Econometrics II

Notes - Midterm

William Radaic Peron

EESP-FGV

September 20, 2020

Contents

1 Introduction **3**
- 1.1 Motivation . . . 3
- 1.2 Statistics with dependence . . . 4
  - 1.2.1 Definition of a time series . . . 4
  - 1.2.2 Unconditional expectation . . . 4
  - 1.2.3 Statistical dependence . . . 4
- 1.3 Asymptotic theory with dependence . . . 5
  - 1.3.1 Stationarity . . . 5
  - 1.3.2 Ergodicity . . . 6
  - 1.3.3 A Central Limit Theorem for time series . . . 7

2 ARMA Models **8**
- 2.1 White noise . . . 8
- 2.2 Moving Average processes . . . 8
  - 2.2.1 Moments of an MA(1) model . . . 8
  - 2.2.2 Some examples of MA(1) processes . . . 9
  - 2.2.3 Deriving the Autocorrelation function of the MA(1) process . . . 9
- 2.3 Generalizing the MA model . . . 10
  - 2.3.1 Moments of an MA(q) process . . . 10
  - 2.3.2 Deriving the Autocorrelation function of the MA(q) process . . . 11
- 2.4 The MA(∞) model . . . 11
- 2.5 The Wold Decomposition . . . 11
- 2.6 Autoregressive models . . . 12
  - 2.6.1 Moments of an AR(1) process . . . 12
  - 2.6.2 Some examples of AR(1) series . . . 12
  - 2.6.3 Autocorrelation function of an AR(1) process . . . 12
  - 2.6.4 Partial Autocorrelation Function . . . 12
  - 2.6.5 Conditions for stationarity . . . 13
- 2.7 Generalizing the AR model . . . 14
  - 2.7.1 Moments of an AR(p) process . . . 14
  - 2.7.2 ACF of an AR(p) process . . . 15
- 2.8 The Lag Operator . . . 15
  - 2.8.1 The lag operator as a polynomial . . . 15
  - 2.8.2 Stationarity and the lag operator . . . 16
- 2.9 Finally, the ARMA(p,q) process . . . 16
  - 2.9.1 Stationarity and invertibility of an ARMA(p,q) process . . . 17
  - 2.9.2 Moments of an ARMA(p,q) process . . . 18

2.9.3 ACF of an ARMA(p,q) process 18
2.10 Testing for time dependence 18
2.10.1 Hypothesis testing 19
2.10.2 Testing autocorrelations or regressions? 20

3 Problem 1: Modelling exchange rates 21

4 Problem 2: Estimating ARMA models 38

5 Problem 3: Identification of ARMA models 39

6 Forecasting 51
6.1 Forecasting with an AR(1) model 51
6.1.1 Forecast 51
6.1.2 Forecast error 52
6.1.3 Mean reversion 52
6.2 Forecasting with an AR(p) model 53
6.2.1 Forecast 53
6.2.2 Forecast error 53
6.3 Forecasting with a MA(1) model 53
6.3.1 Forecast 54
6.3.2 Forecast error 54
6.4 Forecasting with a MA(q) model 54
6.5 Forecast with an ARMA(p,q) model 54
6.5.1 Forecast 55
6.6 Confidence intervals for forecasts 55
6.6.1 Normally distributed errors 55
6.6.2 Bootstrapping 56

7 Problem 4: Cross-validation and bootstrap 58
7.1 Identification and estimation 58
7.2 Cross-validation 63
7.3 Bootstrapping 66

Chapter 1

Introduction

1.1 Motivation

This course will be dedicated to *time series analysis.* Informally, a *time series* is any type of data collected over time – or, more formally, it is the realization of a stochastic process indexed in time. We usually denote the time series as follows:

$$y_1, ..., y_T; \quad \{y_t\}_{i=1}^T; \quad \{y_t\}_t$$

Time series analysis is useful for a number of different applications:

- **Forecasting.**
  - Uni and multivariate models
  - **ARIMA** models: mean and confidence interval forecasting
  - **ARCH** models: variance forecasting – especially useful in finance for volatility and risk
- **Dynamics.** Evaluate the impact of one variable in another over time.
  - Multivariate models including VAR, ECM
  - Contemporaneous lagged structural relations

It is important to address a first and simple question. **Why time series are different from other data?** The answer is also simple but incredibly relevant: *time series observations are not serially independent!*

$$Y_t \not\perp Y_{t-j}$$

In fact, they don't even have to be identically distributed:

$$F_{Y_t} \neq F_{Y_{t-j}}$$

This means that the essential *iid* hypothesis for traditional Econometrics *does not hold.* This means that we'll have to make some adjustments to our methods. That is the task of time series analysis.

1.2 Statistics with dependence

Let's begin with a proper definition of a time series.

1.2.1 Definition of a time series

Suppose that we have a probability space $(\Omega, S, \mathbb{P})$. Ω is the sample space; S is the set of all events; $\mathbb{P}$ is a measure of probability $\mathbb{P} : S \to [0, 1]$. From this, we define a random variable $Y : \Omega \to \mathbb{R}$. A realization of this r.v. is denoted by $y = Y(\omega)$ with fixed ω.

From this, we can define multiple random variables in the same sample space, indexed by integers:

$$Y = \{..., Y_{t-2}, Y_{t-1}, Y_t, ...\}$$

This is equivalent to writing:

$$Y : \Omega x \mathbb{Z} \to \mathbb{R}$$

We now arrive at our formal definition of a time series: $\{Y_t, t \in \mathbb{Z}\}$ is a time-indexed stochastic process.

- $Y(\cdot, t) : \Omega \to \mathbb{R}$ is a r.v. for fixed t.
- $Y(\omega, \cdot) : \mathbb{Z} \to \mathbb{R}$ is a *sequence of real numbers* for a fixed ω. In other words, this represents the *observed time series.*
- For fixed t, ω, $Y(\omega, t) \in \mathbb{R}$.

1.2.2 Unconditional expectation

An important concept to make clear here is *unconditional expectation.* With fixed t,

$$\mathbb{E}(Y_t) = \int_{-\infty}^{\infty} x f_{Y_t}(x) dx$$

Note the Y_t subscript on the probability density function f_{Y_t}. This means that $\mathbb{E}(Y_t)$ is not calculated with the values assumed by Y_{t-1}, Y_{t+1}. This raises an important problem: *how would you be able to estimate* $\mathbb{E}(Y_t)$*?* Note that we only observe $Y_t = y_t$, i.e., one realization of the r.v.

1.2.3 Statistical dependence

For any random variables X, Y, we can define multiple measures of dependency:

- **Linear:** $Cov(X, Y) \equiv \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y)$
- **Quadratic:** $Cov(X^2, Y^2)$
- **General:** $Cov(f(X), g(Y))$. This is a measure of covariance between two general functional forms of X and Y.

With this general definition, we arrive at an equivalent definition for independent random variables:

- $F_{X,Y}(x, y) = F_X(x) * F_Y(y)$, i.e., joint pdf is equal to the product of the marginal pdfs.
- $Cov(f(X), g(Y)) = 0$ for every pair of bounded functions f, g.

From this, we now define the *autocovariance and autocorrelation functions.*

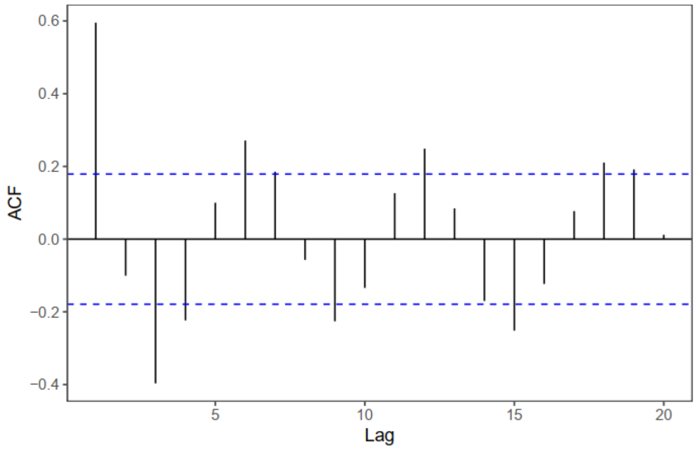
Definition 1.2.1. $\gamma_{j,t} := Cov(Y_t, Y_{t-j})$ *is the* ***autocovariance function*** *for a given time series* $\{Y_t, t \in \mathbb{Z}\}$.

Definition 1.2.2. $\rho_{j,t} := \dfrac{\gamma_{j,t}}{\sqrt{\gamma_{0,t}\gamma_{0,t-j}}}$ *is the* ***autocorrelation function*** *for a given time series* $\{Y_t, t \in \mathbb{Z}\}$.

Note that, if *iid* holds:

$$\gamma_{j,t} = \begin{cases} 0 & j \neq 0, \forall t \\ Var(Y) & otherwise \end{cases}$$

This is an example of an autocorrelation function.



1.3 Asymptotic theory with dependence

Some form of asymptotic theory is needed to enable *any kind of statistical analysis.* Namely, we need to have some form of Law of Large Numbers (LLN) and Central Limit Theorem (CLT) that are analogous to the *iid* environment. This will be achieved in our setting with some conditions called *stationarity* and *ergodicity.*

1.3.1 Stationarity

Definition 1.3.1. *A process* $\{Y_t, t \in \mathbb{Z}\}$ *is* ***strictly stationary*** *if, for all finite set of indexes* $\{t_1, ..., t_r\}$ *and for all* $m \in \mathbb{Z}$, $F(y_{t_1}, ..., y_{t_r}) = F(y_{t_1+m}, ..., y_{t_r+m})$ *holds, where* $F(y_{t_1}, ..., y_{t_r})$ *is the joint cdf of* $(Y_{t_1}, ..., Y_{t_r})$.

More informally, a given process is called *strictly stationary* if its statistical properties depend only on the *relative position* between observations, and not its *absolute position.*

We'll usually adopt a weaker definition of stationarity for our models. Henceforth, we will refer to stationarity in this sense.

Definition 1.3.2. *A process* $\{Y_t, t \in \mathbb{Z}\}$ *is* ***stationary*** *(or weakly stationary) if there exists* $\mu \in \mathbb{R}$ *and* $\{\gamma_j\}_{j \in \mathbb{N}}$ *such that:*

- $\mathbb{E}(Y_t) = \mu, \quad \forall t$
- $\mathbb{E}[(Y_t - \mu)(Y_{t-j} - \mu)] = \gamma_j, \quad \forall (t, j) \in \mathbb{N}^2$

Note that, from the second condition in the definition, we have $\mathbb{E}(Y_t - \mu)^2 = \gamma_0 \in \mathbb{R}, \forall t \in \mathbb{N}$. In other words, *the unconditional variance of the time series is constant.*

Some important remarks on stationarity:

- Stationarity does not imply strict stationarity
- Stricy stationarity does not imply stationarity
- Every strictly stationary process with finite variance is stationary
- Every iid process is strictly stationary
- Every strictly stationary process is identically distributed
- A stationary process is not necessarily identically distributed

1.3.2 Ergodicity

Stationarity is not enough to guarantee that we have even a Law of Large Numbers. To see why that is the case, consider the following example:

$$Y_t = X + \varepsilon_t, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2), \quad X \sim \mathcal{N}(0, 1), \quad X \perp\!\!\!\perp \varepsilon_t$$

Is this process stationary? No, because the sample *time average* $\bar{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$ does not converge to the population *ensemble average* $\mathbb{E}(Y_t) = \mu$.

We need some condition that guarantees that the dependence structure of the time series decays as the observation get further from each other. That is the intuition behind *ergodicity.*

Definition 1.3.3. *A strictly stationary process $\{Y_t, t \in \mathbb{Z}\}$ is called* ***ergodic*** *if*

$$\lim_{J \to \infty} \frac{1}{J} \sum_{j=1}^{J} Cov[f(X_1), g(X, j)] = 0,$$

for all pairs of bounded functions f, g.

This is a kind of mean asymptotic independence, in which the asymptotic independence would be defined by $Cov[f(X_1), g(X_J)] \to 0$ as $J \to \infty$.

Now, we can define a Law of Large Numbers – also called the *Ergodic Theorem.*

Theorem 1.3.1. *Given an ergodic stochastic process $\{Y_t, t \in \mathbb{Z}\}$ such that $\mathbb{E}|Y_1| < \infty$,*

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} Y_t = \mathbb{E}(Y_1) \quad almost\, sure$$

This theorem is the generalization of the strong LLN. However, it presupposes *strict stationarity,* which is a very strong assumption most of the time. Fortunately, this theorem gave rise to other definitions that arrive at our objective, namely, a LLN for the first two moments.

Definition 1.3.4. *A stationary process $\{Y_t, t \in \mathbb{Z}\}$ is said to be* ***ergodic for the mean*** *if*

$$\frac{1}{T} \sum_{t=1}^{T} Y_t \to_p \mathbb{E}(Y_t), \quad T \to \infty$$

Definition 1.3.5. *A stationary process* $\{Y_t, t \in \mathbb{Z}\}$ *is said to be* ***ergodic for the second moment*** *if, for every* j,

$$\frac{1}{T-j}\sum_{t=j+1}^{T} Y_t Y_{t-j} \to_p \mathbb{E}(Y_t), \quad T \to \infty$$

Proposition 1.3.2. $\sum_{j=0}^{\infty} |\gamma_j| < \infty$ *is a suficient condition for ergodicity for the mean.*

Proof. Let $Z_t := Y_t - \mu$ and $\bar{Z}_t := \frac{1}{T}\sum_{t=1}^{T} Z_t$, where $\{Y_t, t \in \mathbb{Z}\}$ is a stationary process. We will show that $\bar{Z}_t$ converges to 0 in mean square.

$$\begin{aligned}
\mathbb{E}\left(\bar{Z}_T^2\right) &= \mathbb{E}\left[\left(\frac{1}{T}\sum_{t=1}^{T} Z_t\right)\left(\frac{1}{T}\sum_{t=1}^{T} Z_t\right)\right] = \frac{1}{T^2}\mathbb{E}\left(\sum_{s=1}^{T}\sum_{t=1}^{T} Z_s Z_t\right) \\
&= \frac{1}{T^2}\sum_{s=1}^{T}\sum_{t=1}^{T}\mathbb{E}\left(Z_s Z_t\right) = \frac{1}{T^2}\sum_{s=1}^{T}\sum_{t=1}^{T}\gamma_{s-t} = \frac{1}{T}\sum_{j=-T+1}^{T-1}\frac{T-|j|}{T}\gamma_j \\
&\leq \frac{1}{T}\sum_{j=-T+1}^{T-1}\frac{T-|j|}{T}|\gamma_j| \leq \frac{1}{T}\sum_{j=-T+1}^{T-1}|\gamma_j| \to 0
\end{aligned}$$

□

1.3.3 A Central Limit Theorem for time series

The conditions that guarantee the existence of a CLT for stationary and ergodic processes are much more envolving than in the *iid* environment. However, we have a relatively simple result that will be useful to us in time series analysis. It will now be presented without proof.

Theorem 1.3.3. *Let* $\{Y_t, t \in \mathbb{Z}\}$ *be a* ***linear*** *stationary process, i.e., that can be written in the form* $Y_t = \mu + \sum_{j=0}^{\infty}\psi_j \epsilon_{t-j}$, *where* $\varepsilon \sim_{iid} (0, \sigma^2)$ *and* $\sum_{j=-\infty}^{\infty}|\psi_j| < \infty$. *Then,*

$$\sqrt{T}(\bar{Y}_t - \mu) \to_d \mathcal{N}(0, \omega^2),$$

where $\omega^2 := \sum_{j=-\infty}^{\infty}\gamma_j < \infty$

Chapter 2

ARMA Models

ARMA is a class of models that we'll employ frequently in time series analysis. Let's begin with some definitions.

2.1 White noise

We call *white noise* stationary time series with mean zero that do not have serial correlation.

Definition 2.1.1. *$\{Y_t, t \in \mathbb{Z}\}$ is* ***white noise****, denoted by $Y_t \sim wn(0, \sigma^2)$, if*

$$\mathbb{E}(Y_t) = 0; \quad \mathbb{E}(Y_t, Y_{t-j}) = \begin{cases} \sigma^2 & j = 0 \\ 0 & j \neq 0 \end{cases}$$

This is the most simple time series – except for the *iid* case, where independence also holds. It will be the building block for a number of processes that we will study.

2.2 Moving Average processes

Let's begin with the simplest form of MA processes: MA(1).

Definition 2.2.1. *A stationary process $\{Y_t, t \in \mathbb{Z}\}$ is called* ***MA(1)****, or a* ***moving average of order 1****, if it follows the following form:*

$$Y_t = c + \varepsilon_t + \theta\varepsilon_{t-1}, \quad \varepsilon_t \sim wn(0, \sigma^2),$$

where c, θ are constant.

