coefficients:
##      ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
## 0.1275  0.0569 -0.0466 -0.0375  0.0683  0.0201 -0.0900 -0.0421
##      ar9      ar10     ar11     ar12     ma1      s'1      s'2      s'3
## 0.0756 -0.0371  0.0907  0.5794 -0.4573  5.4014  5.3894  5.5297
##      s'4      s'5      s'6      s'7      s'8      s'9      s'10     s'11
## 5.5085  5.5162  5.6484  5.7624  5.7632  5.6286  5.5005  5.3669
##      s'12
## 5.4908
##
## sigma^2 estimated as 0.001402:  log likelihood=263.75
## AIC=-525.5   AICc=-525.47   BIC=-522.54
```

Comme on a fixé les coefficients la procédure ne permet pas de vérifier si le modèle est valide et non simplifiable. On a déjà testé la normalité des résidus, ce sont les résidus du modèle modMS2.

#### 16.1.2.2.4.1.2.5 Explicitation du modèle SARIMA(12,0,1)(0,0,0)[12] final

Explicitons notre modèle final.

```
# Calcul de c
# summary(modeleM3a)$ incompatible atomic vector
# coeftest(modeleM3a) ne fonctionne pas
summary(lmMS)$coefficients

##              Estimate Std. Error  t value    Pr(>|t|)
## mois.January  5.401390  0.1269105 42.56064 5.868475e-79
## mois.February 5.389404  0.1269105 42.46619 7.713919e-79
## mois.March    5.529700  0.1269105 43.57167 3.246563e-80
## mois.April    5.508500  0.1269105 43.40462 5.216380e-80
## mois.May      5.516196  0.1269105 43.46526 4.390623e-80
## mois.June     5.648411  0.1269105 44.50706 2.348482e-81
## mois.July     5.762422  0.1269105 45.40542 1.972012e-82
## mois.August   5.763196  0.1269105 45.41152 1.939412e-82
## mois.September 5.628631  0.1269105 44.35120 3.625654e-81
## mois.October  5.500541  0.1269105 43.34191 6.235393e-80
## mois.November 5.366892  0.1269105 42.28881 1.291005e-78
## mois.December 5.490826  0.1269105 43.26535 7.755142e-80

summary(lmMS)$coefficients[1:12,1] # ou summary(lmMS)$coefficients[,1]

##      mois.January  mois.February  mois.March  mois.April  mois.May
##      5.401390      5.389404      5.529700      5.508500      5.516196
##      mois.June    mois.July    mois.August mois.September mois.October
##      5.648411      5.762422      5.763196      5.628631      5.500541
##      mois.November mois.December
##      5.366892      5.490826

c <- mean(summary(lmMS)$coefficients[1:12,1])
c

## [1] 5.542176
```

$$\forall t \in T \quad \Delta(\log(X_t) - c - s(t)) = B_t$$

avec

- $c = \frac{1}{r} \sum_{i=1}^r s'_i = \frac{1}{12} \sum_{i=1}^{12} \text{summary}(\text{lmMS})\$coefficients[i,1] = 5.542176$
- $s(t) = \sum_{i=1}^r s_i \chi_{i+r\mathbb{Z}} = \sum_{i=1}^r (s'_i - c) \chi_{i+r\mathbb{Z}} = \sum_{i=1}^{12} (\text{summary}(\text{lmMS})\$coefficients[i,1] - c) \chi_{i+12\mathbb{Z}}$   
où
 

$s_1 = -0.1408$	$s_2 = -0.1528$	$s_3 = -0.0125$	$s_4 = -0.0337$
$s_5 = -0.026$	$s_6 = 0.1062$	$s_7 = 0.2202$	$s_8 = 0.221$
$s_9 = 0.0865$	$s_{10} = -0.0416$	$s_{11} = -0.1753$	$s_{12} = -0.0513$
- $(B_t)_{t \in T}$  un ARMA(12,1) défini par

$$\begin{aligned}
 B_t &= \sum_{k=1}^p \varphi_k B_{t-k} + \sum_{k=0}^q \theta_k \varepsilon_{t-k} \\
 &= \sum_{k=1}^{12} \text{coef test}(\text{modMS2})[k,1] B_{t-k} + \varepsilon_t + \text{coef test}(\text{modMS2})[13,1] \varepsilon_{t-1}
 \end{aligned}$$

où

$\varphi_1 = 0.1275$	$\varphi_2 = 0.0569$	$\varphi_3 = -0.0466$	$\varphi_4 = -0.0375$
$\varphi_5 = 0.0683$	$\varphi_6 = 0.0201$	$\varphi_7 = -0.09$	$\varphi_8 = -0.0421$
$\varphi_9 = 0.0756$	$\varphi_{10} = -0.0371$	$\varphi_{11} = 0.0907$	$\varphi_{12} = 0.5794$
$\theta_1 = -0.4573$			

et

$$\varepsilon_t \sim \mathcal{N}(0, 0.0014)$$

#### 16.1.2.2.4.1.2.6 Prévisions

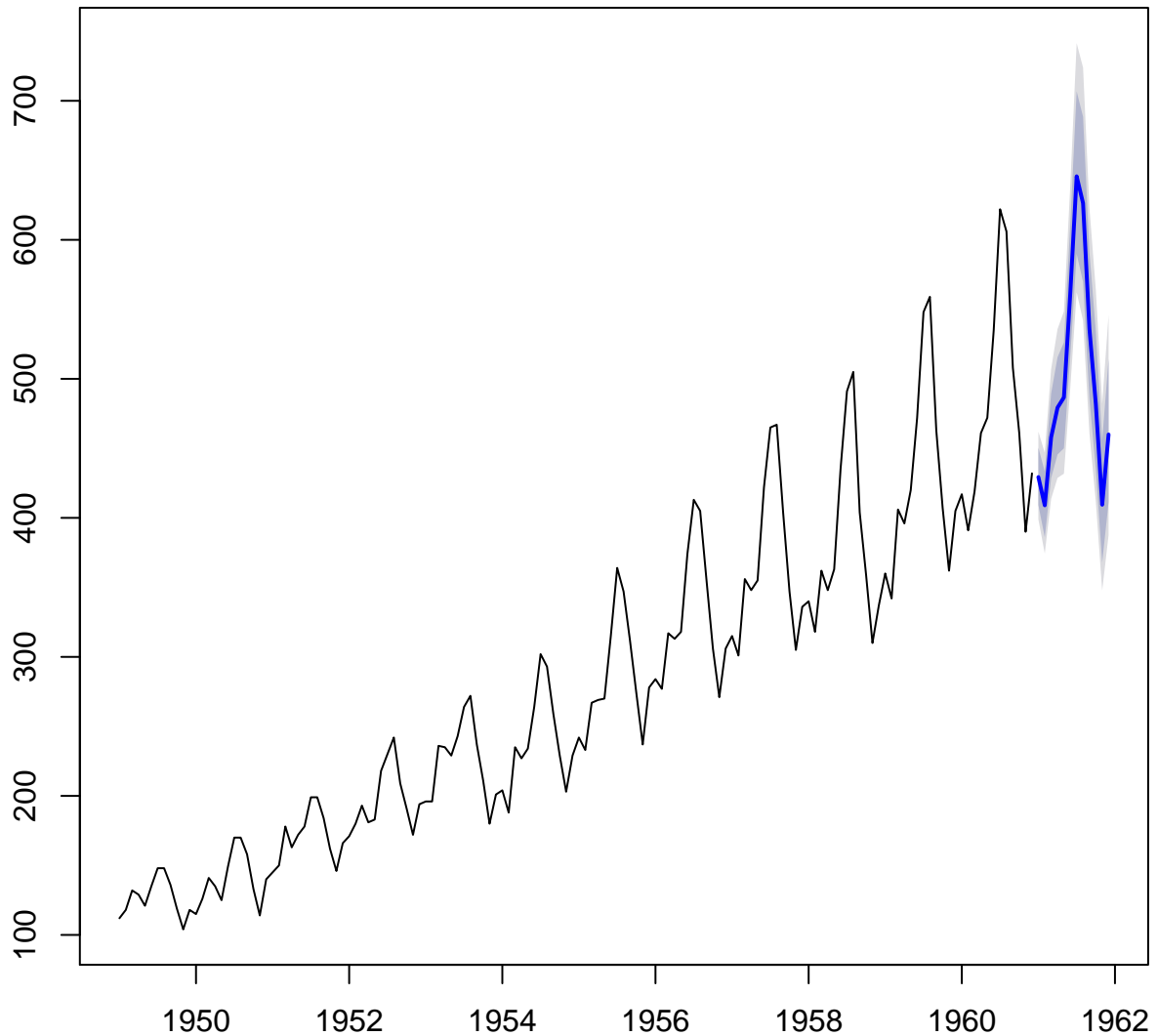
Calculons les prévisions associées à ce modèle.

```
prevM3a <- forecast(modeleM3a, xreg=MatInd, h=12)

## Warning in forecast.forecast_ARIMA(modeleM3a, xreg = MatInd, h = 12): xreg contains different
## column names from the xreg used in training. Please check that the regressors are in the same order.

plot(prevM3a)
```

## Forecasts from Regression with ARIMA(12,1,1) errors



### 16.1.2.2.4.2 Par régression harmonique

Supposons donc qu'il existe  $m \in \mathbb{R}^T$ ,  $(c, \beta_1, \beta_2) \in \mathbb{R}^3$  et  $(B_t)_{t \in T}$  un processus stationnaire centré tel que

$$\forall t \in T \quad \log(X_t) = m(t) + c + \sum_{k=1}^3 \alpha_k \cos\left(\frac{2k\pi(t-1)}{r}\right) + \beta_k \sin\left(\frac{2k\pi(t-1)}{r}\right) + B_t$$

```
har. <- harmonic(log(AirPassengers), 3)
lmRH <- lm(log(AirPassengers)~har.)
lmRH

##
## Call:
## lm(formula = log(AirPassengers) ~ har.)
```

```
##
## Coefficients:
##      (Intercept)  har.cos(2*pi*t)  har.cos(4*pi*t)  har.cos(6*pi*t)
##           5.54218         -0.15756           0.04666          -0.01880
## har.sin(2*pi*t)  har.sin(4*pi*t)  har.sin(6*pi*t)
##          -0.00942           0.04166          -0.03736
```

#### 16.1.2.2.4.2.1 Analyse des résidus

Assurons-nous que les conditions de validité de la régression sont vérifiées.

```
bgtest(lmRH)

##
## Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lmRH
## LM test = 137.67, df = 1, p-value < 2.2e-16

hmcetest(lmRH)

##
## Harrison-McCabe test
##
## data:  lmRH
## HMC = 0.53833, p-value = 0.752
```

Les résidus ne sont pas issus d'un bruit blanc.

Testons s'ils sont issus d'un processus stationnaire.

```
kpss.test(lmRH$res)

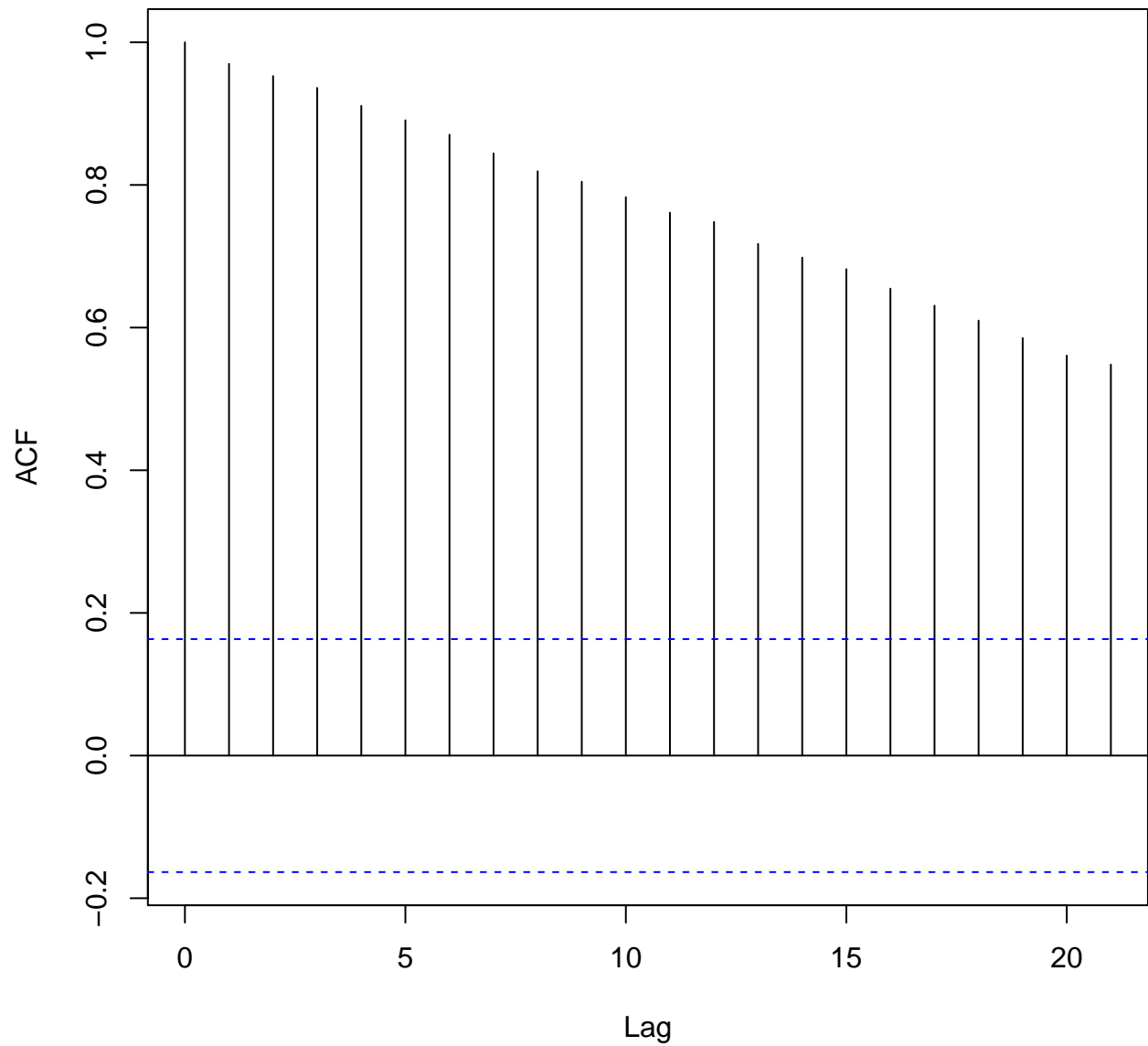
## Warning in kpss.test(lmRH$res): p-value smaller than printed p-value

##
## KPSS Test for Level Stationarity
##
## data:  lmRH$res
## KPSS Level = 2.9254, Truncation lag parameter = 4, p-value = 0.01
```

Les résidus ne sont pas stationnaires. Il reste en effet une composante de tendance que l'on visualise sur le tracé de l'acf.

```
acf(lmRH$res)
```

### Series lmRH\$res



#### 16.1.2.2.4.2.2 Étude de la composante tendancielle

##### 16.1.2.2.4.2.2.1 Identification de la nature de la tendance

```
adf.test(lmRH$res)

##
## Augmented Dickey-Fuller Test
##
## data: lmRH$res
## Dickey-Fuller = -2.8593, Lag order = 5, p-value = 0.2186
## alternative hypothesis: stationary

adf.test(diff(lmRH$res))
```

```
## Warning in adf.test(diff(lmRH$res)): p-value smaller than printed p-value

##
## Augmented Dickey-Fuller Test
##
## data: diff(lmRH$res)
## Dickey-Fuller = -6.9298, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary

kpss.test(diff(lmRH$res))

## Warning in kpss.test(diff(lmRH$res)): p-value greater than printed p-value

##
## KPSS Test for Level Stationarity
##
## data: diff(lmRH$res)
## KPSS Level = 0.029269, Truncation lag parameter = 4, p-value = 0.1
```

Les réponses aux tests sont *Oui*, *Non*, *Oui*. Une modélisation par tendance stochastique d'ordre 1 est préconisée. Explicitons notre modèle. Il existe  $(d, \beta_1, \beta_2) \in \mathbb{N}^* \times \mathbb{R}^2$  et un processus stationnaire centré  $(B_t)_{t \in T}$  tel que

$$\forall t \in T \quad \Delta^d \left( \log(X_t) - c - \sum_{k=1}^3 \left( \alpha_k \cos \left( \frac{2k\pi(t-1)}{r} \right) + \beta_k \sin \left( \frac{2k\pi(t-1)}{r} \right) \right) \right) = B_t$$

**Remarque** Une autre méthode orienterait vers un autre modèle :

```
pp.test(lmRH$res)

## Warning in pp.test(lmRH$res): p-value smaller than printed p-value

##
## Phillips-Perron Unit Root Test
##
## data: lmRH$res
## Dickey-Fuller Z(alpha) = -68.452, Truncation lag parameter = 4, p-value
## = 0.01
## alternative hypothesis: stationary

pp.test(diff(lmRH$res))

## Warning in pp.test(diff(lmRH$res)): p-value smaller than printed p-value

##
## Phillips-Perron Unit Root Test
##
## data: diff(lmRH$res)
## Dickey-Fuller Z(alpha) = -176.72, Truncation lag parameter = 4, p-value
## = 0.01
## alternative hypothesis: stationary

kpss.test(diff(lmRH$res))

## Warning in kpss.test(diff(lmRH$res)): p-value greater than printed p-value

##
## KPSS Test for Level Stationarity
##
## data: diff(lmRH$res)
## KPSS Level = 0.029269, Truncation lag parameter = 4, p-value = 0.1
```

Les réponses sont *Non, Non, Oui* donc une modélisation par tendance linéaire est préconisée. Nous avons déjà proposé cette modélisation.

#### 16.1.2.2.4.2.2 Identification du degré de différenciation

Différencions la série.