2.2.1 Moments of an MA(1) model

The expected value of an MA(1) is:

$$\mu \equiv \mathbb{E}(Y_t) = \mathbb{E}(c + \varepsilon_t + \theta\varepsilon_{t-1}) = c$$

With this result, we can rewrite the model as:

$$(Y_t - \mu) = \varepsilon_t + \theta\varepsilon_{t-1}$$

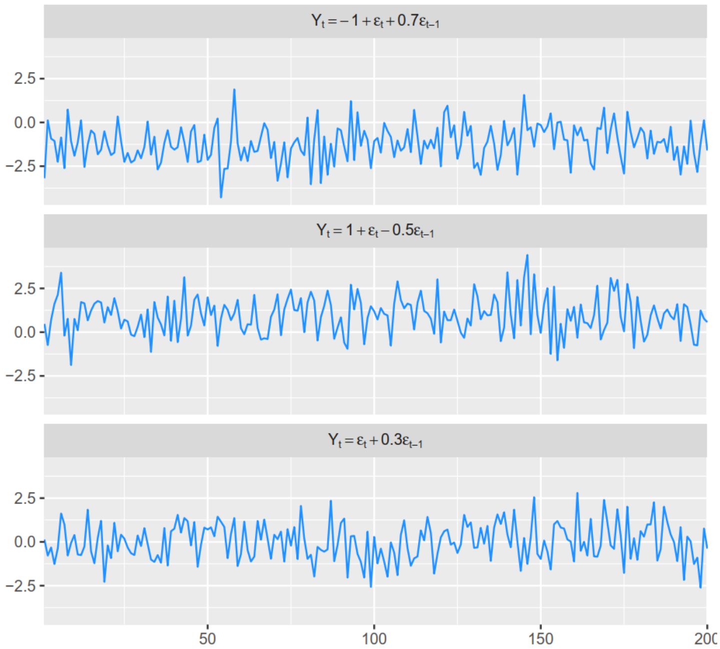
Multiplying both sides by $(Y_{t-j} - \mu)$ yields:

$$\begin{aligned}(Y_t - \mu)(Y_{t-j} - \mu) &= (\epsilon_t + \theta\epsilon_{t-1})(\epsilon_{t-j} + \theta\epsilon_{t-j-1}) \\ &= \epsilon_t\epsilon_{t-j} + \theta\epsilon_t\epsilon_{t-j-1} + \theta\epsilon_{t-1}\epsilon_{t-j} + \theta^2\epsilon_{t-1}\epsilon_{t-j-1}\end{aligned}$$

Applying the expected value operator to both sides, we have the autocovariances of the model.

$$\gamma_j \equiv \mathbb{E}\left[(Y_t - \mu)(Y_{t-j} - \mu)\right] = \begin{cases} (1+\theta^2)\,\sigma^2 & j = 0 \\ \theta\sigma^2 & j = \pm 1 \\ 0 & |j| > 1 \end{cases}$$

2.2.2 Some examples of MA(1) processes



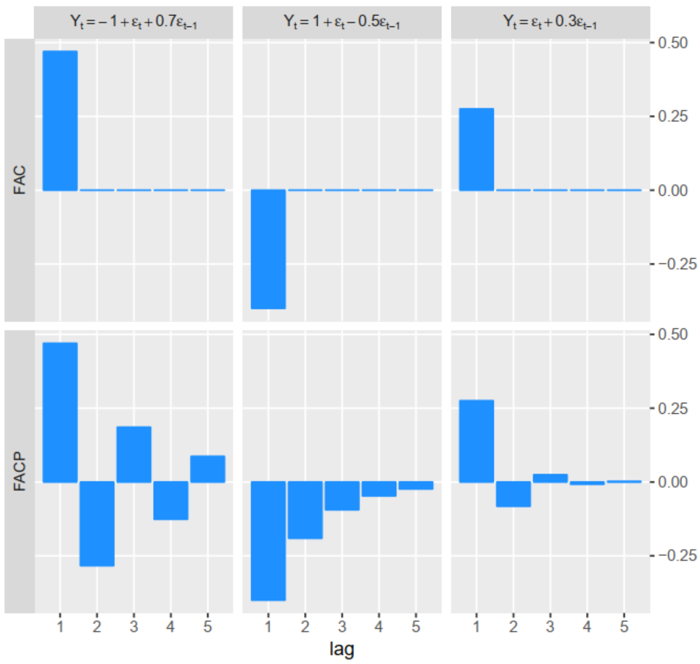
2.2.3 Deriving the Autocorrelation function of the MA(1) process

While deriving the moments of the MA(1), it became clear that the process is stationary and ergodic to the mean. Note that the time average $\bar{y}_t$ converges to $\mathbb{E}(Y_t)$, the *ensemble average*, the absolute sum of all covariances is clearly finite ($\gamma_j = 0, \forall j > 1$) and the dependence structure depends only on the relative positions of the observations.

Let's use the results of the autocovariances to construct the ACF:

$$\rho_j \equiv \frac{\gamma_j}{\gamma_0} = \begin{cases} 1 & j = 0 \\ \frac{\theta}{1+\theta^2} & j = \pm 1 \\ 0 & |j| > 1 \end{cases}$$

Note that the ACF of an MA(1) process is *truncated* in zero for lags greater than 1.



2.3 Generalizing the MA model

We can now generalize the MA(1) model for a moving average of order q.

Definition 2.3.1. *A stationary process* $\{Y_t, t \in \mathbb{Z}\}$ *is called* ***MA(q)****, or a* ***moving average of order q****, if it follows the following form:*

$$Y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}, \quad \varepsilon \sim wn(0, \sigma^2),$$

where $c, \theta_1, \ldots, \theta_q \in \mathbb{R}, q \in \mathbb{Z}^+$.

2.3.1 Moments of an MA(q) process

The expected value of an MA(q) is:

$$\mu \equiv \mathbb{E}\left(Y_t\right) = \mathbb{E}\left(c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}\right) = c$$

Again, using the first result, we can rewrite the model as:

$$\left(Y_t - \mu\right) = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

Multiplicando ambos os lados por $\left(Y_{t-j} - \mu\right)$, temos

$$\begin{aligned}
\left(Y_t - \mu\right)\left(Y_{t-j} - \mu\right) &= \left(\sum_{k=0}^{q} \theta_k \varepsilon_{t-k}\right)\left(\sum_{k=0}^{q} \theta_k \varepsilon_{t-j-k}\right) \\
&= \sum_{k=0}^{q} \sum_{\ell=0}^{q} \theta_k \theta_\ell \varepsilon_{t-k} \varepsilon_{t-j-\ell},
\end{aligned}$$

where $\theta_0 = 1$. Applying the expectation operator, we have:

$$\gamma_j = \begin{cases} \left(\theta_j + \theta_{j+1}\theta_1 + \theta_{j+2}\theta_2 + \cdots + \theta_q \theta_{q-j}\right)\sigma^2 & |j| = 0, 1, \ldots, q \\ 0 & |j| > q \end{cases}$$

2.3.2 Deriving the Autocorrelation function of the MA(q) process

Again, we can clearly see that the MA(q) model is *stationary* and *ergodic.* Note that the time average $\bar{y_t}$ converges to $\mathbb{E}(Y_t)$, the *ensemble average*, the absolute sum of all covariances is clearly finite ($\gamma_j = 0, \forall j > q$) and the dependence structure depends only on the relative positions of the observations.

The autocorrelation function is given by:

$$\rho_j \equiv \frac{\gamma_j}{\gamma_0} = \begin{cases} 1 & j = 0 \\ \frac{\theta_j + \theta_{j+1}\theta_1 + \theta_{j+2}\theta_2 + \cdots + \theta_q\theta_{q-j}}{1 + \theta_1^2 + \cdots + \theta_q^2} & |j| = 1, 2, \ldots, q \\ 0 & |j| > q \end{cases}$$

Now, the ACF is truncated in zero for lags *greater than q.*

2.4 The MA(∞) model

Consider a special case of a MA(q) model where $q \to \infty$. This yields a moving average of infinite order, MA(∞).

Definition 2.4.1. *A stationary process $\{Y_t, t \in \mathbb{Z}\}$ is called **MA(∞)**, or a **moving average of infinite order**, if it follows the following form:*

$$Y_t = c + \sum_{i=0}^{\infty} \theta_i \varepsilon_{t-i}, \quad \sim wn(0, \sigma^2),$$

where $c, \theta_1, ..., \theta_q \in \mathbb{R}, \theta_0 = 1$.

We also assume that $\sum_{i=0}^{\infty} |\theta_i| < \infty$. This guarantees that the process is *ergodic.*[1] With this assumption, we can obtain the moments of the MA(∞) simply by taking the limit of the finite case MA(q) – because it enables us to exchange the order between the sum and the expectation operator.

This means that $\mu = c$, as in the previous cases, and:

$$\gamma_j = \left(\sum_{i=0}^{\infty} \theta_{j+i}\theta_i \right) \sigma^2$$

2.5 The Wold Decomposition

This result motivates all ARMA models. It can be defined informally as "any stationary process has a MA(∞) representation".

Theorem 2.5.1. ***Wold Representation Theorem.*** *Any process $\{Y_t, t \in \mathbb{Z}\}$ purely nondeterministic can be written as*

$$Y_t = \mu + \sum_{j=0}^{\infty} \psi_j \varepsilon_{t-j},$$

where $\varepsilon_t = Y_t - \pi(Y_t|1, Y_{t-1}, Y_{t-2}, ...)$, i.e., ε_t is the error of the linear projection of Y_t in $(1, Y_{t-1}, Y_{t-2}, ...)$.

[1] Details in Hamilton (1994), Appendix, 3.A.

2.6 Autoregressive models

Again, we'll begin with its simplest form, AR(1).

Definition 2.6.1. *A stationary process $\{Y_t, t \in \mathbb{Z}\}$ is called **AR(1)**, or an **autoregressive process of order 1**, if it follows the following form:*

$$Y_t = c + \phi Y_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2),$$

where c, θ are constant.

2.6.1 Moments of an AR(1) process

With AR($\cdot$) models, we will work in the opposite direction when it comes to stationarity. We'll first *assume* that is holds, and then provide reasoning for why the assumption is valid.

With the assumption of stationarity, we can take expectations and variances on both sides:

$$\mu = c + \phi\mu \iff \mu = \mathbb{E}(Y_t) = \frac{c}{1-\phi}$$

$$\gamma_0 = \phi^2\gamma_0 + \sigma^2 \iff \gamma_0 = Var(Y_t) = \frac{\sigma^2}{1-\phi^2}$$

Using the first result, we can rewrite the model as:

$$(Y_t - \mu) = \phi(Y_{t-1} - \mu) + \varepsilon_t$$

Multiplying both sides by $(Y_{t-j} - \mu)$ and taking expectations yields:

$$\gamma_j = \phi\gamma_{t-j}, \quad = 1, 2, ...$$

2.6.2 Some examples of AR(1) series

2.6.3 Autocorrelation function of an AR(1) process

Given that $\gamma_j = \phi\gamma_{t-j}$, it is easy to see that the autocovariance is given by:

$$\gamma_j = \phi^{|j|}\gamma_0, \quad j \in \mathbb{Z}$$

Therefore, $\rho_j = \dfrac{\gamma_j}{\gamma_0} = \phi^{|j|}$.

2.6.4 Partial Autocorrelation Function

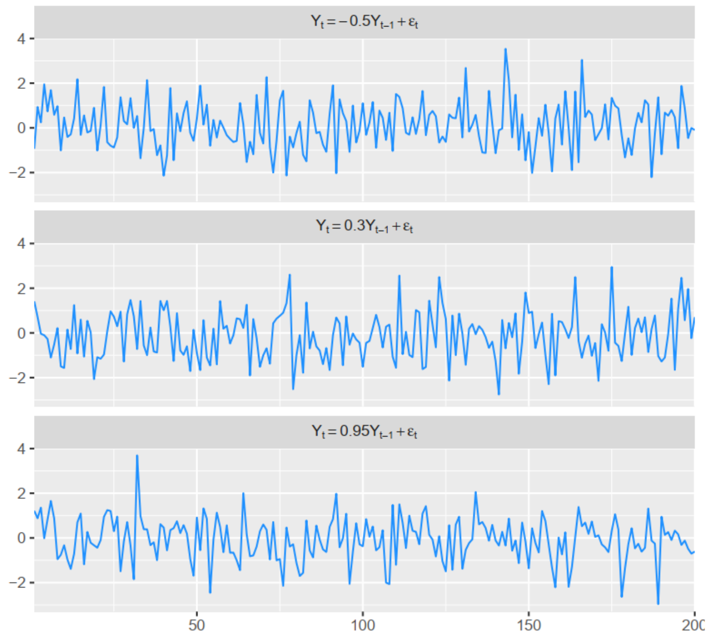
Note that Y_t, Y_{t-2} *are correlated.* Can we isolate the correlation between Y_t, Y_{t-2} from the effects of Y_{t-1}?

$$Cor(Y_t, Y_{t-2}|Y_{t-1}) = Cor(c + \phi Y_{t-1} + \varepsilon_t, Y_{t-2}|Y_{t-1}) = 0$$

This is the intuition behind the *partial autocorrelation function* (PACF).

Definition 2.6.2. *The **partial autocorrelation function** of a stationary process $\{Y_t, t \in \mathbb{Z}\}$ is given by:*

$$\alpha_j = \begin{cases} \text{Cor}\,(Y_t, Y_{t-1}) =: \rho_1 & j = 1 \\ \text{Cor}\,(Y_t, Y_{t-j} \mid Y_{t-1}, \dots, Y_{t-j+1}) & j \geq 2 \end{cases}$$



To estimate the ACF of a given time series, we need to use its sample equivalent and a version of the Law of Large Numbers, presented in the previous section, because we're only looking for correlations – i.e., population moments. To estimate the PACF, that is not enough. We're now looking for *partial correlation.*

It so happens that OLS gives us the *ceteris paribus* effects. Note that a general form for β is given by: $\beta = \frac{Cov(X,Y)}{Var(X)}$. Therefore, we can estimate using OLS the following models for every j:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \dots + \alpha_j Y_{t-j} + u_t$$

The last coefficient of *each regression*, $\hat{\alpha}_j$, is a consistent estimator for α_j. It is important to highlight, here, that a new model shall be estimated *for each* j, as it guarantees that the coefficient α_j will be conditional on all t *prior* to j.

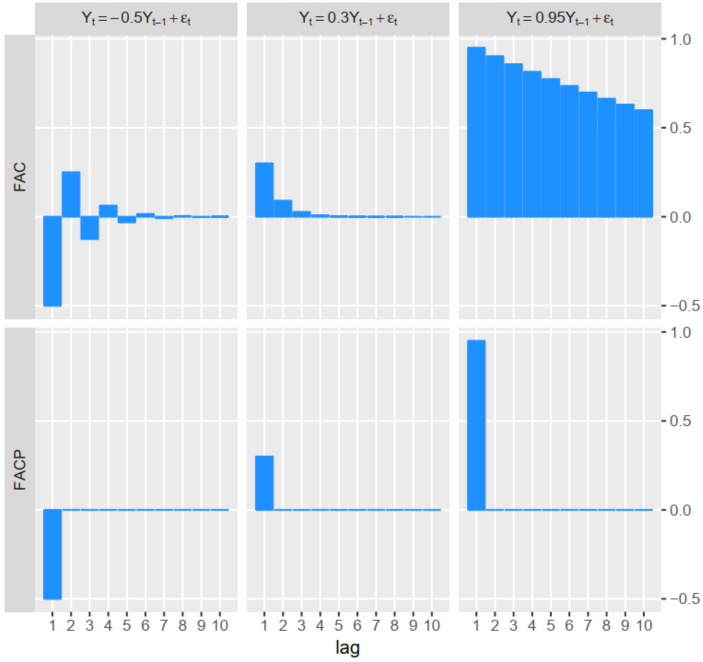
The following plots showcase ACFs and PACFs for AR(1) processes.

2.6.5 Conditions for stationarity

When is an AR(1) process stationary? Note that:

$$\begin{aligned}
Y_t &= c + \phi Y_{t-1} + \varepsilon_t \\
&= c + \phi\left(c + \phi Y_{t-2} + \varepsilon_{t-1}\right) + \varepsilon \\
&= c + \phi\left(c + \phi\left(c + \phi Y_{t-3} + \varepsilon_{t-2}\right) + \varepsilon_{t-1}\right) + \varepsilon_t \\
&\dots \\
&= c \sum_{j=0}^{k-1} \phi^j + \phi^k Y_{t-k} + \sum_{j=0}^{k-1} \phi^j \varepsilon_{t-j}
\end{aligned}$$

Assuming that $|\phi| < 1$ and taking the limit $k \to \infty$, we have:



$$Y_t = \frac{c}{1-\phi} + \sum_{j=0}^{\infty} \phi^j \varepsilon_{t-j}$$

The first term follows from the sum of an infinite geometric sequence. This means that *an AR(1) process can be written as a MA(∞)* with $\sum_{j=0}^{\infty} |\theta_j| < \infty$. Note that this is equivalent to saying that the Wold Representation Theorem holds, with $\mu = \frac{c}{1-\phi}$, $\psi_j = \phi^j$. This guarantees that the AR(1) process is *stationary* and *ergodic*.

2.7 Generalizing the AR model

Definition 2.7.1. *A stationary process $\{Y_t, t \in \mathbb{Z}\}$ is called* ***AR(p)****, or an* ***autoregressive process of order p****, if it follows the following form:*

$$Y_t = c + \phi Y_{t-1} + ... + \phi_p Y_{t-p} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2),$$

where $c, \theta_1, ..., \theta_p$ are constant.

2.7.1 Moments of an AR(p) process

Assuming stationarity, we can apply again expectations on both sides:

$$\mu = c + \phi_1 \mu + \cdots + \phi_p \mu \Longleftrightarrow \mu = \frac{c}{1 - \phi_1 - \cdots - \phi_p}$$

Using this result, we can rewrite the model as:

$$(Y_t - \mu) = \phi_1 (Y_{t-1} - \mu) + \cdots + \phi_p (Y_{t-p} - \mu) + \epsilon_t$$

Multiplying both sides by $(Y_{t-j} - \mu)$ and taking expectations, we have:

$$\gamma_j = \begin{cases} \phi_1 \gamma_{j-1} + \cdots + \phi_p \gamma_{j-p} & j = 1, 2, \ldots \\ \phi_1 \gamma_1 + \cdots + \phi_p \gamma_p + \sigma^2 & j = 0 \end{cases}$$

Note that the last term in γ_0 is implied by $\mathbb{E}\left(\epsilon_t\right)\left(Y_t - \mu\right) = \sigma^2$.

2.7.2 ACF of an AR(p) process

Dividing the previous result by γ_0 yields:

$$\rho_j = \phi_1 \rho_{j-1} + ... + \phi_p \rho_{j-p}$$

Evaluating at $j = 1, 2, ..., p-1$ and using $p_i = p_{-i}$, we have the following system of difference equations (aka. Yule-Walker Equations):

$$\begin{cases} \rho_1 = \phi_1 + \phi_2 \rho_1 + \cdots + \phi_p \rho_{p-1} \\ \rho_2 = \phi_1 \rho_1 + \phi_2 + \cdots + \phi_p \rho_{p-2} \\ \vdots \\ \rho_p = \phi_1 \rho_{p-1} + \phi_2 \rho_{p-2} + \cdots + \phi_p \end{cases}$$

To solve this, we need to find $\rho_1, \rho_2, ..., \rho_j$ as functions of $\phi_1, \phi_2, ..., \phi_j$. The first equation above implies that further correlations from lag j will decay exponentially[2]. This means that the ACF pattern of an AR(p) looks like the one from the simple AR(1) model.