```
kpss.test(diff(lmRH$res))

## Warning in kpss.test(diff(lmRH$res)): p-value greater than printed p-value

##
## KPSS Test for Level Stationarity
##
## data: diff(lmRH$res)
## KPSS Level = 0.029269, Truncation lag parameter = 4, p-value = 0.1
```

La série différenciée est modélisable par un ARMA(p,q).

```
eacf(lmRH$res, ar.max=7, ma.max=15)

## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
## 0 x x x x x x x x x x x x x x x
## 1 x o x x o o o x x o x x x o x x
## 2 x x o x o x o x x o o x x o x x
## 3 x o x x o o o x x o o x x o o x
## 4 x o x x x x x x x o o x o o o o
## 5 x x o x o o o x o o o x o o o o
## 6 o x o x o o o x o o o x o o o o
## 7 x o x x o o o x x o o x o o o o
```

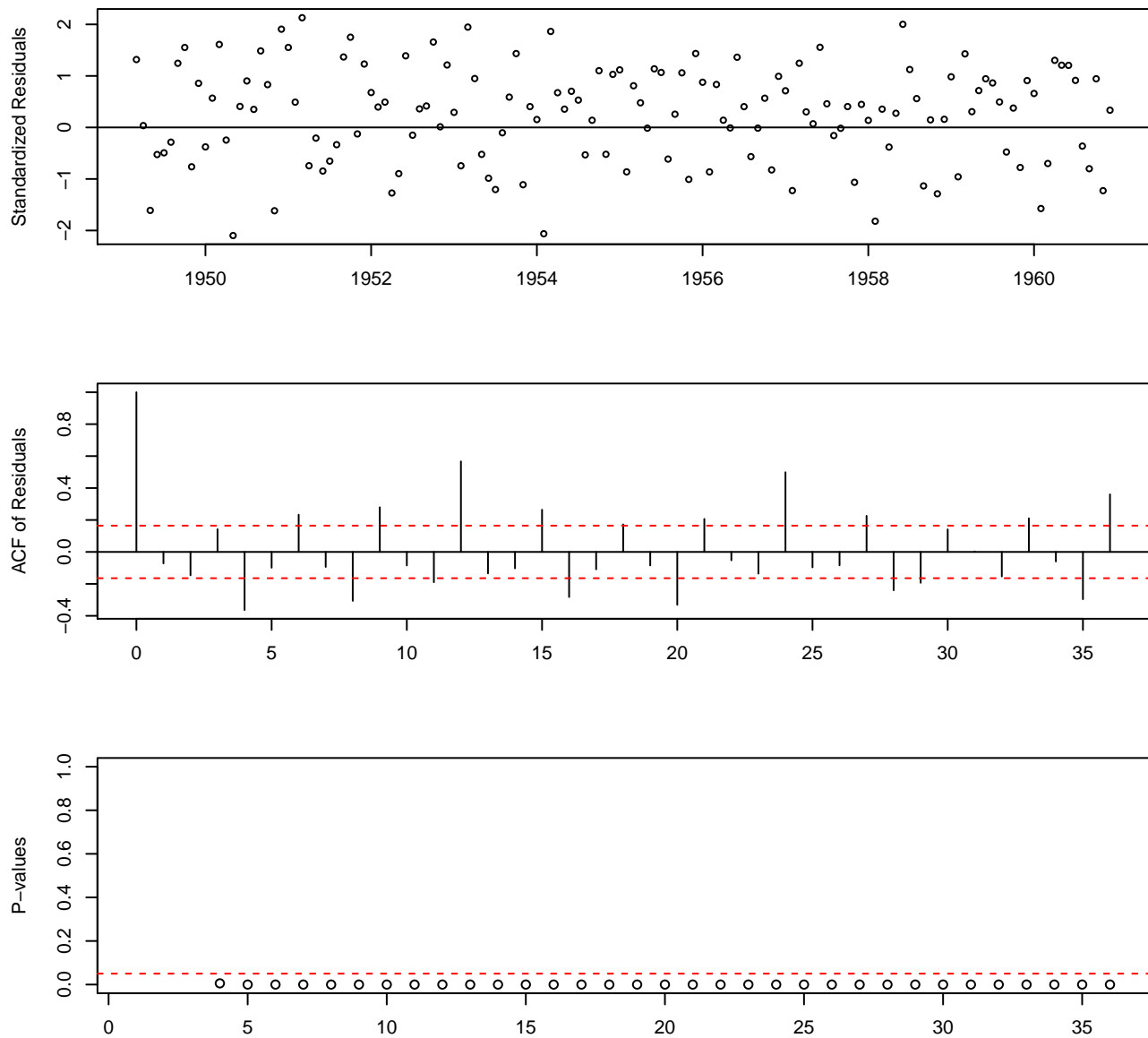
L'eacf est difficilement interprétable et ne permet pas de dégager de modèle candidat. Cherchons un modèle candidat à l'aide de la procédure `auto.arima()`.

```
auto.arima(lmRH$res)

## Series: lmRH$res
## ARIMA(2,1,1) with drift
##
## Coefficients:
##          ar1          ar2          ma1          drift
##         -1.0620      -0.5218       0.645       0.0091
## s.e.       0.1545       0.0735       0.182       0.0031
##
## sigma^2 estimated as 0.003438: log likelihood=204.45
## AIC=-398.89   AICc=-398.45   BIC=-384.08
```

La fonction `auto.arima()` suggère un ARMA(1,2). Construisons le modèle proposé.

```
modRH <- Arima(AirPassengers, xreg=MatRH3, order=c(1,1,2), lambda=0)
tsdiag.Arima(modRH, gof.lag=round(length(log(AirPassengers))/4))
```



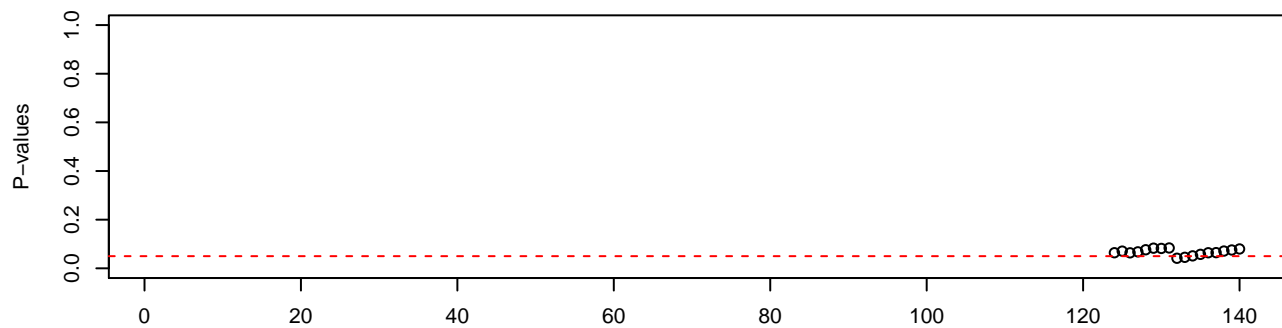
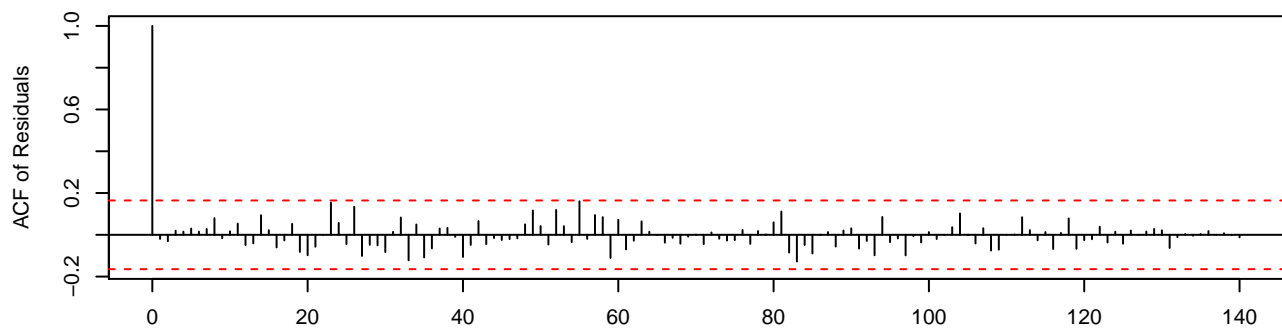
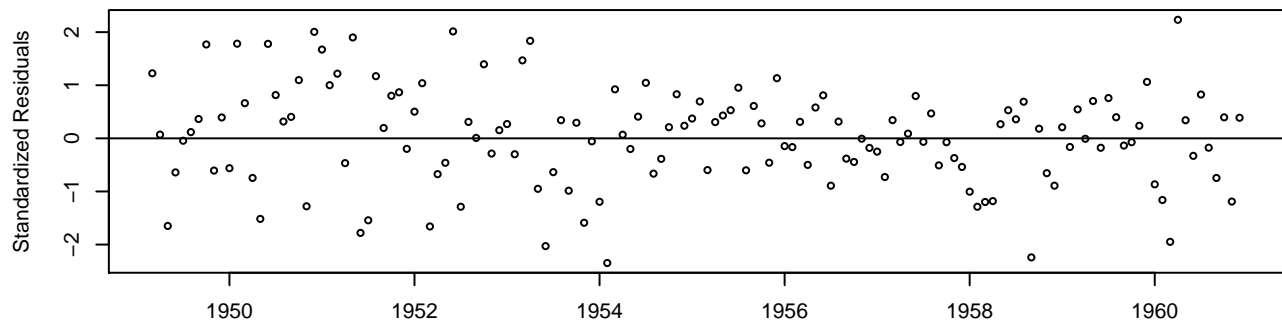
Le modèle n'est pas valide.  
Faisons appel à la procédure MINIC.