2.8 The Lag Operator

Definition 2.8.1. *Given a process $\{Y_t, t \in \mathbb{Z}\}$, the* ***lag operator*** *is defined by:*

$$\begin{aligned} &LY_t := Y_{t-1} \\ &L^2 Y(t) := L\left(LY_t\right) = L\left(Y_{t-1}\right) = Y_{t-2} \\ &\vdots \\ &L^j Y(t) := L\left(L\left(L \ldots LY_t\right) = Y_{t-j}\right. \end{aligned}$$

The lag operator is also commutative with multiplication and distributive with regards to addition:

$$L(cY_t) = c(LY_t)$$

$$L(Y_t + X_t) = LY_t + LX_t$$

2.8.1 The lag operator as a polynomial

Note that we can use the lag operator as a *polynomial.* We can now rewrite an AR(p) with zero mean as:

$$(1 - \phi_1 L - \phi_2 L^2 - ... - \phi_p L^p) Y_t = \varepsilon_t$$

Note that the term multiplying Y_t *is a polynomial in L.* We denote this by:

$$(L) Y_t = \varepsilon_t$$

[2]Review this.

Analogously, we can rewrite an MA(q) process as:

$$
\begin{aligned}
Y_t &= \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \cdots + \theta_q\epsilon_{t-q} \\
&= \left(1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q\right)\epsilon_t \\
&\equiv \Theta_q(L)\epsilon_t
\end{aligned}
$$

We would also like to define an operator $(1-\phi L)^{-1}$ such that:

$$(1-\phi L)^{-1}(1-\phi L) = 1$$

$(1-\phi L)^{-1}$ is well defined when $|\phi| < 1$ and the following condition holds[3]:

$$(1-\phi L)^{-1} := 1 + \phi L + \phi^2 L^2 + \phi^3 L^3 + ...$$

From this, we can rewrite the AR(1) as a MA(∞) by multiplying the AR by $(1-\phi L)^{-1}$ on both sides:

$$Y_t = (1-\phi L)^{-1}\varepsilon_t$$

The $(1-\phi L)^{-1}$ operator will be very useful to translate models between AR and MA representations, aside from highlighting the conditions of stationarity for the process.

2.8.2 Stationarity and the lag operator

We can factor out the polynomial of an AR(p) process as:

$$1 - \phi_1 L - ... - \phi_p L^p = (1-\lambda_1 L)...(1-\lambda_p L),$$

where $\lambda_j = \frac{1}{a_j} \forall j = 1, ..., p$ and $a_1, ..., a_p$ are the p roots of a polynomial of p-th degree. This means that we can rewrite the AR(p) process as:

$$(1-\lambda_1 L)...(1-\lambda_p L) = \varepsilon_t$$

If $|\lambda_p| < 1$ (or, equivalently, $|a_j > 1)\ \forall j = 1, ..., p$, then *the inverse polynomial exists* and we can write the AR(p) process as a MA(∞) – which *we know to be stationary*:

$$
\begin{aligned}
Y_t &= (1-\lambda_1 L)^{-1} \ldots (1-\lambda_p L)^{-1}\epsilon_t \\
&=: \left(1 + \psi_1 L + \psi_2 L^2 + \ldots\right)\epsilon_t \\
&=: \Psi_\infty(L)\epsilon_t
\end{aligned}
$$

2.9 Finally, the ARMA(p,q) process

An ARMA(p,q) model is created by combining an AR(p) with a MA(q).

Definition 2.9.1. *A stationary process $\{Y_t, t \in \mathbb{Z}\}$ is called **ARMA(p,q)**, or an **autoregressive-moving average process of order (p,q)**, if it follows the following form:*

$$Y_t = c + \phi Y_{t-1} + ... + \phi_p Y_{t-p} + \theta_1\varepsilon_{t-1} + ... + \theta_q\varepsilon_{t-q} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2),$$

where $c, \theta_1, ..., \theta_p, \phi_1, ..., \phi_q$ are constant, $p, q \in \mathbb{Z}^+$.

[3] Hamilton (1994), p. 27-29.

Using the lag operator yields an alternate form for the ARMA(p,q) process:

$$(1 - \phi_1 L - \cdots - \phi_p L^p) Y_t = c + (1 + \theta_1 L + \cdots + \theta_q L^q) \epsilon_t$$
$$\Phi_p(L) Y_t = c + \Theta_q(L) \epsilon_t$$

2.9.1 Stationarity and invertibility of an ARMA(p,q) process

Stationarity depends only on the AR part of the process, because all MA($\cdot$) are stationary. It is sufficient to verifiy that *the roots of the polynomial $\Phi_p(L)$ are out of the unit circle:*

$$\Phi_p(L) = 1 - \phi_1 L - \phi_2 L^2 - ... - \phi_p L^p$$

Invertibility depends only on the MA part of the process, because it needs to be able to be rewritten as a linear combination of its past values plus the contemporaneous error term ε_t:

$$Y_t = \alpha + \sum_{s=1}^{\infty} \pi Y_{t-s} + \varepsilon_t$$

for some α and $\{\pi_j\}$.

Consider, for example, the case of $MA(1)$ with $\mu = 0$

$$y_t = \varepsilon_t + \theta \varepsilon_{t-1}$$

which can be rewritten as

$$\varepsilon_t = y_t - \theta \varepsilon_{t-1}$$

Repeated substitution of this relation for the lagged ε_{t-s} terms yields

$$\begin{aligned}
\varepsilon_t &= y_t - \theta (y_{t-1} - \theta \varepsilon_{t-2}) \\
&= y_t - \theta y_{t-1} + \theta^2 \varepsilon_{t-2} \\
&\ldots \\
&= y_t - \theta y_{t-1} + \ldots + (-\theta)^p y_{t-p} + (-\theta)^{p+1} \varepsilon_{t-p+1}
\end{aligned}$$

If $|\theta| < 1$, then the last term in this expression tends to zero in mean-square as $p \to \infty$, so that it make sense to write

$$\varepsilon_t = y_t + \sum_{s=1}^{\infty} (-\theta)^s y_{t-s}$$

Or

$$y_t = \varepsilon_t + \sum_{s=1}^{\infty} (-\theta)^s y_{t-s}$$

so $|\theta| < 1$ is the sufficient condition for a $MA(1)$ process to be invertible. (Powell, Conditions for Stationarity and Invertibility, UC Berkeley.)

In other words, because AR($\cdot$) models *with roots of the polynomial outside of the unit circle* are invertible, being able to write the MA(q) part of the process as an AR(∞) with the root condition is sufficient to guarantee invertibility.

2.9.2 Moments of an ARMA(p,q) process

If the process is stationary, $\Phi_p^{-1}(L)$ exists and we can rewrite ARMA (p,q) as MA(∞)

$$Y_t = \mu + \Psi_\infty(L)\epsilon_t$$

onde

$$\mu \equiv \frac{c}{\Phi(1)}; \quad \Phi(1) = 1 - \sum_{j=1}^{p}\phi_j; \quad \Psi_\infty(L) \equiv \Phi_p(L)^{-1}\Theta_q(L)$$

From the results derived for MA(q) we have for $q=\infty$

$$\mathbb{E}(Y_t) = \mu$$
$$\gamma_j = \left(\sum_{i=0}^{\infty}\psi_{j+i}\psi_j\right)\sigma^2$$

where $\psi_0 = 1$

2.9.3 ACF of an ARMA(p,q) process

It is usually easy to identify an AR(p) or MA(q) visually by inspecting its ACF and PACF, because AR's PACF is truncated on p, MA's ACF is truncated on q. For ARMA(p,q) models it is more complicated: both functions are not truncated! Note, however, that in that case, the ACF decays geometrically after lag q and the PACF decays geometrically after lag p.

Model	**ACF**	**PACF**
AR(p)	Decays	Truncated after lag p
MA(q)	Truncated after lag q	Decays
ARMA(p,q)	Decays after lag q	Decays after lag p

2.10 Testing for time dependence

We've seen that a sufficient condition for ergodicity is convergence of the absolute sum of all covariances. This presents a problem: how can we *estimate* these covariances?

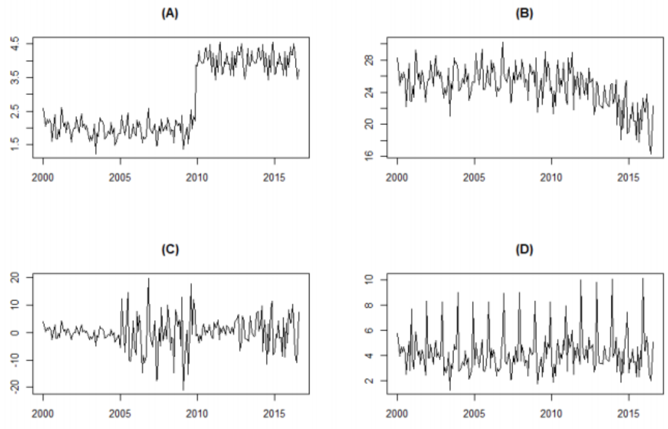
Let Z_t be our series to be tested. Denote the autocovariance of order j as $\gamma_j := Cov(Z_t, Z_{t-j})$. We can try to estimate these parameters with its sample equivalents:

$$\bar{z}_t := \frac{1}{T}\sum_{t=1}^{T} z_t$$

$$\hat{\gamma}_j := \frac{1}{T-j-1}\sum_{t=j+1}^{T}(z_t - \bar{z}_t)(z_{t-j} - \bar{z}_t)$$

But this is not as simple as it seems. We know that $\hat{\gamma}_j$ converges almost sure to γ_j *if the process is ergodic.* If it *isn't,* the information from $\hat{\gamma}_j$ may not be reliable – after all, we won't have a Law of Large Numbers!

Our solution to this problem won't be very rigorous here. We'll plot $\{\hat{\gamma}_j, j \in \mathbb{N}\}$ and check if it looks stationary. If the series passes this intuitive test, we can assume that $\{\hat{\gamma}_j, j \in \mathbb{N}\}$ will be informative about $\{\gamma_j, j \in \mathbb{N}\}$.

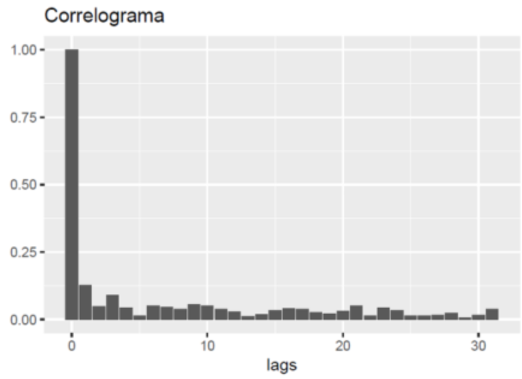


The visual inspection should focus on two main factors: (i) constant mean over time; (ii) constant variance over time. Here are some examples of series to be inspected:

After plotting the time series and assuring that it is well behaved, we can plot its *correlogram:* $\{\hat{\rho}_j := \frac{\hat{\gamma}_j}{\hat{\gamma}_0}, j \in \mathbb{N}\}$. If its sum looks convergent, we will assume that the process is *stationary* and *ergodic* – which will enable us to use sample equivalents as representations of population parameters.

2.10.1 Hypothesis testing

Consider this correlogram:



This series appears to not be correlated with its past. How can we test this?

$$H_0 = \rho_j = 0, \forall j \neq 0$$

This implies, in theory, that we would need to test infinite correlations. In practice, we limit the range to an arbitrary J. Let $\hat{\rho} := (\hat{\rho}_1, \hat{\rho}_2, ..., \hat{\rho}_J)^T, \rho = (\rho_1, \rho_2, ..., \rho_J)^T$. Under the null, $\rho = 0$, and as $T \to \infty$:

$$\sqrt{T}\hat{\rho} \to \mathcal{N}(0, I_J)$$

The intuition here is that, under H_0, $\hat{\rho}$ is a sequence of *iid* variables with mean zero and variance-covariance matrix I_J – which makes the CLT valid.

Given this result, we can now create a statistic that does not depend on the multivariate normal distribution. We will square and sum the expression to arrive at a Chi-squared distribution. This enables us to test the hypothesis with a Wald statistic. Under the null:

$$W_T = T\hat{\rho}^T\hat{\rho} = T\sum_{j=1}^{J}\hat{\rho}_j^{\,2} \to \chi_J^2$$

2.10.2 Testing autocorrelations or regressions?

Note that inferring about the autocorrelations is intimately related to inferring in a regression of a time series on its past values. This can be understood by remembering the linear projection interpretation of OLS. Ordinary Least Squares estimation *always* reports the parameters of the linear projection of Y in X, no matter how the model is specified!

Consider the following model:

$$\begin{aligned} Y_t &= \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_J Y_{t_J} + \varepsilon_t \\ &= \alpha + X_t\beta + \varepsilon_t \end{aligned}$$

where $\beta := (\beta_1, \ldots, \beta_J)^T$ and $X_t = (Y_{t-1}, \ldots, Y_{t-j})$. If we define the coefficients of the model above as the parameters of the linear projection of Y_t on the unit vector and X_t, $\alpha = \mu_Y - \mu_X\beta$ where $\mu_Y = \mathrm{E}(Y_t)$ and $\mu_X = \mathrm{E}(X_t)$

Using this result, we have:

$$Y_t - \mu_Y = (X_t - \mu_X)\beta + \varepsilon_t$$

This means that β can be written as:

$$\beta = \mathbb{E}\left[(X_t - \mu_X)^T(X_t - \mu_X)\right]^{-1}\mathbb{E}\left[(X_t - \mu_X)^T(Y_t - \mu_Y)\right] = \Gamma^{-1}\gamma$$

where the matrix Γ^{-1} is symmetric with diagonal elements all equal to $\gamma_1, \gamma_2, \ldots, \gamma_{J-1}$, due to the assumed stationarity. Note that $\mathbb{E}(Y_t - \mu_Y)^2 = \mathbb{E}(Y_{t-j} - \mu_{Y-j})^2 = \gamma_0 \forall j$.

Thus, $\beta = \overrightarrow{0} \iff \gamma = \overrightarrow{0}$, because Γ is a positive definite matrix. This means that *testing* $\beta = 0$ *is equivalent to testing* $\gamma = 0$.

It is important to highlight that this analysis is based upon the inference of $\gamma_j = \mathbb{E}(z_t - \bar{z}_T)(z_{t-j} - \bar{z}_T)$. If we were interested in *other types of relations between* Z_t *and its past, the analysis would have to be adapted* – for example, Z_t^2. It would be necessary to check again for stationarity and ergodicity.

Chapter 3

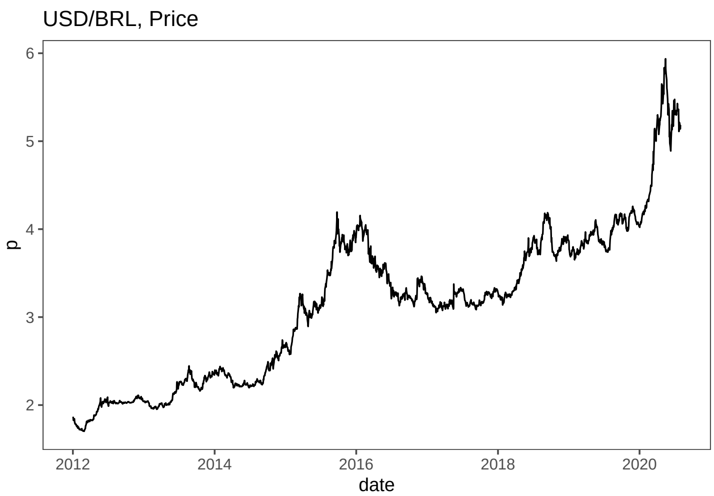
Problem 1: Modelling exchange rates

Loading the database and creating dummy variables:

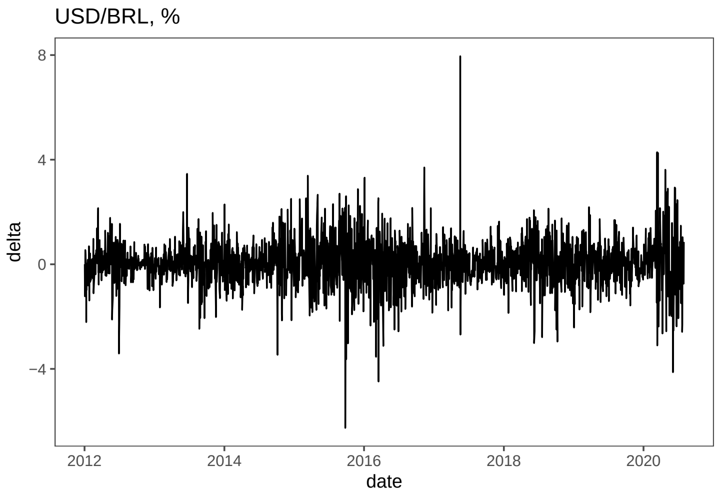
```
df <- read_excel("RS_USD.xlsx")


names(df)[names(df) == "R$/US$"] <- "p"

names(df)[names(df) == "Variação (em %)"] <- "delta"

names(df)[names(df) == "Data"] <- "date"

sign <- as.numeric(df$delta > 0)

count <- c(1:2153)

df <- data.frame(count, df, sign)
```

Before constructing our models, we need to check (intuitively) if the series at hand is *stationary* and *ergodic*. For this, we're going to plot the time series, its autocorrelations and partial autocorrelations.

```
pplot <- ggplot(data = df, aes(x = date, y = p)) + geom_line() +
    ggtitle("USD/BRL, Price") + theme_few()
pplot
```

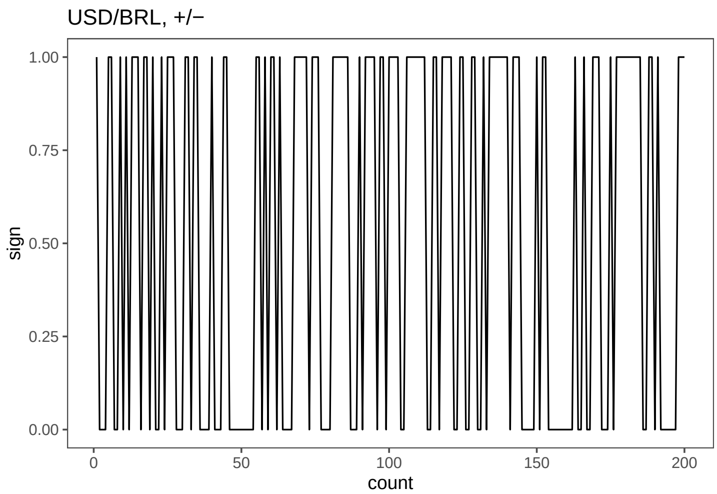


```
deltaplot <- ggplot(data = df, aes(x = date, y = delta)) + geom_line() +
    ggtitle("USD/BRL, %") + theme_few()
deltaplot
```