```
armaselect(lmRH$res, nbmod=4)
```

```
##      p  q      sbc
## [1,] 13  9 -875.2329
## [2,] 10 14 -874.3085
## [3,] 13 10 -873.9766
## [4,] 13  4 -873.0497
```

La procédure MINIC suggère de préférence un ARMA(14,2).

```
modRH1 <- Arima(AirPassengers, xreg=MatRH3, order=c(14,1,2), lambda=0)
tsdiag.Arima(modRH1, gof.lag=140)
```



```
# round(length(AirPassengers)/4) pas suffisant
```

Le modèle ARMA(14,2) est valide.

```
t_stat(modRH1)
```

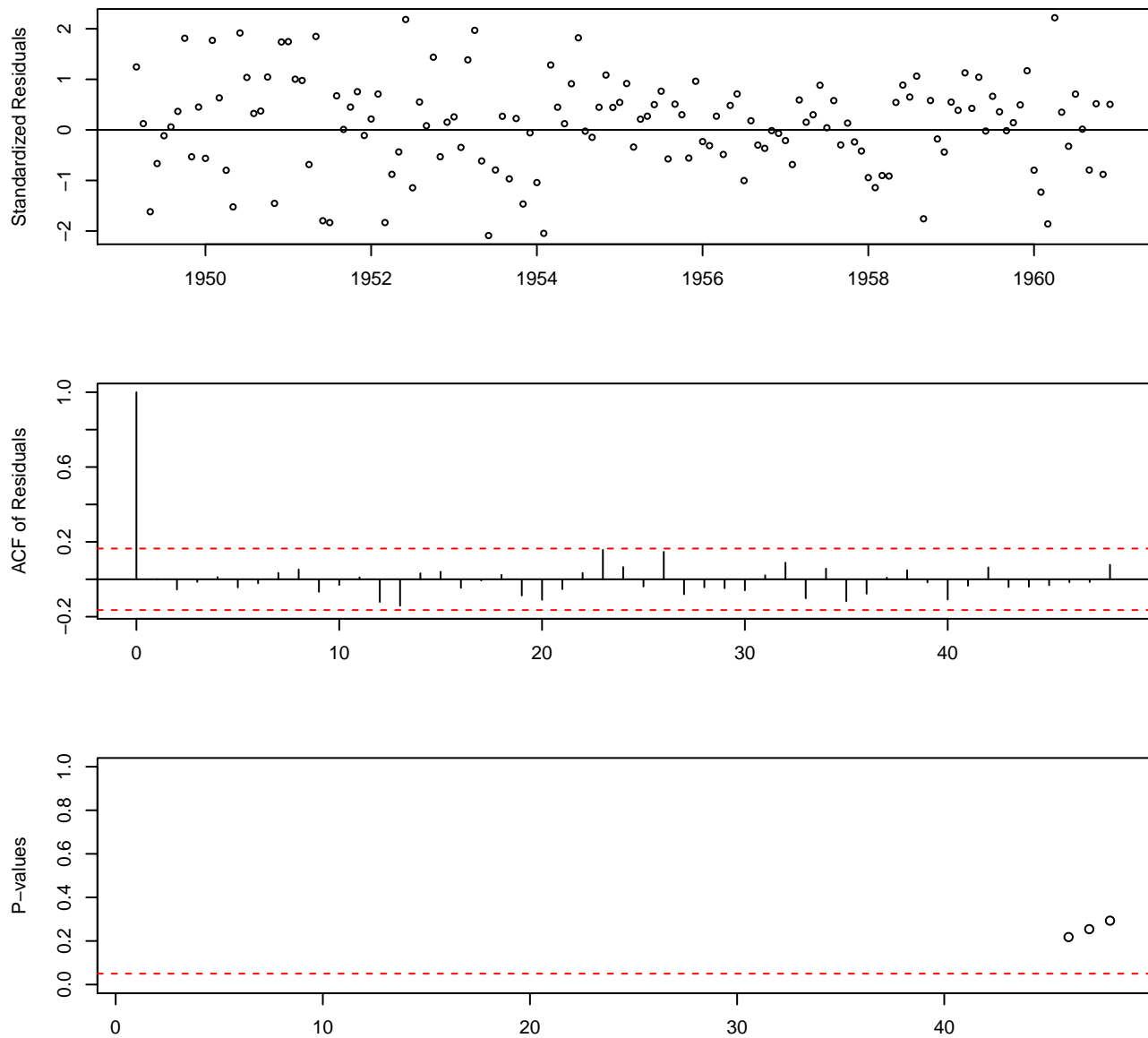
```
##          ar1      ar2      ar3      ar4      ar5      ar6      ar7
## t.stat -0.911021 8.006237 1.073373 -2.509169 1.126299 2.536373 -0.519166
## p.val  0.362284 0.000000 0.283104 0.012102 0.260039 0.011201 0.603645
##          ar8      ar9      ar10     ar11     ar12     ar13     ar14
## t.stat -2.804076 2.332112 2.116658 -2.376197 2.816621 2.186399 -3.970540
## p.val  0.005046 0.019695 0.034289 0.017492 0.004853 0.028786 0.000072
##          ma1      ma2      Cos1      Sin1      Cos2      Sin2      Cos3
## t.stat -1.760999 -5.737755 -14.33193 2.600206 9.06019 9.303244 -3.724533
```

```
## p.val    0.078239  0.000000   0.00000 0.009317 0.00000 0.000000  0.000196
##          Sin3
## t.stat -12.26499
## p.val    0.00000
```

Le modèle ARMA(14,2) est simplifiable en un ARMA(13,2).

```
modRH2 <- Arima(AirPassengers, xreg=MatRH3, order=c(13,1,2), lambda=0)
```

```
## Warning in 1/seq(n - 1, n - lag) * obs^2: longer object length is not a multiple of shorter object
length
## Warning in 1/seq(n - 1, n - lag) * obs^2: longer object length is not a multiple of shorter object
length
## Warning in 1/seq(n - 1, n - lag) * obs^2: longer object length is not a multiple of shorter object
length
```



Le modèle ARMA(13,2) est valide.

#### 16.1.2.2.4.2.2.3 Comparaison des modèles

Vérifions que les résidus sont gaussiens.

```
shapiro.test(modRH1$res)

##
##  Shapiro-Wilk normality test
##
## data:  modRH1$res
## W = 0.99029, p-value = 0.4216

dagoTest(modRH1$res)@test$p.value[1]
```

```
## Omnibus Test
##      0.782672
```

```
shapiro.test(modRH2$res)

##
## Shapiro-Wilk normality test
##
## data:  modRH2$res
## W = 0.98602, p-value = 0.1536

dagoTest(modRH2$res)@test$p.value[1]

## Omnibus Test
##      0.7166346
```

Réalisons un test de rapport de vraisemblance pour savoir si on conserve le modèle simplifié.