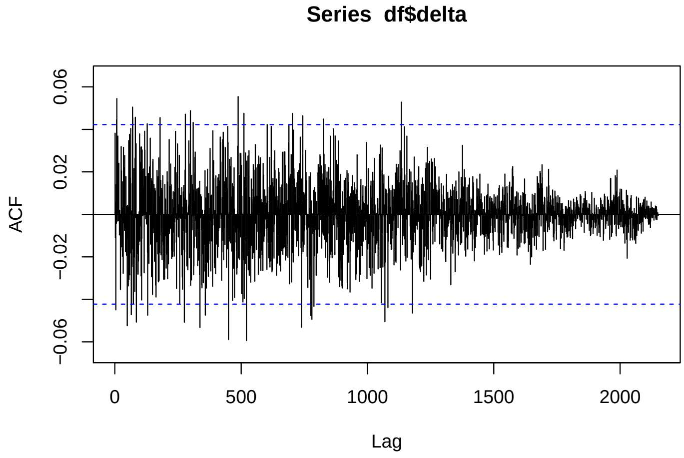



```
dummyplot <- ggplot(data = df, aes(x = count, y = sign)) + geom_line() +
    ggtitle("USD/BRL, +/-") + xlim(1, 200) + theme_few()
dummyplot
```

```
## Warning: Removed 1953 row(s) containing missing values (geom_path).
```




```
# For delta

acf_delta <- Acf(df$delta, lag.max = 5000)
```




```
acf_test_values <- acf_delta$acf/sd(acf_delta$acf)

head(data.frame(acf_test_values))
```

```
##   acf_test_values
## 1      37.9547672
## 2       1.4506537
## 3      -0.4173129
## 4       0.2125873
## 5      -1.7053782
## 6       0.5358210
```

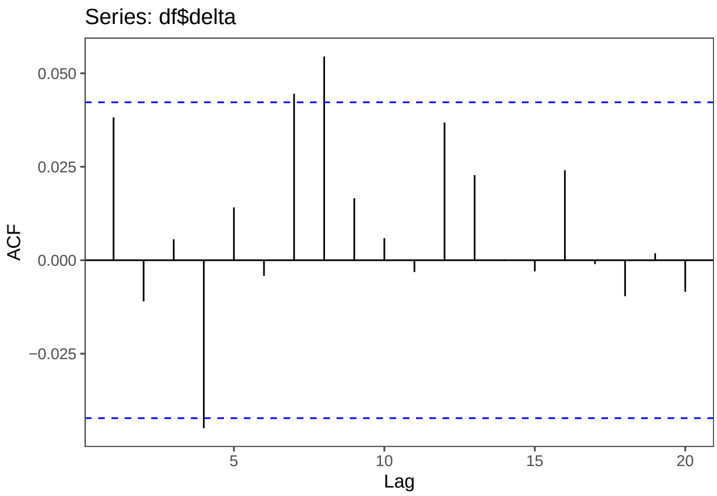
```
facst <- ggAcf(df$delta, type = "correlation", lag.max = 20,
    plot = T) + theme_few()
faclt <- ggAcf(df$delta, type = "correlation", lag.max = 5000,
    plot = T) + theme_few()

facpst <- ggPacf(df$delta, type = "correlation", lag.max = 100,
    plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```

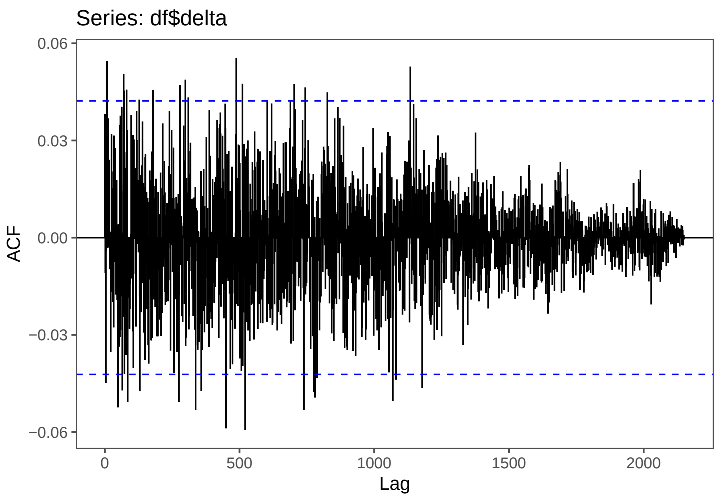
```
facplt <- ggPacf(df$delta, type = "correlation", lag.max = 5000,
    plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```

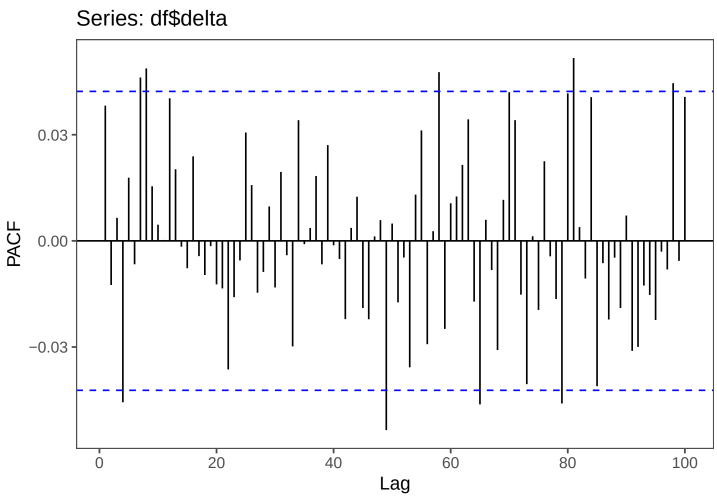
```
facst
```



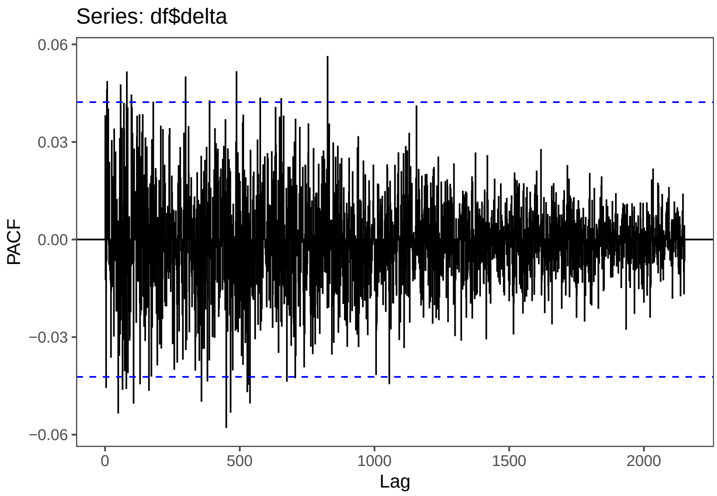

```
faclt
```



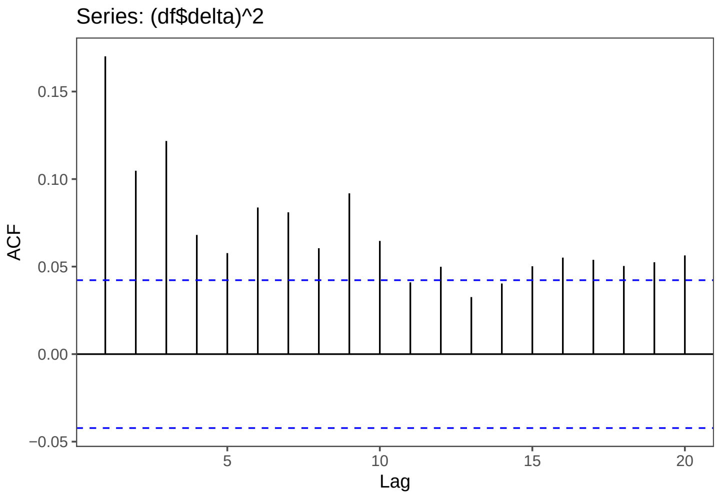

```
facpst
```



```
facplt
```




```
facst2 <- ggAcf((df$delta)^2, type = "correlation", lag.max = 20,
    plot = T) + theme_few()
facst2
```



Let's now create our first ARMA models (equivalent to ARIMA with 2nd argument = 0). We'll begin with the first hypothesis: $\mathbb{P}(+) = \mathbb{P}(-)$. Modelling this with an AR(1), we have:

$$Sign_{t+1} = \alpha + \beta Sign_t + \varepsilon, \qquad \varepsilon \sim wn(0, \sigma^2)$$

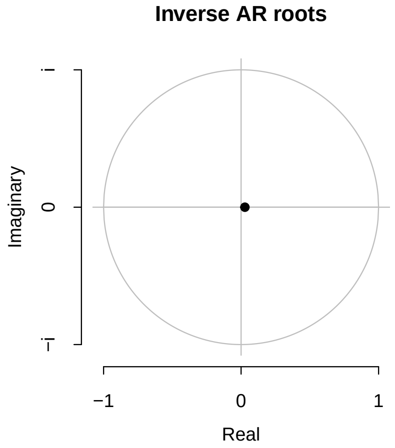
In R, we'll use the package *forecast* to construct this model:

```
AR1sign <- Arima(df$sign, order = c(1, 0, 0))
summary(AR1sign)
```

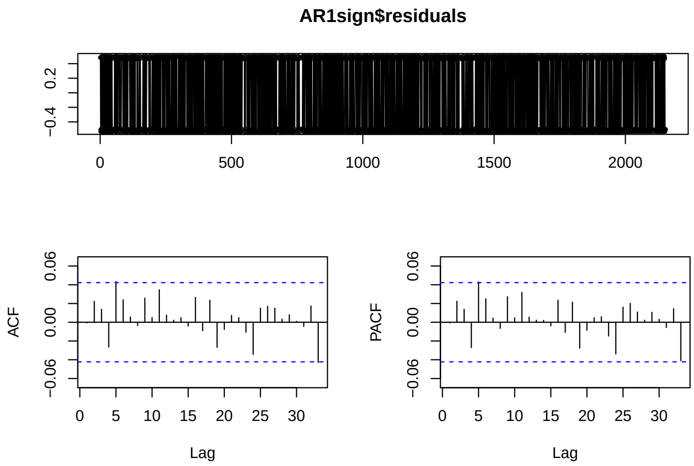
```
## Series: df$sign
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.0278  0.5165
```

```
## s.e.   0.0215   0.0111
##
## sigma^2 estimated as 0.2498:  log likelihood=-1560.63
## AIC=3127.26   AICc=3127.27   BIC=3144.28
##
## Training set error measures:
##                        ME      RMSE       MAE  MPE MAPE     MASE          ACF1
## Training set 2.157119e-05 0.4995356 0.4990724 -Inf  Inf 1.027755 -0.0006313563
```

```
plot(AR1sign)
```




```
tsdisplay(AR1sign$residuals)
```



With the results of the summary, we can now apply a hypothesis test for our first question.[1]

$$H_0 : \beta = 0$$

$$H_1 : \beta \neq 0$$

[1]Testing β is equivalent to testing γ.

$\frac{\hat{ar}_1 - ar_1}{s.e.(ar_1)}$:

```
AR1sign$coef[1]/sqrt(AR1sign$var.coef[1, 1])
```

```
##      ar1
## 1.287942
```

The second hypothesis in the problem refers to the delta of the variation:

$$\mathbb{E}(\Delta|+) \neq \mathbb{E}(\Delta|-).$$

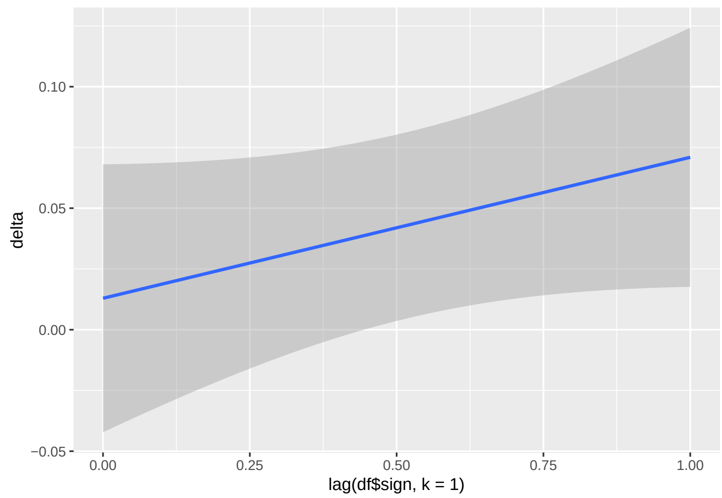
$$\Delta_{t+1} = \alpha + \beta Sign_t + \varepsilon, \qquad \varepsilon \sim wn(0, \sigma^2).$$

```
lmsignt <- lm(delta ~ lag(df$sign, k = 1), data = df)
summary(lmsignt)
```

```
##
## Call:
## lm(formula = delta ~ lag(df$sign, k = 1), data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3285 -0.4706 -0.0060  0.4655  7.9442
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          0.01293    0.02811   0.460    0.646
## lag(df$sign, k = 1)  0.05802    0.03911   1.484    0.138
##
## Residual standard error: 0.9066 on 2150 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.001023,   Adjusted R-squared:  0.000558
## F-statistic: 2.201 on 1 and 2150 DF,  p-value: 0.1381
```

```
ggplot(df, aes(x = lag(df$sign, k = 1), y = delta)) + geom_smooth(method = "lm")
```

```
## Warning: Use of `df$sign` is discouraged. Use `sign` instead.

## `geom_smooth()` using formula 'y ~ x'

## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```



$$\Delta_{t+1} = \alpha + \beta_1 \Delta_t + \beta_2 Sign_t + \varepsilon, \qquad \varepsilon \sim wn(0, \sigma^2)$$

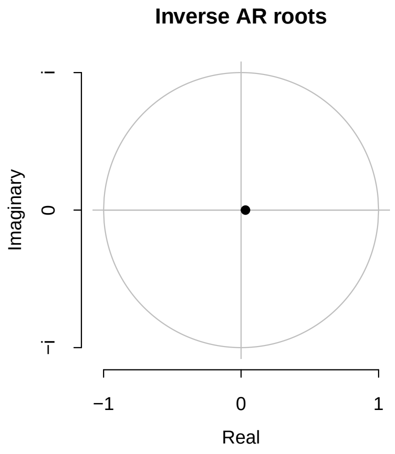
```
AR1delta <- Arima(df$delta, order = c(1, 0, 0), xreg = lag(df$sign,
    k = 1))
summary(AR1delta)
```

```
## Series: df$delta
## Regression with ARIMA(1,0,0) errors
##
## Coefficients:
##          ar1  intercept    xreg
##       0.0321     0.0343  0.0166
## s.e.  0.0306     0.0351  0.0556
##
## sigma^2 estimated as 0.8219:  log likelihood=-2840.96
## AIC=5689.93   AICc=5689.94   BIC=5712.62
##
## Training set error measures:
##                        ME      RMSE      MAE MPE MAPE      MASE         ACF1
## Training set 1.147232e-05 0.9059341 0.643417 NaN  Inf 0.7252668 0.0003998294
```

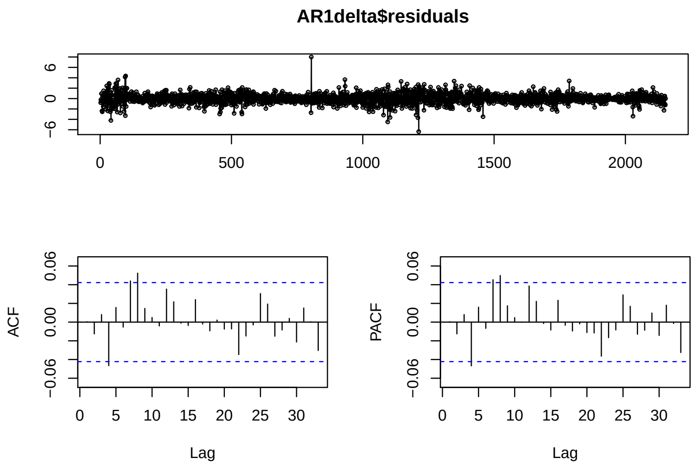
```
AR1delta$coef[1]/sqrt(AR1delta$var.coef[1, 1])
```

```
##     ar1
## 1.04747
```

```
plot(AR1delta)
```




```r
tsdisplay(AR1delta$residuals)
```



The last hypothesis in the problem refers to the variance:

$$\mathbb{E}(\Delta_{t+1}^2|\Delta_t).$$

$$\Delta_{t+1}^2 = \alpha + \beta\Delta_t^2 + \varepsilon, \qquad \varepsilon \sim wn(0, \sigma^2)$$

```r
AR1var <- Arima((df$delta)^2, order = c(1, 0, 0))
summary(AR1var)
```

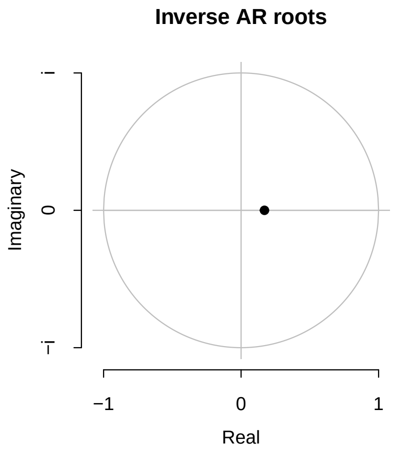
```
## Series: (df$delta)^2
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.1701  0.8238
## s.e.  0.0212  0.0582
##
## sigma^2 estimated as 5.026:  log likelihood=-4792.09
```

```
## AIC=9590.17   AICc=9590.18   BIC=9607.2
##
## Training set error measures:
##                         ME     RMSE       MAE  MPE MAPE      MASE        ACF1
## Training set -6.299133e-05 2.240784 0.9197335 -Inf  Inf 0.8595655 -0.01320252
```

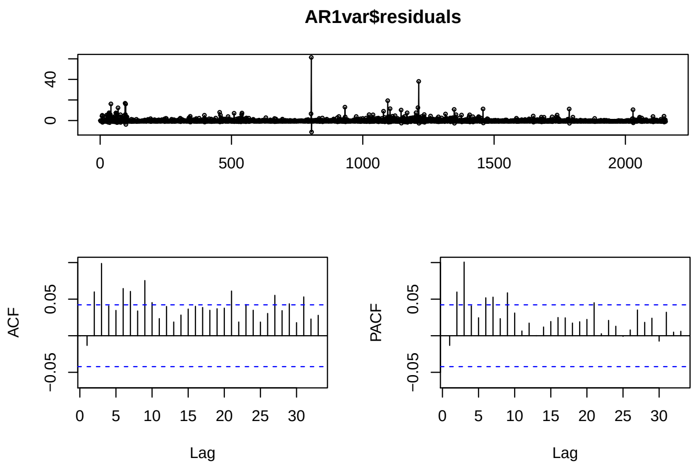
```r
AR1var$coef[1]/sqrt(AR1var$var.coef[1, 1])
```

```
##      ar1
## 8.011092
```

```r
plot(AR1var)
```




```r
tsdisplay(AR1var$residuals)
```



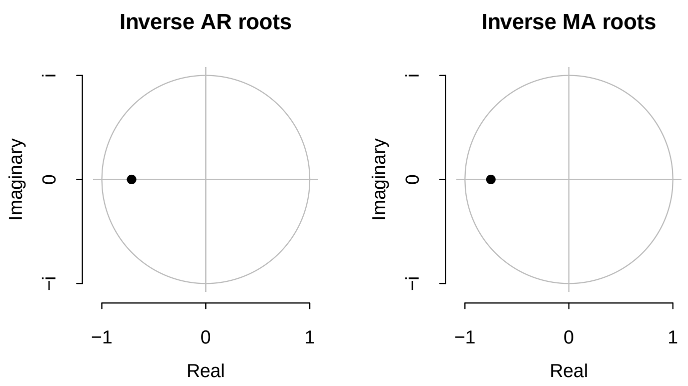
Now, let's run *auto.arima*.