```
lrtest(modRH2, modRH1)

## Likelihood ratio test
##
## Model 1: Arima(y = AirPassengers, order = c(13, 1, 2), xreg = MatRH3,
##      lambda = 0)
## Model 2: Arima(y = AirPassengers, order = c(14, 1, 2), xreg = MatRH3,
##      lambda = 0)
##      #Df LogLik Df  Chisq Pr(>Chisq)
## 1    22 273.46
## 2    23 274.79  1  2.6508    0.1035
```

L'hypothèse nulle  $H_0$  est acceptée donc on conserve le modèle simplifié car il est valide.

#### 16.1.2.2.4.2.2.4 Modèle complet SARIMA(13,0,2)(0,1,0)[12]

Construisons le modèle complet.

```
modeleM3b <- Arima(AirPassengers, xreg=MatRH3, order=c(13,1,2), lambda=0)
modeleM3b

## Series: AirPassengers
## Regression with ARIMA(13,1,2) errors
## Box Cox transformation: lambda= 0
##
## Coefficients:
##      ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8
##    -0.6804  0.3153  0.0765 -0.2886 -0.0020  0.2697  0.0306 -0.2490
## s.e.   0.1468  0.1703  0.0998  0.0934  0.0941  0.0938  0.0934  0.0982
##      ar9      ar10      ar11      ar12      ar13      ma1      ma2      Cos1      Sin1
##    0.1635  0.3202 -0.1145  0.3282  0.5077  0.3148 -0.4211 -0.1481  0.0277
## s.e.   0.0966  0.0968  0.1164  0.1017  0.0789  0.1663  0.1498  0.0121  0.0128
##      Cos2      Sin2      Cos3      Sin3
##    0.0572  0.0592 -0.0084 -0.0277
## s.e.   0.0067  0.0067  0.0020  0.0020
##
## sigma^2 estimated as 0.001417:  log likelihood=273.46
## AIC=-502.93  AICc=-494.49  BIC=-437.74
```

Le logiciel **R** n'estime pas la constante  $c$ , bien qu'on ait conservé l'option par défaut `include.mean=T`. En fait, on peut, dans ce modèle, prendre  $c = 0$  dès que  $d > 0$  ou  $D > 0$ . Explicitons notre modèle final.

```

coeftest(modeleM3b)

##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  -0.6803631  0.1467691  -4.6356 3.559e-06 ***
## ar2   0.3153447  0.1703421   1.8512 0.0641345 .
## ar3   0.0764587  0.0998367   0.7658 0.4437732
## ar4  -0.2886327  0.0934497  -3.0886 0.0020107 **
## ar5  -0.0019559  0.0941110  -0.0208 0.9834185
## ar6   0.2696924  0.0937691   2.8761 0.0040258 **
## ar7   0.0306386  0.0933579   0.3282 0.7427722
## ar8  -0.2489855  0.0981706  -2.5363 0.0112045 *
## ar9   0.1634767  0.0966227   1.6919 0.0906637 .
## ar10  0.3202096  0.0967605   3.3093 0.0009353 ***
## ar11 -0.1145308  0.1163583  -0.9843 0.3249710
## ar12  0.3281584  0.1017316   3.2257 0.0012565 **
## ar13  0.5077413  0.0788788   6.4370 1.219e-10 ***
## ma1   0.3148374  0.1663057   1.8931 0.0583413 .
## ma2  -0.4210653  0.1498079  -2.8107 0.0049434 **
## Cos1 -0.1481418  0.0121310 -12.2118 < 2.2e-16 ***
## Sin1  0.0276750  0.0128399   2.1554 0.0311306 *
## Cos2  0.0571757  0.0066698   8.5723 < 2.2e-16 ***
## Sin2  0.0592138  0.0067277   8.8014 < 2.2e-16 ***
## Cos3 -0.0084177  0.0020282  -4.1504 3.319e-05 ***
## Sin3 -0.0276922  0.0020343 -13.6128 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

names(modeleM3b)

## [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
## [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
## [13] "nobs"      "model"     "aicc"      "bic"       "xreg"      "lambda"
## [19] "x"         "fitted"

```

$$\forall t \in T \quad \Delta(\log(X_t) - s(t)) = B_t$$

avec

$$\bullet \quad s(t) = \sum_{k=1}^3 \alpha_k \cos\left(\frac{2k\pi(t-1)}{12}\right) + \beta_k \sin\left(\frac{2k\pi(t-1)}{12}\right)$$

où

$$\forall k \in \{1, 2, 3\} \quad \alpha_k = \text{coeftest}(\text{modeleM3b})[14+2k, 1] \quad \text{et} \quad \beta_k = \text{coeftest}(\text{modeleM3b})[15+2k, 1]$$

$$\begin{array}{lll} \alpha_1 = -0.1481 & \alpha_2 = 0.0572 & \alpha_3 = -0.0084 \\ \beta_1 = 0.0277 & \beta_2 = 0.0592 & \beta_3 = -0.0277 \end{array}$$

•  $(B_t)_{t \in T}$  un ARMA(13,2) défini par

$$\begin{aligned} B_t &= \sum_{k=1}^p \varphi_k B_{t-k} + \sum_{k=0}^q \theta_k \varepsilon_{t-k} \\ &= \sum_{k=1}^{13} \text{coeftest}(\text{modeleM3b})[k, 1] B_{t-k} + \varepsilon_t + \sum_{k=1}^2 \text{coeftest}(\text{modeleM3b})[13+k, 1] \varepsilon_{t-k} \end{aligned}$$

où

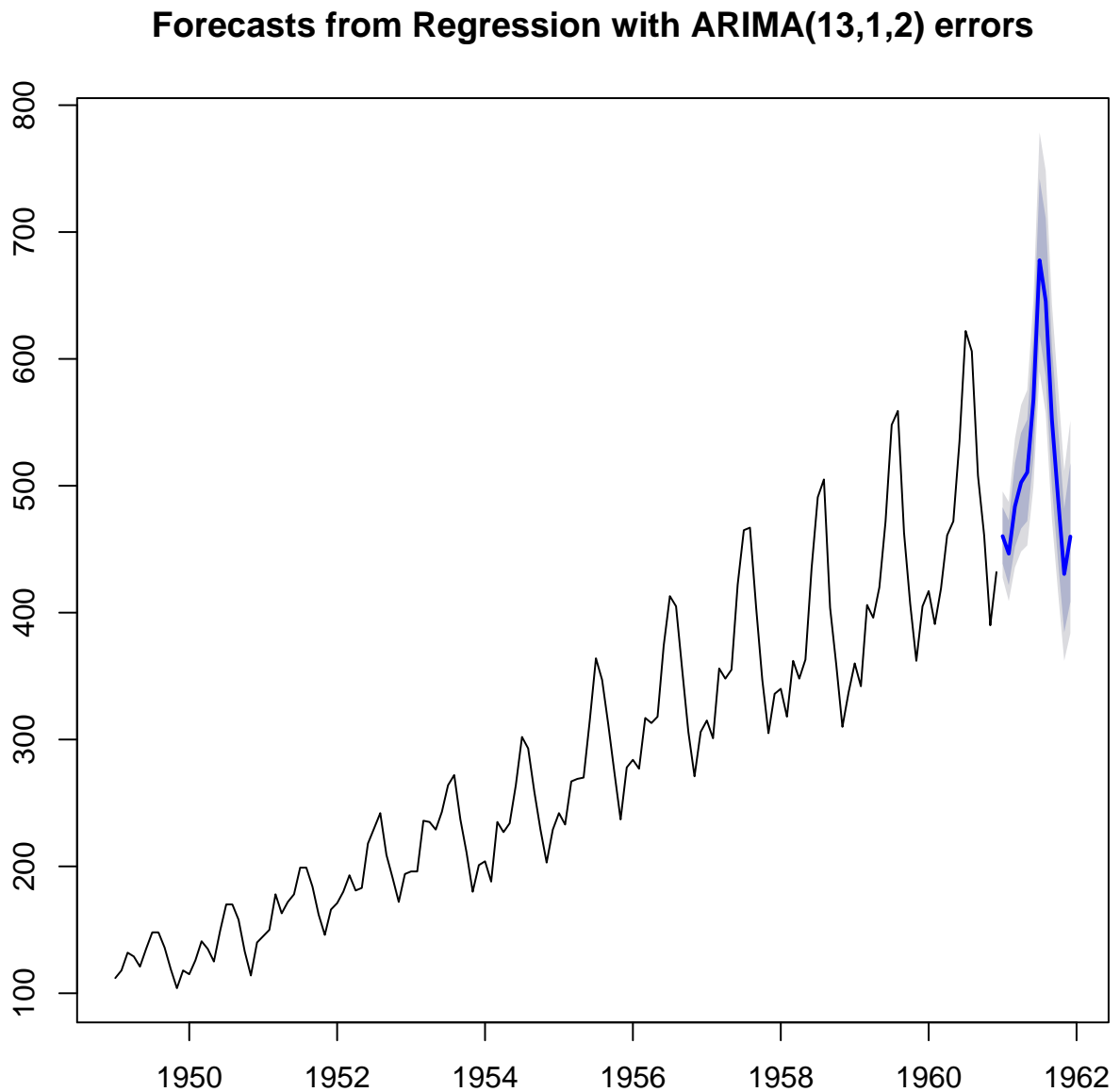
$\varphi_1 = -0.6804$	$\varphi_2 = 0.3153$	$\varphi_3 = 0.0765$	$\varphi_4 = -0.2886$	$\varphi_5 = -0.002$
$\varphi_6 = 0.2697$	$\varphi_7 = 0.0306$	$\varphi_8 = -0.249$	$\varphi_9 = 0.1635$	$\varphi_{10} = 0.3202$
$\varphi_{11} = -0.1145$	$\varphi_{12} = 0.3282$	$\varphi_{13} = 0.5077$	$\theta_1 = 0.3148$	$\theta_2 = -0.4211$

et

$$\varepsilon_t \sim \mathcal{N}(0, 0.0014)$$

#### 16.1.2.2.4.2.2.5 Prévisions