```r
aadelta <- auto.arima(df$delta, stepwise = F)
summary(aadelta)
```

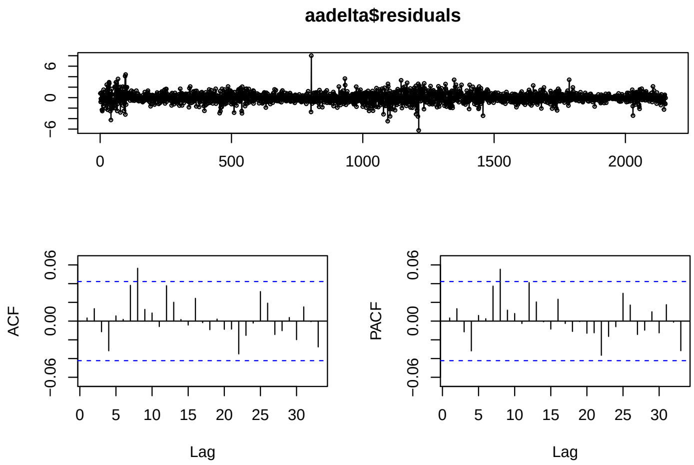
```
## Series: df$delta
## ARIMA(1,0,1) with non-zero mean
##
```

```
## Coefficients:
##           ar1     ma1    mean
##       -0.7138  0.7506  0.0433
## s.e.   0.1486  0.1399  0.0199
##
## sigma^2 estimated as 0.8208:  log likelihood=-2840.87
## AIC=5689.74   AICc=5689.76   BIC=5712.44
##
## Training set error measures:
##                        ME      RMSE       MAE MPE MAPE    MASE         ACF1
## Training set 2.610156e-05 0.9053381 0.6434199 NaN  Inf 0.72527 0.003320553
```

```r
plot(aadelta)
```




```r
tsdisplay(aadelta$residuals)
```



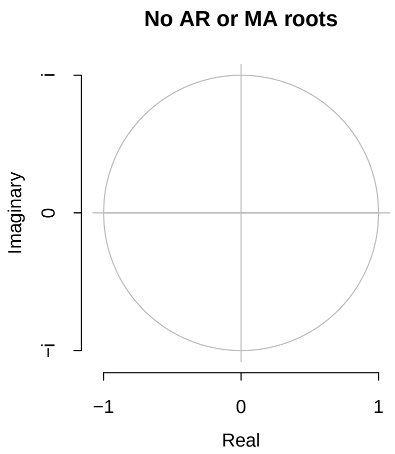

```r
aasign <- auto.arima(df$sign, stepwise = F)
summary(aasign)
```

```
## Series: df$sign
## ARIMA(0,0,0) with non-zero mean
```

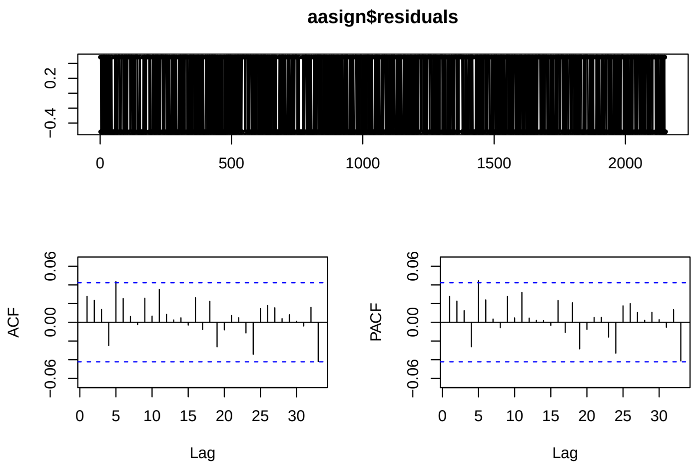
```
##
## Coefficients:
##         mean
##       0.5165
## s.e.  0.0108
##
## sigma^2 estimated as 0.2498:  log likelihood=-1561.46
## AIC=3126.91   AICc=3126.92   BIC=3138.26
##
## Training set error measures:
##                         ME      RMSE       MAE  MPE MAPE     MASE       ACF1
## Training set -2.382602e-13 0.4997281 0.4994563 -Inf  Inf 1.028545 0.02773919
```

```r
plot(aasign)
```

```
## Warning in plot.Arima(aasign): No roots to plot
```



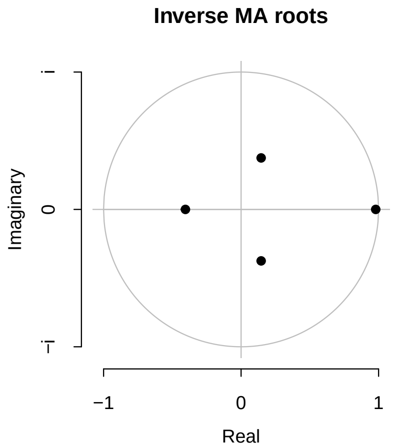

```r
tsdisplay(aasign$residuals)
```



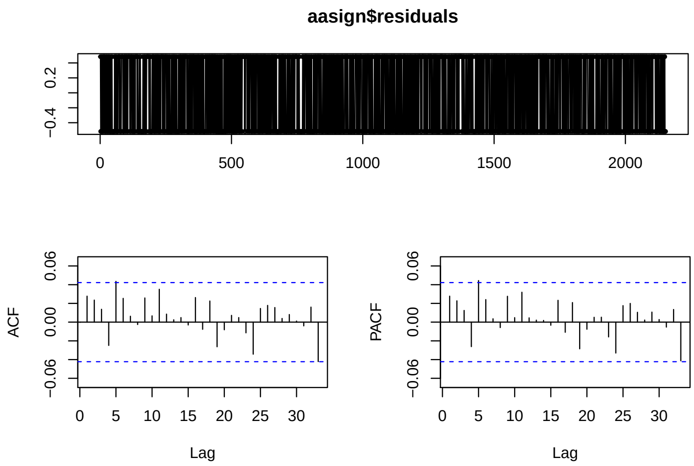

```r
aavar <- auto.arima((df$delta)^2, stepwise = F)
summary(aavar)
```

```
## Series: (df$delta)^2
## ARIMA(0,1,4)
##
## Coefficients:
##           ma1      ma2     ma3      ma4
##       -0.8662  -0.0671  0.0228  -0.0641
## s.e.   0.0215   0.0284  0.0280   0.0214
##
## sigma^2 estimated as 4.892:  log likelihood=-4761.22
## AIC=9532.45   AICc=9532.48   BIC=9560.82
##
## Training set error measures:
##                       ME     RMSE       MAE  MPE MAPE      MASE         ACF1
## Training set -0.02170361 2.209213 0.8904046 -Inf  Inf 0.8321553 0.0001189823
```

```
plot(aavar)
```



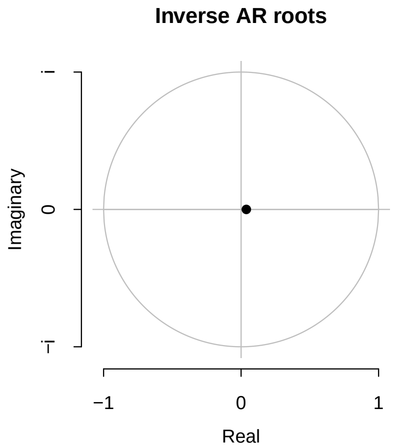

```
tsdisplay(aasign$residuals)
```



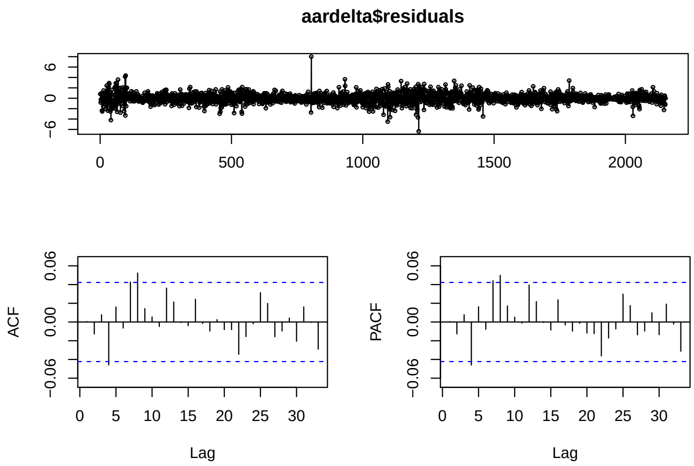

```
aardelta <- auto.arima(df$delta, max.q = 0, stepwise = F)
summary(aardelta)
```

```
## Series: df$delta
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.0382  0.0433
## s.e.  0.0215  0.0203
##
## sigma^2 estimated as 0.8215:  log likelihood=-2842.26
## AIC=5690.52   AICc=5690.53   BIC=5707.54
##
## Training set error measures:
##                         ME      RMSE       MAE MPE MAPE      MASE         ACF1
## Training set -1.464203e-05 0.9059233 0.6434299 NaN  Inf 0.7252813 0.0004805619
```

```r
plot(aardelta)
```




```r
tsdisplay(aardelta$residuals)
```

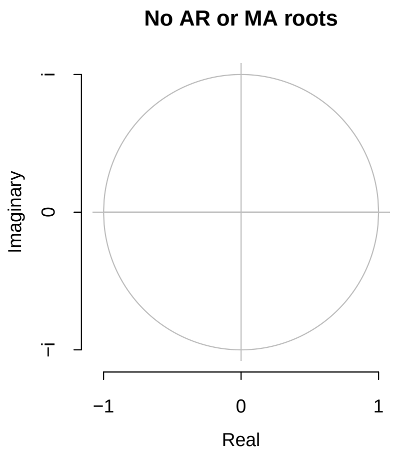



```r
aarsign <- auto.arima(df$sign, max.q = 0, stepwise = F)
summary(aarsign)
```

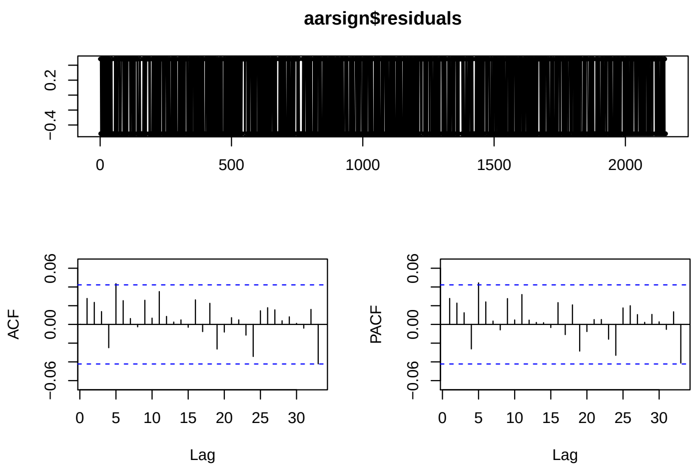
```
## Series: df$sign
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##         mean
##       0.5165
## s.e.  0.0108
##
## sigma^2 estimated as 0.2498:  log likelihood=-1561.46
## AIC=3126.91   AICc=3126.92   BIC=3138.26
##
## Training set error measures:
##                         ME      RMSE       MAE  MPE MAPE     MASE       ACF1
## Training set -2.382602e-13 0.4997281 0.4994563 -Inf  Inf 1.028545 0.02773919
```

```
plot(aarsign)
```

```
## Warning in plot.Arima(aarsign): No roots to plot
```



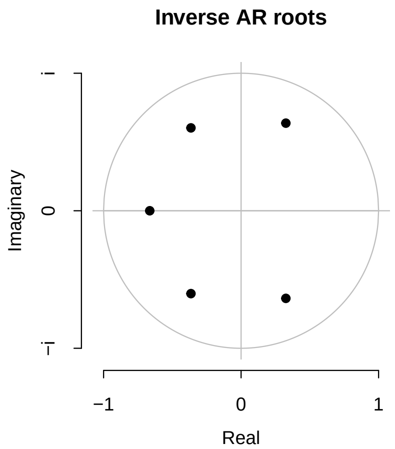

```
tsdisplay(aarsign$residuals)
```



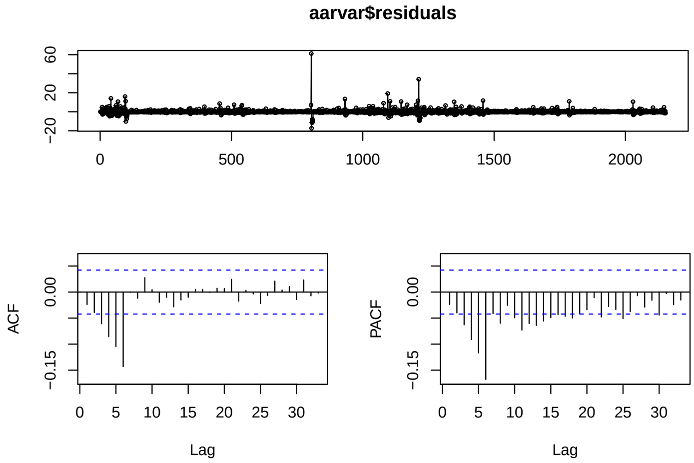
```r
aarvar <- auto.arima((df$delta)^2, max.q = 0, stepwise = F)
summary(aarvar)
```

```
## Series: (df$delta)^2
## ARIMA(5,1,0)
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5
##       -0.7420  -0.5845  -0.4042  -0.2873  -0.1687
## s.e.   0.0213   0.0259   0.0274   0.0258   0.0212
##
## sigma^2 estimated as 5.465:  log likelihood=-4878.95
## AIC=9769.89   AICc=9769.93   BIC=9803.94
##
## Training set error measures:
##                         ME     RMSE       MAE  MPE MAPE      MASE        ACF1
## Training set -3.234587e-05 2.334504 0.9170278 -Inf  Inf 0.8570369 -0.0239684
```

```r
plot(aarvar)
```




```r
tsdisplay(aarvar$residuals)
```



$$\Delta_{t+1} = c + \beta \Delta_t + \varepsilon$$

```
AR1_2 <- Arima(df$delta, order = c(1, 0, 0))

summary(AR1_2)
```

```
## Series: df$delta
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1    mean
##       0.0382  0.0433
## s.e.  0.0215  0.0203
##
## sigma^2 estimated as 0.8215:  log likelihood=-2842.26
## AIC=5690.52   AICc=5690.53   BIC=5707.54
##
## Training set error measures:
##                         ME      RMSE       MAE MPE MAPE      MASE         ACF1
## Training set -1.464203e-05 0.9059233 0.6434299 NaN  Inf 0.7252813 0.0004805619
```