```
prevM3b <- forecast(modeleM3b, xreg=MatRH3P, h=12)
plot(prevM3b)
```



### 16.1.3 Modélisation par transformation de Box-Cox (Tendance et saisonnalité stochastiques)

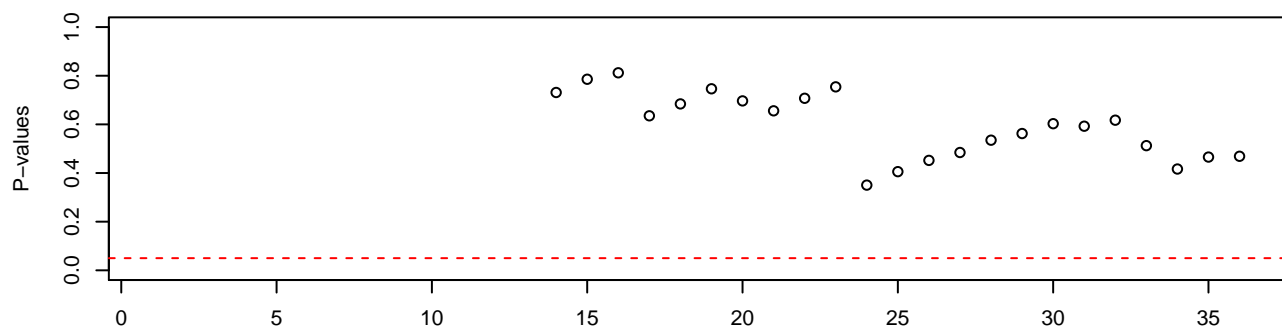
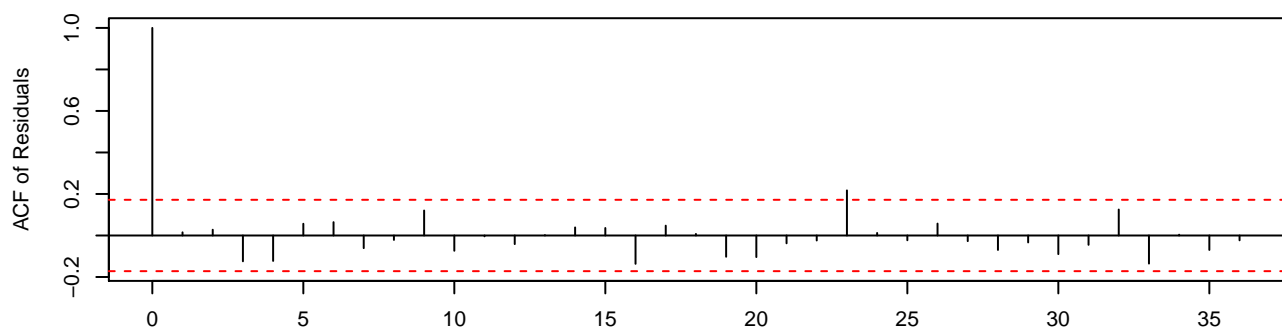
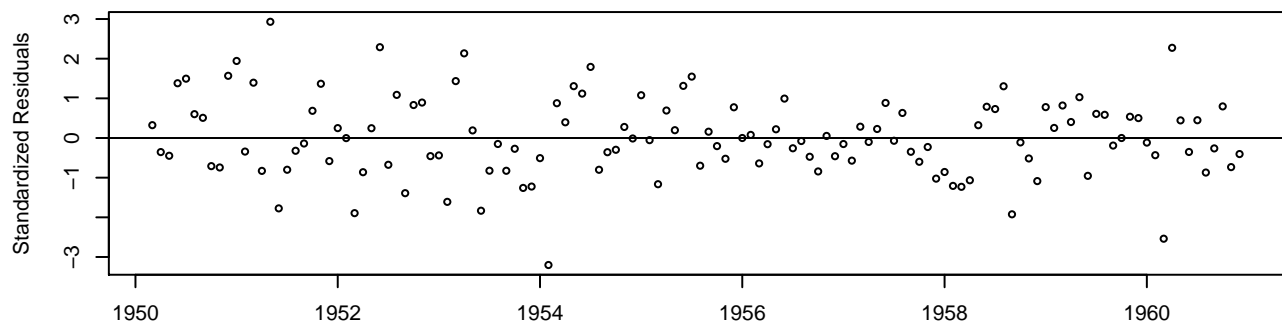
D'après les test de Canova-Hansen (CH) et de (OCSB), la composante saisonnière de `log(AirPassengers)` n'est pas stochastique. On construit néanmoins ce modèle pour le comparer aux autres.

```
modeleM4 <- auto.arima(log(AirPassengers))
modeleM4

## Series: log(AirPassengers)
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##          ma1      sma1
##       -0.4018  -0.5569
## s.e.   0.0896   0.0731
##
## sigma^2 estimated as 0.001371:  log likelihood=244.7
## AIC=-483.4   AICc=-483.21   BIC=-474.77
```

On vérifie que le modèle proposé par la procédure `auto.arima()` est valide.

```
tsdiag.Arima(modeleM4, gof.lag=round(length(log(AirPassengers))/4))
```



Le modèle est valide.

```
t_stat(modeleM4)

##          ma1      sma1
## t.stat -4.482494 -7.618978
## p.val  0.000007  0.000000
```

Le modèle n'est pas simplifiable.

```
dagoTest(modeleM4$res)$test$p.value[1]

## Omnibus Test
##      0.1267659

shapiro.test(modeleM4$res)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: modeleM4$res  
## W = 0.98637, p-value = 0.1674
```

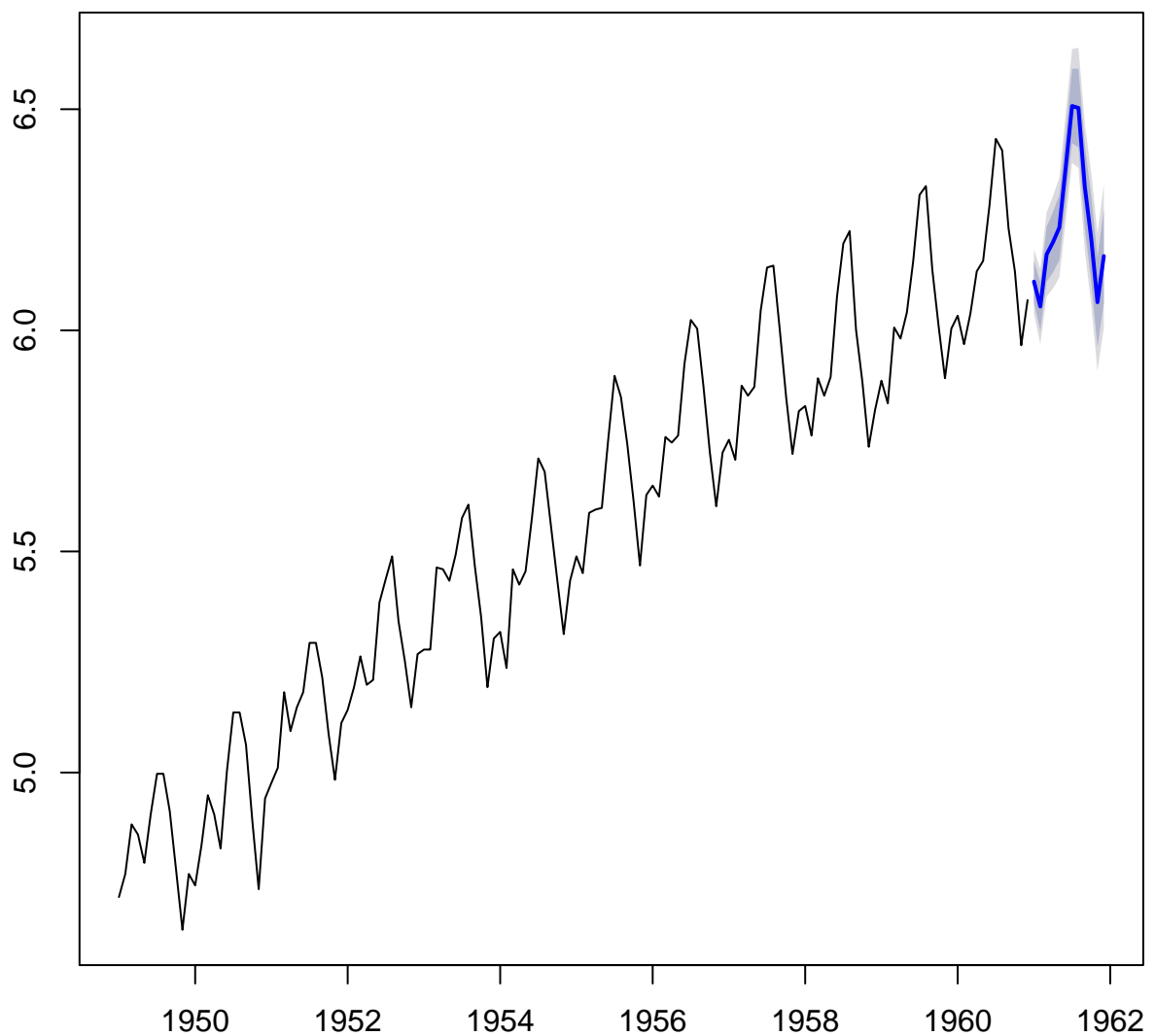
Les résidus sont gaussiens.

#### 16.1.3.1 Prévisions

Calculons les prévisions associées à ce modèle.