```
confint(AR1_2, level = 0.95)
```

```
##                  2.5 %     97.5 %
## ar1       -0.003988796 0.08042284
## intercept  0.003511958 0.08308440
```

Chapter 4

Problem 2: Estimating ARMA models

{HAMILTON 5.1-5.6}

Chapter 5

Problem 3: Identification of ARMA models

In this problem, we'll be tackling the issue of *identification* of an ARMA model. Namely, we will employ the *Box-Jenkins* model selection strategy, based upon the concept of *parsimony*.

The principle of *parsimony* is inspired on the trade-off between *fit*, i.e., R^2, and *degrees of freedom*. "Box and Jenkins argue that parsimonious models produce better forecasts than overparametrized models". (p. 76)

The Box-Jenkins strategy is divided in three main stages:

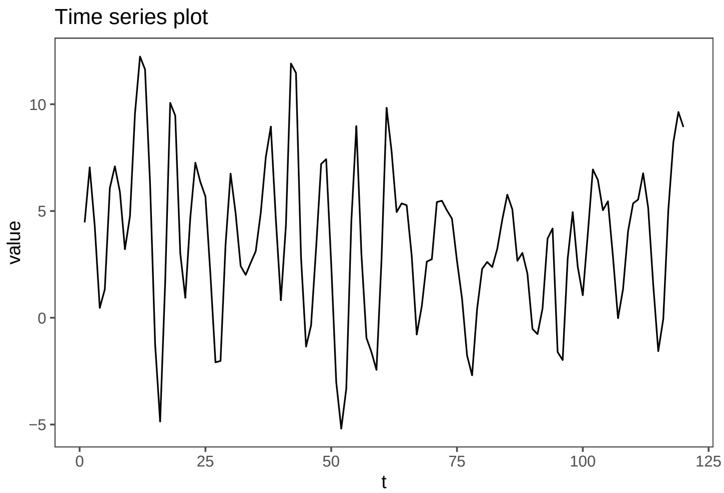
- Identification;
- Estimation;
- Diagnostic checking.

These estimations depend upon two essential conditions (discussed in earlier problems and lectures): *stationarity* and *invertibility*. Stationarity, as we have discussed earlier, is necessary to effectively *employ econometric methods* and to infer characteristics of a population through a given sample. Enders also points out that t-statistics and Q-statistics are based upon the assumption that the data are stationary (p. 77). This implies a condition on the AR process of an ARMA model (roots of characteristic polynomial outside of unity circle).

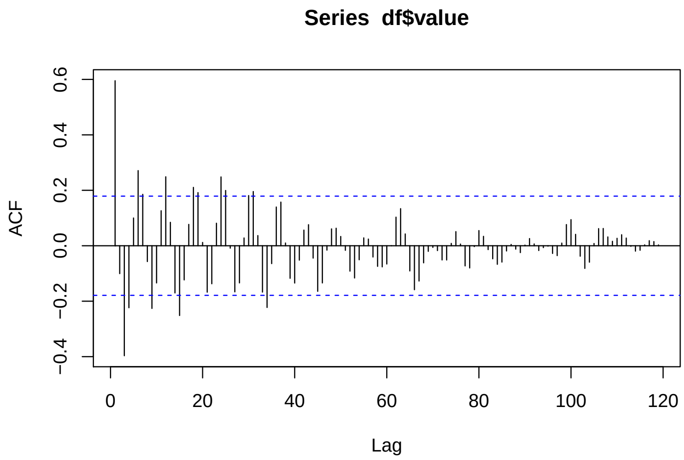
Furthermore, the model shall be *invertible* – i.e., if it can be represented by a finite or convergent AR model. This implies a condition on the MA process – i.e., if it can be written as an AR(∞).

We're going to check these conditions intuitively by plotting the ACFs and PACFs of the time series:

```
df <- data.frame(df)

pplot <- ggplot(data = df, aes(x = t, y = value)) + geom_line() +
    ggtitle("Time series plot") + theme_few()
pplot
```




```
acf_ts <- Acf(df$value, lag.max = 5000)
```




```
acf_test_values <- acf_ts$acf/sd(acf_ts$acf)

head(data.frame(acf_test_values))
```

```
##   acf_test_values
## 1       6.5814152
## 2       3.9180772
## 3      -0.6619326
## 4      -2.6109255
## 5      -1.4713722
## 6       0.6589976
```

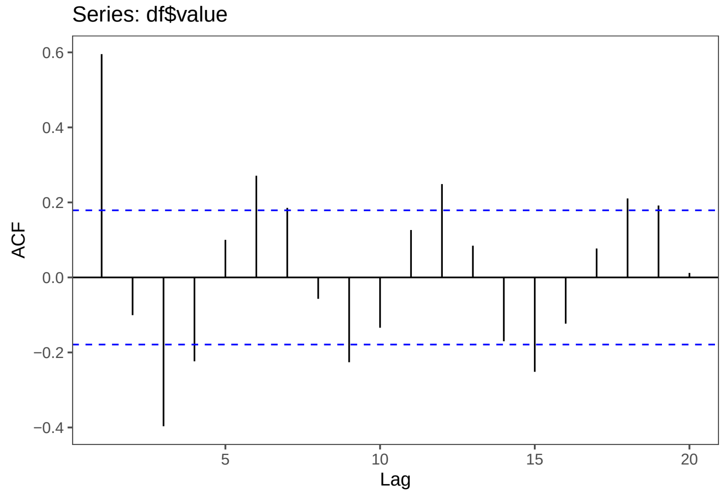
```
facst <- ggAcf(df$value, type = "correlation", lag.max = 20,
plot = T) + theme_few()
faclt <- ggAcf(df$value, type = "correlation", lag.max = 5000,
plot = T) + theme_few()

facpst <- ggPacf(df$value, type = "correlation", lag.max = 100,
plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```

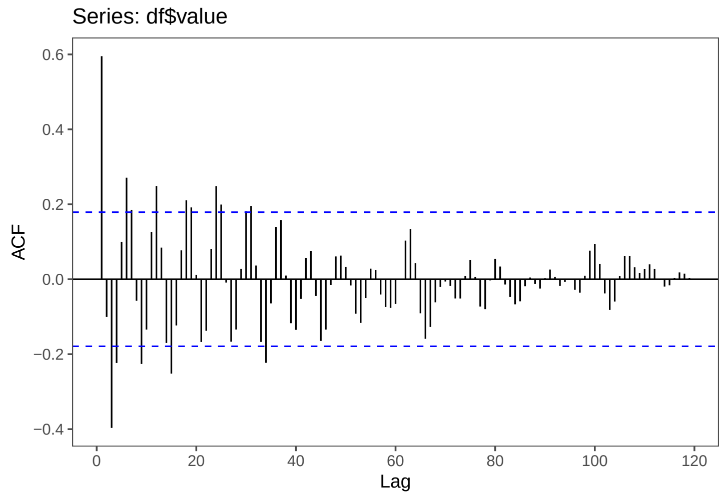
```r
facplt <- ggPacf(df$value, type = "correlation", lag.max = 5000,
plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```

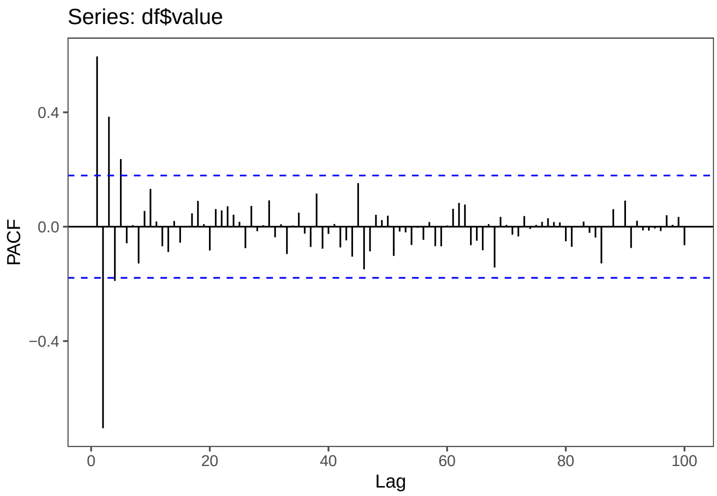
```r
facst
```



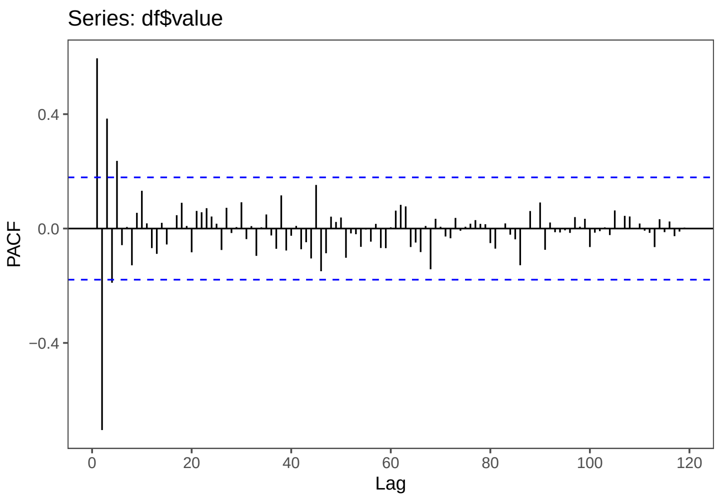

```r
faclt
```




```r
facpst
```




```
facplt
```



Aside from usual methods, we'll employ the following criteria:

- Akaike Information Criterion (AIC).

$$AIC = T * ln(SSR) + 2n$$

- Schwartz Bayesian Criterion (SBC).

$$SBC = T * ln(SSR) + n * ln(T)$$

n denotes the number of parameters estimated (an useful metric given the importance of the degrees of freedom). T denotes the number of *usable* observations. Note that, when comparing different models, it is important to *fix* T to ensure that the AIC and SBC values are comparable and are capturing only variations in the actual model and not the effect of changing T.

The objective with these criteria is to *minimize* their values. "As the fit of the model improves, the AIC and SBC will approach -∞." (p. 70) AIC and SBC have different advantages and drawbacks: while the former is biased toward overparametrization and more powerful in small samples, SBC is consistent and has superior large sample properties. If both metrics point to the same model, we should be fairly confident that it is, indeed, the correct specification.

It is also important to apply hypothesis tests to the estimates of the population parameters μ, σ^2 and $\rho_s - \bar{y}, \hat{sigma}^2, r_s$, respectively. Worthy of note here is r_s, which presents the following distributions under the null that y_t is stationary with $\varepsilon_t \sim \mathcal{N}$:

$$Var(r_s) = T^{-1} \qquad for\, s = 1$$

$$Var(r_s) = T^{-1}(1 + 2\sum_{j=1}^{s-1} r_j^2) \qquad for\, s > 1$$

The Q-statistic is also introduced by Enders in this chapter. It is used to test whether a group of autocorrelations is significantly different from zero.

$$Q = T \sum_{k=1}^{s} r_k^2$$

Under the null of $r_k = 0 \forall k$, Q is asymptotically χ^2 with s degrees of freedom. "Certainly, a white-noise process (in which all autocorrelations should be zero) would have a Q value of zero". (p. 68)

An alternative form for Q is presented by Ljung and Box (1978):

$$Q = T(T+2) \sum_{k=1}^{s} \frac{r_k^2}{(T-k)}$$

Furthermore, it is also important to check whether the residuals of the model are actually *white noise*. This can be done via the Q-statistic, which *should not result in the rejection of the null*. If that is not the case, the model specified is not the best one available, as there's still a relevant underlying variable (y or ε).

Let's now perform the *estimation stage*. This shall be done via the function *auto.arima* from the package *forecast*.

```
aa_model <- auto.arima(df$value, num.cores = 24, max.d = 0, max.D = 0,
stepwise = F)

summary(aa_model)
```

```
## Series: df$value
## ARIMA(2,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ar2    ma1    mean
##       0.7524  -0.5545  0.797  3.5305
## s.e.  0.0818   0.0813  0.064  0.3485
##
## sigma^2 estimated as 3.002:  log likelihood=-235.87
## AIC=481.75   AICc=482.27   BIC=495.68
##
## Training set error measures:
```

```
##                        ME     RMSE      MAE      MPE    MAPE      MASE
## Training set 0.01363546 1.703559 1.426984 5.237664 82.3373 0.5612557
##                     ACF1
## Training set 0.004670695
```

```
print("t-values: ")
```

```
## [1] "t-values: "
```

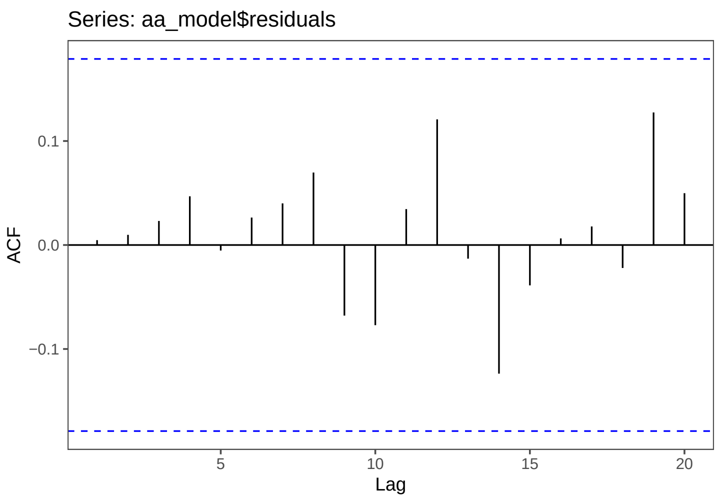
```
aa_t <- matrix(NA, nrow = 4)

for (i in c(1:4)) {

    aa_t[i] <- aa_model$coef[i]/sqrt(aa_model$var.coef[i, i])

}

aa_t <- data.frame(aa_t)

aa_t
```

```
##         aa_t
## 1   9.203615
## 2  -6.822352
## 3  12.444488
## 4  10.129782
```

```
aa_q <- Box.test(aa_model$residuals, lag = aa_model$arma[1] +
    aa_model$arma[2])
aa_q
```

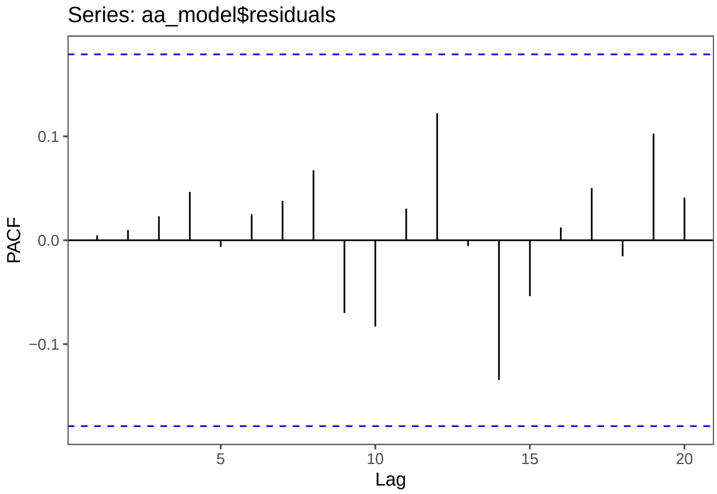
```
##
##  Box-Pierce test
##
## data:  aa_model$residuals
## X-squared = 0.078351, df = 3, p-value = 0.9943
```

```
ggAcf(aa_model$residuals, type = "correlation", lag.max = 20,
plot = T) + theme_few()
```




```
ggPacf(aa_model$residuals, type = "correlation", lag.max = 20,
plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```



The results of *auto.arima* imply that the best model is an ARMA(2,1):

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \theta_1 \varepsilon_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

Furthermore, the Q-statistic *(Box.test)* seems to indicate that ε_t is truly white noise.

Let's now run some different models and compare them against the results of *auto.arima*. We'll begin with some overspecified model. First, an ARMA(2,2):

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```r
arma22 <- Arima(df$value, order = c(2, 0, 2))

summary(arma22)
```

```
## Series: df$value
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ma1     ma2    mean
##       0.7417  -0.5502  0.8113  0.0149  3.5311
## s.e.  0.1442   0.0950  0.1692  0.1631  0.3514
##
## sigma^2 estimated as 3.028:  log likelihood=-235.87
## AIC=483.74   AICc=484.48   BIC=500.46
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.01360342 1.703508 1.426237 4.791703 81.74486 0.5609619
##                     ACF1
## Training set 0.001161589
```

```r
arma22_t <- matrix(NA, nrow = 5)

for (i in c(1:5)) {
```

```
        arma22_t[i] <- arma22$coef[i]/sqrt(arma22$var.coef[i, i])

    }

    arma22_t <- data.frame(arma22_t)

    arma22_t
```

```
##      arma22_t
## 1  5.14206433
## 2 -5.78853038
## 3  4.79577309
## 4  0.09134106
## 5 10.04968297
```

```
arma22_q <- Box.test(arma22$residuals, lag = arma22$arma[1] +
    arma22$arma[2])
arma22_q
```

```
##
##  Box-Pierce test
##
## data:  arma22$residuals
## X-squared = 0.26458, df = 4, p-value = 0.992
```

The t-value of $ma2$ is not able to reject the null hypothesis. Furthermore, the Q-statistic *(Box.test)* seems to indicate that ε_t is truly white noise.

Now, an ARMA(3,1):

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \Phi_3 y_{t-3} + \theta_1 \varepsilon_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```
arma31 <- Arima(df$value, order = c(3, 0, 1))

summary(arma31)
```

```
## Series: df$value
## ARIMA(3,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3     ma1    mean
##       0.7610  -0.565  0.0112  0.7923  3.5312
## s.e.  0.1215   0.137  0.1174  0.0825  0.3516
##
## sigma^2 estimated as 3.028:  log likelihood=-235.87
## AIC=483.74   AICc=484.48   BIC=500.46
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.01359734 1.703504 1.426202 4.779284 81.71699 0.5609482
##                      ACF1
## Training set 0.0008885573
```

```r
arma31_t <- matrix(NA, nrow = 5)

for (i in c(1:5)) {

    arma31_t[i] <- arma31$coef[i]/sqrt(arma31$var.coef[i, i])

}

arma31_t <- data.frame(arma31_t)

arma31_t
```

```
##      arma31_t
## 1  6.26539086
## 2 -4.12346904
## 3  0.09501299
## 4  9.59788435
## 5 10.04172094
```

```r
arma31_q <- Box.test(arma31$residuals, lag = arma31$arma[1] +
    arma31$arma[2])
arma31_q
```

```
##
##  Box-Pierce test
##
## data:  arma31$residuals
## X-squared = 0.25911, df = 4, p-value = 0.9923
```

The t-value of $ar3$ is not able to reject the null hypothesis. Furthermore, the Q-statistic *(Box.test)* seems to indicate that ε_t is truly white noise.