```
prevM4 <- forecast(modeleM4, h=12)  
plot(prevM4)
```

### Forecasts from ARIMA(0,1,1)(0,1,1)[12]

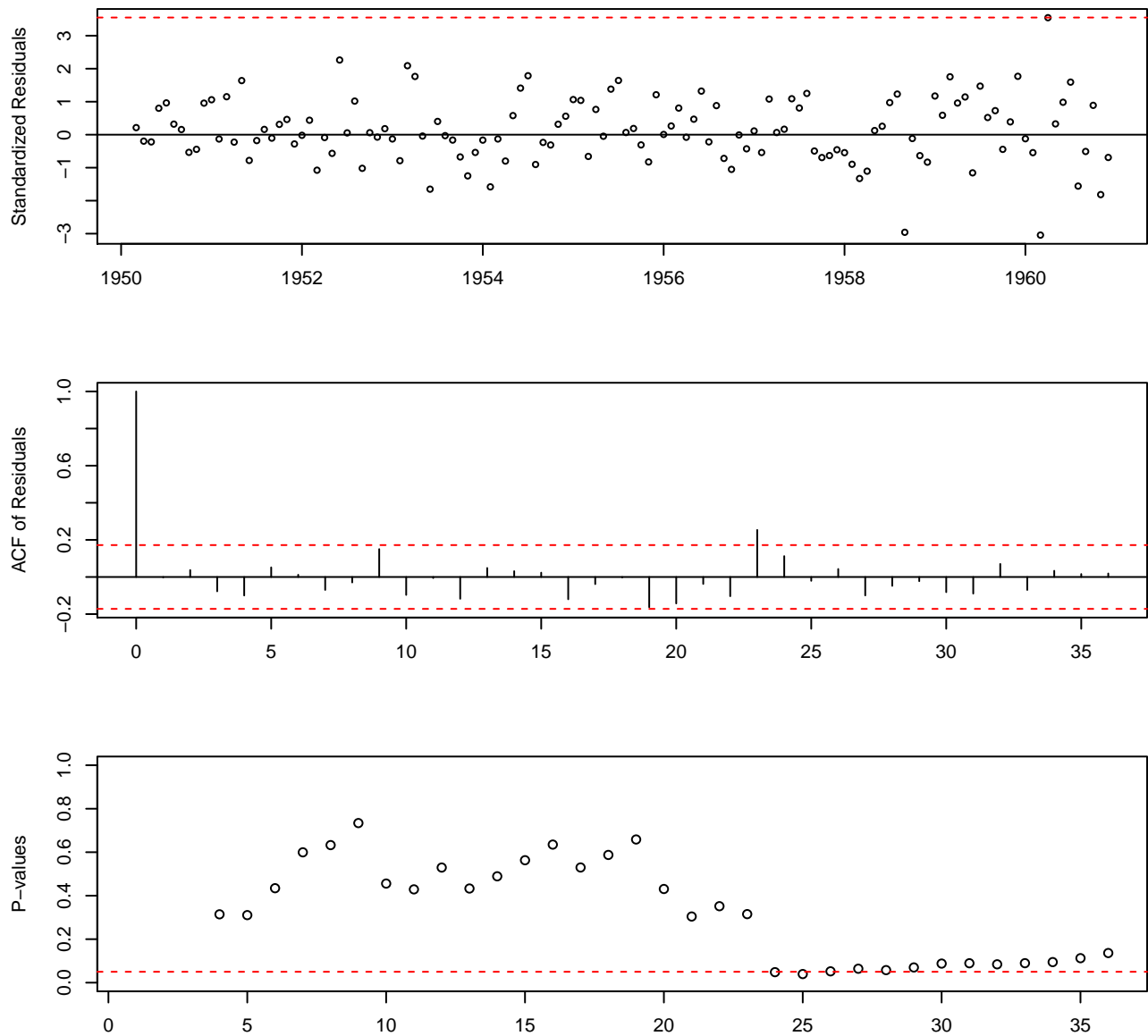


### 16.1.4 Tendance et saisonnalité stochastiques

```
modeleM4A <- auto.arima(AirPassengers)
modeleM4A

## Series: AirPassengers
## ARIMA(2,1,1)(0,1,0)[12]
##
## Coefficients:
##          ar1      ar2      ma1
##      0.5960  0.2143 -0.9819
## s.e.  0.0888  0.0880  0.0292
##
## sigma^2 estimated as 132.3:  log likelihood=-504.92
## AIC=1017.85   AICc=1018.17   BIC=1029.35

tsdiag.Arima(modeleM4A, gof.lag=round(length(AirPassengers)/4))
```



Ce modèle-ci est tout juste valide.

```
t_stat(modeleM4A)

##          ar1      ar2      ma1
## t.stat 6.710138 2.435875 -33.62396
## p.val  0.000000 0.014856  0.00000
```

Ce modèle est non simplifiable.

Dans le cadre d'une modélisation classique on écarterait ce modèle. Ici on continue à considérer ce modèle, dans le simple but de le comparer à tous les autres modèles déjà construits. On continue donc à préciser le modèle en regardant si les résidus sont gaussiens.

```

shapiro.test(modeleM4A$res)

##
##  Shapiro-Wilk normality test
##
## data:  modeleM4A$res
## W = 0.97235, p-value = 0.005165

dagoTest(modeleM4A$res)$p.value[1]