Now, let's try some *underspecified models*. Beginning with an ARMA(2,0):

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```r
arma20 <- Arima(df$value, order = c(2, 0, 0))

summary(arma20)
```

```
## Series: df$value
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2    mean
##       1.0226  -0.7153  3.5194
## s.e.  0.0634   0.0635  0.2694
##
## sigma^2 estimated as 4.24:  log likelihood=-256.37
## AIC=520.74   AICc=521.09   BIC=531.89
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE      ACF1
## Training set 0.01106086 2.033314 1.630805 73.16585 128.4039 0.6414218 0.2915253
```

```r
arma20_t <- matrix(NA, nrow = 3)

for (i in c(1:3)) {

    arma20_t[i] <- arma20$coef[i]/sqrt(arma20$var.coef[i, i])

}

arma20_t <- data.frame(arma20_t)

arma20_t
```

```
##    arma20_t
## 1  16.12226
## 2 -11.26848
## 3  13.06561
```

```r
arma20_q <- Box.test(arma20$residuals, lag = arma20$arma[1] +
    arma20$arma[2])
arma20_q
```

```
##
##  Box-Pierce test
##
## data:  arma20$residuals
## X-squared = 13.728, df = 2, p-value = 0.001045
```

The Q-statistic indicates that there is an ommited variable – namely, ε_{t-1} that we have just excluded from the model.

Now, an ARMA(1,1):

$$y_t = c + \Phi_1 y_{t-1} + +\theta_1 \varepsilon_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```r
arma11 <- Arima(df$value, order = c(1, 0, 1))

summary(arma11)
```

```
## Series: df$value
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ma1    mean
##       0.4476  0.9244  3.6027
## s.e.  0.0831  0.0323  0.6234
##
## sigma^2 estimated as 4.026:  log likelihood=-253.74
## AIC=515.48   AICc=515.82   BIC=526.63
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.006283661 1.981184 1.621537 51.07775 142.3988 0.6377767
##                   ACF1
## Training set 0.2028683
```

```r
arma11_t <- matrix(NA, nrow = 3)

for (i in c(1:3)) {

    arma11_t[i] <- arma11$coef[i]/sqrt(arma11$var.coef[i, i])

}

arma11_t <- data.frame(arma11_t)

arma11_t
```

```
##    arma11_t
## 1  5.387948
## 2 28.624391
## 3  5.779126
```

```r
arma11_q <- Box.test(arma11$residuals, lag = arma11$arma[1] +
    arma11$arma[2])
arma11_q
```

```
##
##  Box-Pierce test
##
## data:  arma11$residuals
## X-squared = 12.668, df = 2, p-value = 0.001775
```

Again, the Q-statistic indicates that there is an ommited variable – namely, y_{t-1} that we have just excluded from the model.

Finally, let's compare the AIC and BIC values for all these models.

```r
criteria <- matrix(NA, nrow = 5, ncol = 3)

aa_criteria <- data.frame("ARMA(2,1)*", aa_model$aic, aa_model$bic)

names(aa_criteria) <- c("Model", "AIC", "BIC")

arma22_criteria <- data.frame("ARMA(2,2)", arma22$aic, arma22$bic)

names(arma22_criteria) <- c("Model", "AIC", "BIC")

arma31_criteria <- data.frame("ARMA(3,1)", arma31$aic, arma31$bic)

names(arma31_criteria) <- c("Model", "AIC", "BIC")

arma20_criteria <- data.frame("ARMA(2,0)", arma20$aic, arma20$bic)

names(arma20_criteria) <- c("Model", "AIC", "BIC")

arma11_criteria <- data.frame("ARMA(1,1)", arma11$aic, arma11$bic)

names(arma11_criteria) <- c("Model", "AIC", "BIC")


criteria <- rbind.data.frame(aa_criteria, arma22_criteria, arma31_criteria,
    arma20_criteria, arma11_criteria)

criteria
```

```
##        Model      AIC      BIC
## 1 ARMA(2,1)* 481.7460 495.6834
## 2  ARMA(2,2) 483.7376 500.4625
```

```
## 3  ARMA(3,1) 483.7369 500.4619
## 4  ARMA(2,0) 520.7381 531.8880
## 5  ARMA(1,1) 515.4753 526.6253
```

As we can clearly see, the model chosen by *auto.arima* is the optimal choice according both to AIC and BIC.

Chapter 6

Forecasting

Suppose that you observe a time series up to period T, $Y_1, Y_2, ..., Y_T$, and would like to forecast its value in $T+1$, or, more generically, up to a time horizon $h \geq 1$: $Y_{T+1}, ..., Y_{T+h}$.

Let $g(Y_1, ..., Y_T)$ be a general predictor of Y_{T+1} built with information up to period T. We can measure its utility with its *mean squared error (MSE)*:

$$MSE[g(Y_1, ..., Y_T)] := \mathbb{E}[(Y_{T+1} - g(Y_1, ..., Y_T))^2]$$

We know that the conditional mean is the predictor that minimizes MSE:

$$\mathbb{E}\left(Y_{T+1} \mid Y_T, \ldots, Y_1\right) = \operatorname{argmin}_g \mathbb{E}\left[\left(Y_{t+1} - g\left(Y_1, Y_2, \ldots, Y_T\right)\right)^2\right]$$

Unfortunately, we usually don't have the functional form of $\mathbb{E}\left(Y_{T+1} \mid Y_T, \ldots, Y_1\right)$, and postulate its best linear form:

$$\pi\left(Y_{T+1} \mid Y_T, \ldots, Y_1\right) = \alpha + \beta_0 Y_T + \beta_1 Y_{T-1} + \ldots \beta_{T-1} Y_1$$

Denote the best linear predictor with information up to T as $Y_{T+1|T} := \pi(Y_{T+1}|Y_t, ..., Y_1)$, and its estimated version as $\hat{Y}_{T+1|T} := \hat{\pi}(Y_{T+1}|Y_t, ..., Y_1) = \hat{\alpha} + \hat{\beta}_0 Y_T + \hat{\beta}_1 Y_{T-1} + \ldots \hat{\beta}_{T-1} Y_1$.

6.1 Forecasting with an AR(1) model

Remember from Definition 2.6.1 that an AR(1) model has the following form:

$$Y_t = c + \phi Y_{t-1} + \varepsilon_t$$

6.1.1 Forecast

Given that $\mathbb{E}(\varepsilon_{t+1}|Y_T) = 0$ (as $\epsilon \sim wn(0, \sigma^2)$), we have:

$$Y_{T+1|T} := \pi\left(Y_{T+1} \mid Y_T, \ldots, Y_1\right) = c + \phi Y_T$$

For $T+2$, we have:

$$\begin{aligned} Y_{T+2|T} &:= \pi\left(Y_{T+2} \mid Y_T, \ldots, Y_1\right) \\ &= \pi\left(\pi\left(Y_{T+2} \mid Y_{T+1}, Y_T, \ldots, Y_1\right) \mid Y_T, \ldots, Y_1\right) \\ &= \pi\left(c+\phi Y_{T+1} \mid Y_T, \ldots, Y_1\right) \\ &= c+\phi\left(c+\phi Y_T\right)=(1+\phi) c+\phi^2 Y_T \end{aligned}$$

The clear pattern here yields, more generally:

$$Y_{T+h|T}=\left(1+\phi+\phi^2+\cdots+\phi^{h-1}\right) c+\phi^h Y_T$$

6.1.2 Forecast error

It is also easy to verify that the *forecast error* for $h=1$ is given by:

$$u_{T+1|T}:=Y_{T+1}-Y_{T+1|T}=Y_{T+1}-c-\phi Y_T=\varepsilon_{T+1}$$

For $h=2$, we have:

$$\begin{aligned} u_{T+2|T} &:= Y_{T+2}-Y_{T+2|T} \\ &= c+\phi Y_{T+1}+\epsilon_{T+2}-\left(c+\phi Y_{T+1|T}\right) \\ &= \phi\left(Y_{T+1}-Y_{T+1|T}\right)+\epsilon_{T+2} \\ &= \phi u_{T+1|T}+\epsilon_{T+2}=\phi \epsilon_{T+1}+\epsilon_{T+2} \end{aligned}$$

More generally, for a given time horizon h:

$$u_{T+h|T}=\left(\epsilon_{T+h}+\phi \epsilon_{T+h-1}+\cdots+\phi^{h-1} \epsilon_{T+1}\right)$$

6.1.3 Mean reversion

Note that, as h increases, the prediction *reverts to the unconditional mean:*

$$\lim_{h \rightarrow \infty} Y_{T+h|T}=c \lim_{h \rightarrow \infty} \sum_{i=0}^{h-1} \phi^i+Y_T \lim_{h \rightarrow \infty} \phi^h=\frac{c}{1-\phi}=: \mu$$

The variance of the forecast error is given by:

$$Var(u_{T+h|T})=(1+\phi^2+\phi^4+...+\phi^{2(h-1)})\sigma^2$$

Taking the limit as $h \rightarrow \infty$, the variance of the forecast error *also approaches the unconditional variance of the process:*

$$\lim_{h \rightarrow \infty} \mathbb{V}\left(u_{T+h|T}\right)=\sigma^2 \lim_{h \rightarrow \infty} \sum_{i=0}^{h-1} \phi^{2i}=\frac{\sigma^2}{1-\phi^2}=: \gamma_0$$

6.2 Forecasting with an AR(p) model

6.2.1 Forecast

The same procedure can be applied to an AR(p). For $h = 1$:

$$Y_{T+1|T} = \pi\left(Y_{T+1} \mid Y_T, \ldots, Y_1\right) = c + \phi_1 Y_T + \cdots + \phi_p Y_{T-p+1}$$

For a given h, proceed recursively:

$$\begin{aligned}
Y_{T+2|T} &= c + \phi_1 Y_{T+1|T} + \phi_2 Y_T + \phi_3 Y_{T-1} + \cdots + \phi_p Y_{T+2-p} \\
Y_{T+3|T} &= c + \phi_1 Y_{T+2|T} + \phi_2 Y_{T+1|T} + \phi_3 Y_T + \cdots + \phi_p Y_{T+3-p} \\
Y_{T+4|T} &= c + \phi_1 Y_{T+3|T} + \phi_2 Y_{T+2|T} + \phi_3 Y_{T+1|T} + \cdots + \phi_p Y_{T+4-p} \\
&\vdots \\
Y_{T+h|T} &= c + \phi_1 Y_{T+h-1|T} + \phi_2 Y_{T+h-2|T} + \cdots + \phi_p Y_{T+h-p|T}
\end{aligned}$$

6.2.2 Forecast error

For $h = 1$, the forecast error is given by:

$$u_{T+1|T} := Y_{T+1} - Y_{T+1|T} = \varepsilon_{T+1}$$

As h increases:

$$\begin{aligned}
u_{T+2|T} &= \phi_1 u_{T+1|T} + \epsilon_{T+2} \\
u_{T+3|T} &= \phi_1 u_{T+2|T} + \phi_2 u_{T+1|T} + \epsilon_{T+3} \\
&\vdots \\
u_{T+h|T} &= \phi_1 u_{T+h-1|T} + \cdots + \phi_{h-1} u_{T+1|T} + \epsilon_{T+h},
\end{aligned}$$

where $\phi_{h-1} = 0$ for $h - 1 > p$.

6.3 Forecasting with a MA(1) model

Remember from 2.2.1 that a MA(1) model is given by

$$Y_t = c + \theta\varepsilon_{t-1} + \varepsilon_t$$

$$Y_{T+1|T} := \pi(Y_{T+1}|Y_T, ..., Y_1)$$

We have seen that, if the MA(1) is invertible, we can write it as an AR(∞). This would mean, however, that the forecast errors would depend on *all past values* – notwithstanding the decreasing dependence, due to ergodicity. Given a fixed ε_0, we can reconstruct the entire error series:

$$\begin{aligned}
\epsilon_1 &= Y_1 - c - \theta\epsilon_0 \\
\epsilon_2 &= Y_2 - c - \theta\epsilon_1 \\
&\vdots \\
\epsilon_T &= Y_T - c - \theta\epsilon_{T-1}
\end{aligned}$$

If we do not know ε_0, we can approximate it by its (known!) mean, $\tilde{\varepsilon}_0 = 0$. If T is large enough and $|\theta| < 1$, this is a good approximation. That is the case because for large T, the influence of the assumption $\tilde{\varepsilon}_0 = 0$ is geometrically diminished over time, given $|\theta| < 1$.

6.3.1 Forecast

We are now able to construct the forecast using the estimated $\tilde{\varepsilon}_T$.

$$Y_{T+1|T} := c + \theta\tilde{\varepsilon}_T$$

Iteratively for larger horizons:

$$Y_{T+2|T} = c + \theta\widetilde{\epsilon}_{T+1} = c + \theta\left(Y_{T+1|T} - c - \theta\widetilde{\epsilon}_T\right) = c$$

Note that *the MA(1) process is not predictable for $h > 1$.* The forecast immediately mean reverts at $h > 1$.

6.3.2 Forecast error

The forecast error, assuming $\tilde{\varepsilon}_T \approx \varepsilon_T$, is given by:

$$u_{T+1|T} := Y_{T+1} - Y_{T+1|T} = \varepsilon_{T+1}$$

This means that the variance of the forecast error is σ^2. For larger horizons, the variance converges to the unconditional variance:

$$u_{T+h|T} := Y_{T+h} - Y_{T+h|T} = \varepsilon_{T+h} + \theta\varepsilon_{T+h-1}, \forall h > 1$$

Thus, $Var(u_{T+h|T}) = (1+\theta)^2\sigma^2 = \gamma_0, \forall h > 1$.

6.4 Forecasting with a MA(q) model

We can proceed iteratively for the MA(q) model.

$$\begin{aligned}
Y_{T+1|T} &= c + \theta_1\widetilde{\epsilon}_T + \cdots + \theta_q\widetilde{\epsilon}_{T+1-q} \\
Y_{T+2|T} &= c + \theta_2\widetilde{\epsilon}_T + \cdots + \theta_q\widetilde{\epsilon}_{T+2-q} \\
Y_{T+3|T} &= c + \theta_3\widetilde{\epsilon}_T + \cdots + \theta_q\widetilde{\epsilon}_{T+3-q} \\
&\vdots \\
Y_{T+q|T} &= c + \theta_q\widetilde{\epsilon}_T
\end{aligned}$$

Note that, after q periods, the prediction is the unconditional mean c.

6.5 Forecast with an ARMA(p,q) model

Let's combine the procedures presented for AR(p) and MA(q). For $h = 1$:

$$Y_{T+1|T} = c + \phi_1 Y_T + \cdots + \phi_p Y_{T-p+1} + \theta_1\widetilde{\varepsilon}_T + \cdots + \theta_q\widetilde{\varepsilon}_{T+1-q},$$

where $\{\widetilde{\varepsilon}_T, ..., \widetilde{\varepsilon}_{T+1-q}\}$ were obtained from the reconstruction process explained in the MA(q) case. However, here we need, aside from the first q innovation values, also the *last p observations before the first one.* That is the case because of the expression for $\widetilde{\varepsilon}_1$:

$$\widetilde{\epsilon}_1 = Y_1 - c - \phi_1 Y_0 - \cdots - \phi_p Y_{p-1} - \theta_1\widetilde{\epsilon}_0 - \theta_2\widetilde{\epsilon}_{-1} - \cdots - \theta_q\widetilde{\epsilon}_{1-q}$$

Alternatively, we can start reconstructing the residuals from $p+1$.

6.5.1 Forecast

From this, we can obtain the forecasts for any horizon $h \geq 1$, given $h > p > q$:

$$\begin{aligned} Y_{T+1|T} &= c + \phi_1 Y_T + \cdots + \phi_p Y_{T+1-p} + \theta_1 \widetilde{\epsilon}_T + \cdots + \theta_q \widetilde{\epsilon}_{T+1-q} \\ Y_{T+2|T} &= c + \phi_1 Y_{T+1|T} + \cdots + \phi_p Y_{T+2-p} + \theta_2 \widetilde{\epsilon}_T + \cdots + \theta_q \widetilde{\epsilon}_{T+2-q} \\ Y_{T+3|T} &= c + \phi_1 Y_{T+2|T} + \cdots + \phi_p Y_{T+3-p} + \theta_3 \widetilde{\epsilon}_T + \cdots + \theta_q \widetilde{\epsilon}_{T+3-q} \\ &\vdots \end{aligned}$$

$$\begin{aligned} Y_{T+q|T} &= c + \phi_1 Y_{T+q-1|T} + \cdots + \phi_p Y_{T+q-p} + \theta_q \widetilde{\epsilon}_T \\ Y_{T+q+1|T} &= c + \phi_1 Y_{T+q|T} + \cdots + \phi_p Y_{T+q+1-p} \\ &\vdots \\ Y_{T+h|T} &= c + \phi_1 Y_{T+h-1|T} + \cdots + \phi_p Y_{T+h-p|T} \end{aligned}$$

6.6 Confidence intervals for forecasts

Given a forecast $\hat{Y}_{T+h|T}$, we'd like to have a *confidence interval* for it. In practice, forecasting involves two types of errors:

- **Estimation error:** $Y_{T+h|T} - \hat{Y}_{T+h|T}$. This is due to the estimation of the ARMA model, as we are always working with a sample.
- **Forecast error:** $u_{T+h|T} = Y_{T+h} - Y_{T+h|T}$. This is the portion of the error that would exist event if we knew the true population parameters.

From this, we have the *aggregate error:*

$$Y_{T+h} - \hat{Y}_{T+h|T} = (Y_{T+h} - Y_{T+h|T}) + (Y_{T+h|T} - \hat{Y}_{T+h|T})$$

6.6.1 Normally distributed errors

A first way to tackle the issue of confidence intervals for forecasts is to *assume normality*: $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$.

In that case, we know that $Y_{T+h}, \hat{Y}_{T+h|T}$ will also be normally distributed, as the first term consists of a combination of past errors and past values of Y, and the second term is asymptotically normal given the properties of ARMA estimators. This means that the confidence interval is known to us, for $\alpha = 5\%$:

$$\left[\widehat{Y}_{T+h|T} \pm 1,96\sqrt{\mathbb{V}\left(u_{T+h|T}\right)}\right]$$

We clearly do not observe $Var(u_{T+h|T})$, but we know its functional form and, thus, can consistently estimate it. For an AR(1), for example:

$$\widehat{\mathbb{V}}\left(u_{T+h|T}\right) = \widehat{\sigma}^2 \sum_{i=0}^{h-1} \widehat{\phi}^{2i}$$

Note that, even if we assume that the innovation terms are normally distributed, we *still need an asymptotic argument* to guarantee that $\widehat{Y}_{T+h|T} \xrightarrow{p} Y_{T+h|T}$ and $\widehat{\mathbb{V}}\left(u_{T+h|T}\right) \xrightarrow{p} \mathbb{V}\left(u_{T+h|T}\right)$.

6.6.2 Bootstrapping

What if the errors *are not normally distributed?* This makes the issue a lot more complicated. Note that, even if we apply the CLT to argue that $Y_{T+h|T} - \hat{Y}_{T+h|T} \sim \mathcal{N}(\cdot)$, what makes us believe that $(Y_{T+h} - Y_{T+h|T})$ is also normally distributed? In the last subsection, this condition was valid because of the distribution of the errors. There's no asymptotical argument to be made.

In practice, we don't know the distribution of the error terms. In fact, they are *rarely normally distributed!* A procedure to construct confidence intervals in this setting is called *bootstrap.* For ARMA models, it consists in generating simulated samples from the actual sample and repeat the estimations for each simulated sample. This yields an arbitrarily large number of estimates *from the same data.* From this, we can use its *empirical distribution* to find the confidence interval.