## Omnibus Test
## 0.01978571

```

Les résidus ne sont pas gaussiens.

Explicitons notre modèle SARIMA(2,1,1)(0,1,0)[12].

$$\forall t \in T \quad (1 - L)(1 - L^{12})(\varphi_1 L + \varphi_2 L^2)(1)X_t = (\theta_1 L)(1)\varepsilon_t$$

*#test*

# Démonstrations

### Démonstration – Fonction d'autocovariance d'un processus MA( $q$ )

Remarquons que,

$$\forall (t_1, t_2) \in T^2 \quad \mathbb{E}(\varepsilon_{t_1} \varepsilon_{t_2}) = \text{Cov}(\varepsilon_{t_1}, \varepsilon_{t_2}) - \mathbb{E}(\varepsilon_{t_1})\mathbb{E}(\varepsilon_{t_2}) = \text{Cov}(\varepsilon_{t_1}, \varepsilon_{t_2}) = \begin{cases} 0 & \text{si } t_1 \neq t_2 \\ \text{Var}(\varepsilon) = \sigma^2 & \text{sinon} \end{cases}$$

Soit  $(h, t) \in \mathbb{N}^2$ . On a

$$\begin{aligned} \gamma(h) &= \text{Cov}(X_t, X_{t+h}) \\ &= \mathbb{E}(X_t X_{t+h}) - \mathbb{E}(X_t)\mathbb{E}(X_{t+h}) \\ &= \mathbb{E}(X_t X_{t+h}) \\ &= \mathbb{E}\left(\left(\varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}\right) \left(\varepsilon_{t+h} + \sum_{j=1}^q \theta_j \varepsilon_{t+h-j}\right)\right) \\ &= \mathbb{E}(\varepsilon_t \varepsilon_{t+h}) + \mathbb{E}\left(\varepsilon_t \sum_{j=1}^q \theta_j \varepsilon_{t+h-j}\right) + \mathbb{E}\left(\varepsilon_{t+h} \sum_{i=1}^q \theta_i \varepsilon_{t-i}\right) + \mathbb{E}\left(\left(\sum_{i=1}^q \theta_i \varepsilon_{t-i}\right) \left(\sum_{j=1}^q \theta_j \varepsilon_{t+h-j}\right)\right) \\ &= \mathbb{E}(\varepsilon_t \varepsilon_{t+h}) + \sum_{j=1}^q \theta_j \mathbb{E}(\varepsilon_t \varepsilon_{t+h-j}) + \sum_{i=1}^q \theta_i \mathbb{E}(\varepsilon_{t+h} \varepsilon_{t-i}) + \sum_{i=1}^q \sum_{j=1}^q \theta_i \theta_j \mathbb{E}(\varepsilon_{t-i} \varepsilon_{t+h-j}) \\ &= \mathbb{E}(\varepsilon_t \varepsilon_{t+h}) + \sum_{j=1}^q \theta_j \mathbb{E}(\varepsilon_t \varepsilon_{t+h-j}) + \sum_{i=1}^q \sum_{j=1}^q \theta_i \theta_j \mathbb{E}(\varepsilon_{t+h-i} \varepsilon_{t-j}) \end{aligned}$$

- Supposons que  $h > q$ .
  - \* On a  $q > 0$  donc  $h > 0$  en particulier  $h \neq 0$ . Pour tout  $t \in T$ , on en déduit que  $t + h \neq t$  puis que  $\mathbb{E}(\varepsilon_t \varepsilon_{t+h}) = 0$ .
  - \* Soit  $j \in \{1, \dots, q\}$ . On a  $j \leq q$  et  $q < h$  donc  $j < h$ . En particulier,  $j \neq h$  donc  $t \neq t + h - j$ . On en déduit que  $\mathbb{E}(\varepsilon_t \varepsilon_{t+h-j}) = 0$ . Finalement,

$$\sum_{j=1}^q \theta_j \mathbb{E}(\varepsilon_t \varepsilon_{t+h-j}) = 0$$

- \* On a  $h - i > q - i$ . Or,  $i \leq q$  donc  $q - i \geq 0$ . Par ailleurs  $j \geq 1$  donc  $-j \leq -1$ . On en déduit que  $h - i > -j$ . Finalement, pour tout  $t \in T$ ,  $t + h - i > t - j$  donc  $t + h - i \neq t - j$ .

$$\sum_{i=1}^q \sum_{j=1}^q \theta_i \theta_j \mathbb{E}(\varepsilon_{t+h-i} \varepsilon_{t-j}) = 0$$

On en conclut que

$$\gamma(h) = 0$$

- Sinon  $h \leq q$ .
  - \* Soit  $(t, i, j) \in T \times \{1, \dots, q\}^2$ ,  $t + h - i = t - j$  si, et seulement si,  $j = i - h$ .

$$\sum_{i=1}^q \sum_{j=1}^q \theta_i \theta_j \mathbb{E}(\varepsilon_{t+h-i} \varepsilon_{t-j}) = \sum_{\substack{1 \leq i \leq q \\ i-h \in \{1, \dots, q\}}} \theta_i \theta_{i-h} \mathbb{E}(\varepsilon_{t+h-i}^2) = \sum_{i=h+1}^q \theta_i \theta_{i-h} \mathbb{E}(\varepsilon_{t+h-i}^2) = \sigma^2 \sum_{i=h+1}^q \theta_i \theta_{i-h} = \sigma^2 \sum_{k=1}^{q-h} \theta_k \theta_{k+h}$$

- \* Soit  $j \in \{1, \dots, q\}$ . On a  $t = t + h - j$  si, et seulement si,  $j = h$ . On en déduit que

$$\sum_{j=1}^q \theta_j \mathbb{E}(\varepsilon_t \varepsilon_{t+h-j}) = \theta_h \mathbb{E}(\varepsilon_t^2) = \theta_h \sigma^2$$

On en conclut que

$$\gamma(h) = \theta_h \sigma^2 + \sigma^2 \sum_{k=1}^{q-h} \theta_k \theta_{k+h}$$

**Démonstration – Fonction d'autocovariance d'un processus MA(1)**

Soit  $\theta_1 \in \mathbb{R}$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible) centré. Notons  $(X_t)_{t \in T}$  le processus MA(1) de paramètre  $\theta$  et d'erreur  $(\varepsilon_t)_{t \in T}$  i.e. défini par

$$\forall t \in T \quad X_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} = (\mathbf{1} + \theta_1 L) \varepsilon_t$$

•

$$\begin{aligned} \gamma(0) &= \text{Cov}(X_t, X_t) \\ &= \text{Var}(X_t) \\ &= \text{Var}(\varepsilon_t + \theta_1 \varepsilon_{t-1}) \\ &= \text{Var}(\varepsilon_t) + \text{Var}(\theta_1 \varepsilon_{t-1}) + 2\text{Cov}(\varepsilon_t, \theta_1 \varepsilon_{t-1}) \\ &= \text{Var}(\varepsilon_t) + \theta_1^2 \text{Var}(\varepsilon_t) + 2\theta_1 \text{Cov}(\varepsilon_t, \varepsilon_{t-1}) \\ &= (1 + \theta_1^2) \text{Var}(\varepsilon_t) \\ &= (1 + \theta_1^2) \sigma^2 \end{aligned}$$

•

$$\begin{aligned} \gamma(1) &= \text{Cov}(X_t, X_{t+1}) \\ &= \text{Cov}(\varepsilon_t + \theta_1 \varepsilon_{t-1}, \varepsilon_{t+1} + \theta_1 \varepsilon_t) \\ &= \text{Cov}(\varepsilon_t, \varepsilon_{t+1}) + \text{Cov}(\varepsilon_t, \theta_1 \varepsilon_t) + \text{Cov}(\theta_1 \varepsilon_{t-1}, \varepsilon_{t+1}) + \text{Cov}(\theta_1 \varepsilon_{t-1}, \theta_1 \varepsilon_t) \\ &= \theta_1 \text{Cov}(\varepsilon_t, \varepsilon_t) \\ &= \theta_1 \text{Var}(\varepsilon_t) \\ &= \theta_1 \sigma^2 \end{aligned}$$

•

$$\begin{aligned} \forall h \in \llbracket 2, +\infty \llbracket \quad \gamma(h) &= \text{Cov}(X_t, X_{t+h}) \\ &= \text{Cov}(\varepsilon_t + \theta_1 \varepsilon_{t-1}, \varepsilon_{t+h} + \theta_1 \varepsilon_{t-1+h}) \\ &= \text{Cov}(\varepsilon_t, \varepsilon_{t+h}) + \text{Cov}(\varepsilon_t, \theta_1 \varepsilon_{t-1+h}) + \text{Cov}(\theta_1 \varepsilon_{t-1}, \varepsilon_{t+h}) + \text{Cov}(\theta_1 \varepsilon_{t-1}, \theta_1 \varepsilon_{t-1+h}) \\ &= \theta_1 \text{Cov}(\varepsilon_t, \varepsilon_{t-1+h}) + \theta_1 \text{Cov}(\varepsilon_{t-1}, \varepsilon_{t+h}) + \theta_1^2 \text{Cov}(\varepsilon_{t-1}, \varepsilon_{t-1+h}) \\ &= 0 \end{aligned}$$

**Démonstration – Fonction d'autocorrélation d'un processus MA(1)**

Soit  $\theta_1 \in \mathbb{R}$  et  $(\varepsilon_t)_{t \in T}$  un bruit blanc (faible) centré. Notons  $(X_t)_{t \in T}$  le processus MA(1) de paramètre  $\theta$  et d'erreur  $(\varepsilon_t)_{t \in T}$  i.e. défini par

$$\forall t \in T \quad X_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} = (\mathbf{1} + \theta_1 L) \varepsilon_t$$

• On a  $\sigma \neq 0$  et

$$\rho(1) = \frac{\gamma(1)}{\gamma(0)} = \frac{\theta_1 \sigma^2}{(1 + \theta_1^2) \sigma^2} = \frac{\theta_1}{(1 + \theta_1^2)}$$

•

$$\forall h \in \llbracket 2, +\infty \llbracket \quad \rho(h) = \frac{\gamma(h)}{\gamma(0)} = 0$$

aaa