Bootstrapping has a number of advantages over the traditional procedures for obtaining a confidence interval. In many cases, the convergence is faster than the usual $1/\sqrt{T}$, and it frequently performs better in small sample environments. It can also be applied in settings in which asymptotic theory is very intricate. In practice, we almost always use some sort of bootstrap method.

The essential idea behind bootstrapping is to assume that ε_t is *iid* with a cdf F. Suppose that you have T residuals, $\hat{\varepsilon}_t$. Its empirical distribution is simply:

$$\hat{F}_T(v) = \frac{\{no. \, residuals \leq v\}}{T}, \quad v \in \mathbb{R}$$

The Law of Large Numbers guarantees that, as $T \to \infty$,

$$\hat{F}_T(v) \to_{a.s.} F(v)$$

This means that the empirical distribution of the residuals becomes arbitrarily close to the real distribution as T grows.

Procedure

Bootstrapping for an ARMA(p,q) model involves the following steps:

1. - Estimate ARMA(p,q)

$$Y_t = c + \sum_{j=1}^{p} \phi_j Y_{t-j} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

   - Calculate the residuals of the regression:

$$\hat{\varepsilon}_t := Y_t - (\hat{c} + \sum_{j=1}^{p} \hat{\phi}_j Y_{t-j} + \sum_{j=1}^{q} \hat{\theta}_j \varepsilon_{t-j})$$

   - If the residuals do not have mean 0, create the centered residuals:

$$\tilde{\varepsilon}_t = \hat{\varepsilon}_t - \frac{1}{t} \sum_{t=1}^{T} \hat{\varepsilon}_t$$

2. - Select at random, with replacement, a sample with $T + m$ elements, $m >> 0$:

$$\{\varepsilon_1^*, ..., \varepsilon_{T+m}^*\}$$

- Create a series $\{Y_t^*\}_{t=1}^{T+m}$:

$$Y_t^* = Y_t, 1 \leq t \leq max(p, q)$$

$$Y_t^* = \hat{c} + \sum_{j=1}^{p} \hat{\phi}_j Y_{t-j} + \sum_{j=1}^{q} \hat{\theta}_j \varepsilon_{t-j}^* + \varepsilon_t^*, max(p, q) < t \leq T + m$$

3. - Using the simulated sample $\{Y_t^*\}_{t=1}^{T+m}$, create a forecast for $h > 0$ periods using the estimated coefficients *obtained with the real sample.*

- This yields a vector of dimension h containing the forecasts in the form:

$$(\hat{Y}_{T+1}^*, ..., \hat{Y}_{T+h}^*)$$

- Repeat steps 2 and 3 for S times. Create a matrix with the results.
- This yields a S x h matrix where each row is equal to the aforementioned vector.

Chapter 7

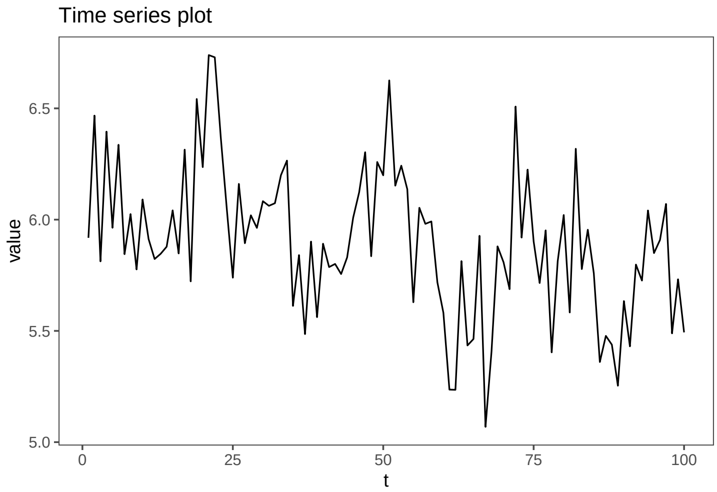
Problem 4: Cross-validation and bootstrap

In this problem, we'll be tackling the issue of *forecasting* of an ARMA model. The problem is split in two parts: (i) *cross-validation*; and (ii) *bootstrapping*.

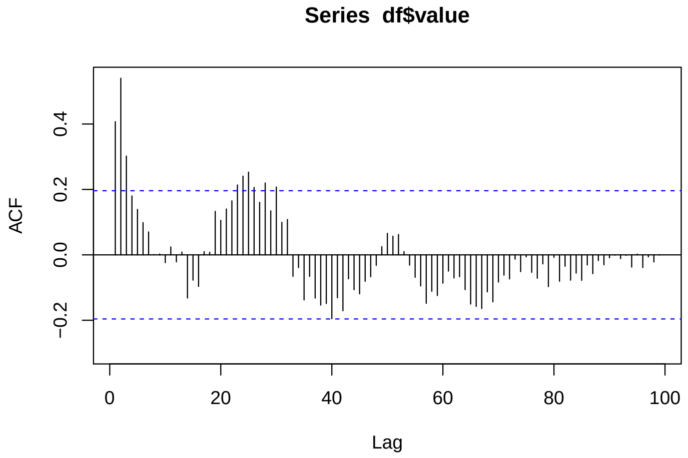
7.1 Identification and estimation

First, let's identify the best model for our time series.

```
df <- data.frame(df)

pplot <- ggplot(data = df, aes(x = t, y = value)) + geom_line() +
    ggtitle("Time series plot") + theme_few()
pplot
```



```r
acf_ts <- Acf(df$value, lag.max = 5000)
```




```r
acf_test_values <- acf_ts$acf/sd(acf_ts$acf)

head(data.frame(acf_test_values))
```

```
##   acf_test_values
## 1        6.176432
## 2        2.515951
## 3        3.335438
## 4        1.864909
## 5        1.112884
## 6        0.858639
```

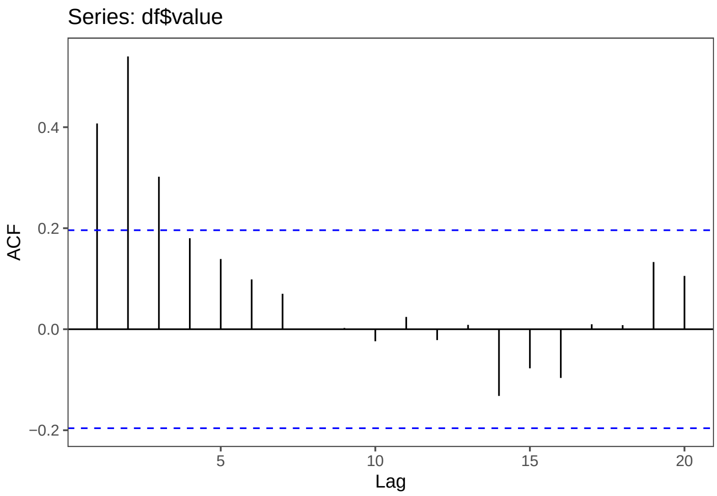
```r
facst <- ggAcf(df$value, type = "correlation", lag.max = 20,
plot = T) + theme_few()
faclt <- ggAcf(df$value, type = "correlation", lag.max = 5000,
plot = T) + theme_few()

facpst <- ggPacf(df$value, type = "correlation", lag.max = 100,
plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```

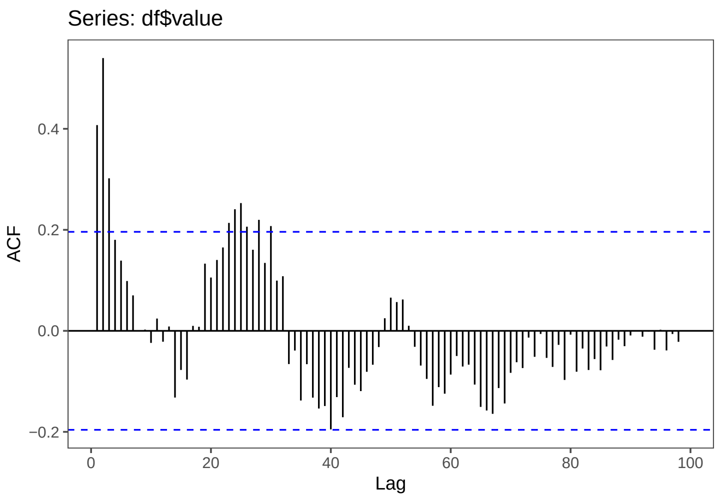
```r
facplt <- ggPacf(df$value, type = "correlation", lag.max = 5000,
plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```

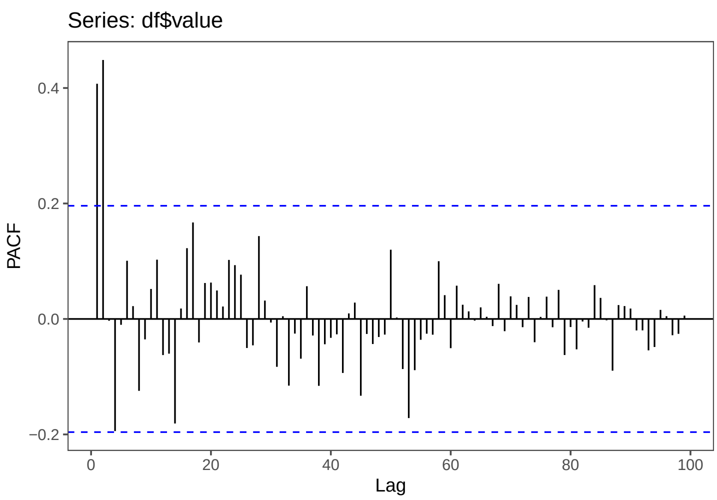
```r
facst
```



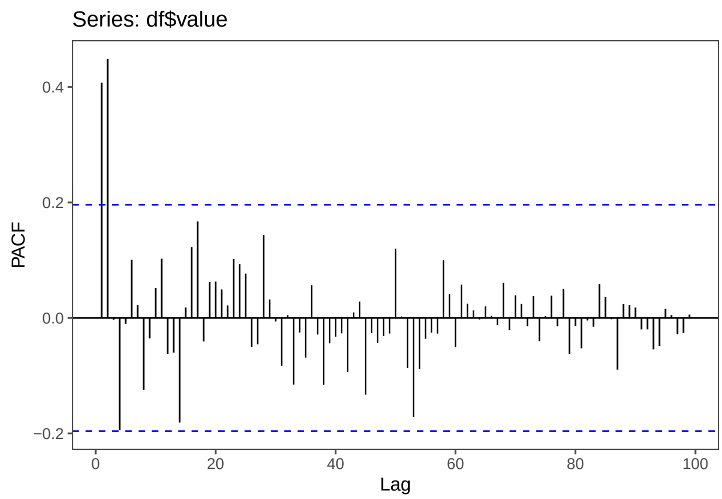

```
faclt
```




```
facpst
```



```r
facplt
```



We'll now use the function *auto.arima* from the package *forecast* to identify and estimate the model.

```r
aa_model <- auto.arima(df$value, num.cores = 24, max.d = 0, stepwise = F)

summary(aa_model)
```

```
## Series: df$value
## ARIMA(0,0,3) with non-zero mean
##
## Coefficients:
##          ma1     ma2     ma3    mean
##       0.1814  0.6647  0.4001  5.8982
## s.e.  0.0852  0.0750  0.0949  0.0562
##
## sigma^2 estimated as 0.0667:  log likelihood=-5.42
## AIC=20.85   AICc=21.49   BIC=33.88
##
## Training set error measures:
##                        ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.002315954 0.2530428 0.2131067 -0.2268814 3.612855 0.7314965
##                    ACF1
## Training set 0.03868106
```

```r
print("t-values: ")
```

```
## [1] "t-values: "
```

```r
aa_t <- matrix(NA, nrow = aa_model$arma[1] + aa_model$arma[2])

for (i in c(1:4)) {

    aa_t[i] <- aa_model$coef[i]/sqrt(aa_model$var.coef[i, i])

}

aa_t <- data.frame(aa_t)
```

```r
aa_t
```

```
##          aa_t
## 1   2.128691
## 2   8.861580
## 3   4.216481
## 4 105.004537
```

```r
aa_q <- Box.test(aa_model$residuals, lag = aa_model$arma[1] +
    aa_model$arma[2])
aa_q
```

```
##
##  Box-Pierce test
##
## data:  aa_model$residuals
## X-squared = 0.35002, df = 3, p-value = 0.9504
```

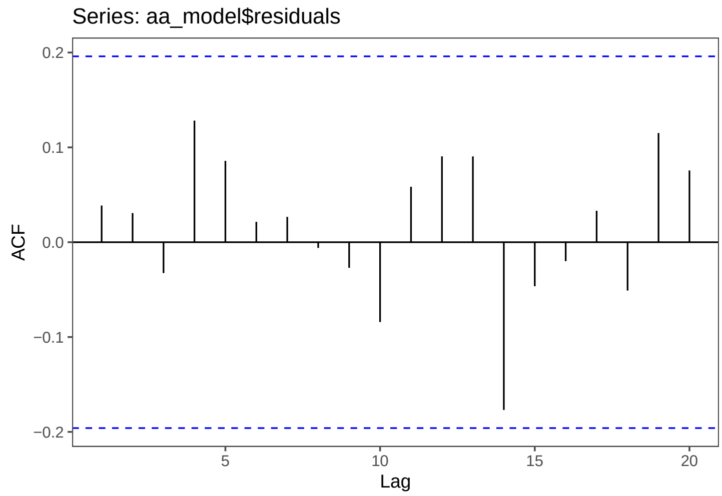
```r
criteria <- matrix(NA, nrow = 1, ncol = 3)

aa_criteria <- data.frame("MA(3)*", aa_model$aic, aa_model$bic)

names(aa_criteria) <- c("Model", "AIC", "BIC")

aa_criteria
```

```
##    Model      AIC      BIC
## 1 MA(3)* 20.84963 33.87549
```

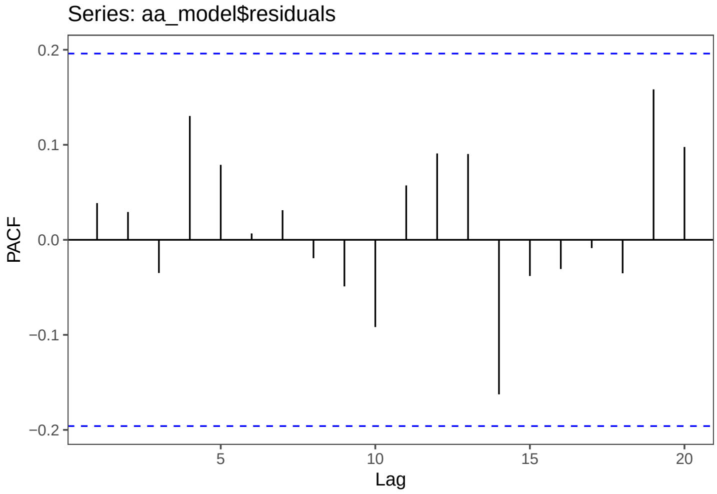
```r
fac_e <- ggAcf(aa_model$residuals, type = "correlation", lag.max = 20,
plot = T) + theme_few()

facp_e <- ggPacf(aa_model$residuals, type = "correlation", lag.max = 20,
plot = T) + theme_few()
```

```
## Warning: Ignoring unknown parameters: type
```

```r
fac_e
```




```r
facp_e
```




```r
mean(aa_model$residuals)
```

```
## [1] -0.002315954
```

The results of *auto.arima* imply that the best model is an ARMA(0,3) – i.e., a MA(3):

$$y_t = c + +\theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \theta_3\varepsilon_{t-3} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

Furthermore, the Q-statistic *(Box.test)* seems to indicate that ε_t is truly white noise.

7.2 Cross-validation

Let's now cross-validate or model. This will now be done manually; afterwards, an automatized version from *fpp* shall be presented.

Let $h := 5$; $frac = 0.2$. T is the size of our sample; k is the *training* database. The remainder shall be used for testing purposes.

As we have discovered previously, *auto.arima* yields a $\mathbf{MA(3)}$ model. It will now be used.

```r
h <- 5

frac <- 0.2

T <- length(df$value)

k <- floor((1 - frac) * T)

# Estimating MA(3) with k = 80
fit <- Arima(df$value[1:k], order = c(0, 0, 3))

# Generating predictions from the model
pred <- predict(fit, n.ahead = h)

# Calculating errors between the predicted values of the
# model and the actual values of the testing database

e <- df$value[(k + h)] - pred$pred[h]

e
```

```
## [1] -0.1951299
```

Let's now update our training database iteratively with a for loop.

```
e <- matrix(NA, nrow = 100)

# Updating the model

for (i in k:(T - h)) {

    fit <- Arima(df$value[1:i], order = c(0, 0, 3))

    pred <- predict(fit, n.ahead = h)

    e[i, 1] <- df$value[(i + h)] - pred$pred[h]

}
```

With the matrix e in hands, we can now calculate MSE:

```
mse <- mean(e^2, na.rm = T)
```

This procedure can now be used to compare other models against the model from *auto.arima*.

```
max_p <- 5

max_q <- 5

e <- matrix(NA, nrow = 100, ncol = (max_p + 1) * (max_q + 1))

pred <- vector("list", (max_p + 1) * (max_q + 1))

fit <- vector("list", (max_p + 1) * (max_q + 1))

# Updating the model
for (u in 0:max_q) {

for (j in 0:max_p) {


for (i in k:(T - h)) {

            fit[[(((max_p + 1) * j) + u + 1)]] <- Arima(df$value[1:i],
order = c(j, 0, u))

# fit <- append(fit, Arima(df$value[1:i], order = c(j,0,u)))

# pred <- append(pred, predict(fit[[(j+u)]], n.ahead = h))

            pred[[(((max_p + 1) * j) + u + 1)]] <- predict(fit[[(((max_p +
                1) * j) + u + 1)]], n.ahead = h)

            e[i, (((max_p + 1) * j) + u + 1)] <- df$value[(i +
                h)] - pred[[(((max_p + 1) * j) + u + 1)]]$pred[h]

        }

    }

}

mse <- matrix(NA, nrow = ((max_p + 1) * (max_q + 1)), ncol = 1)



mse <- colMeans(e^2, na.rm = T)

mse
```

```
##  [1] 0.1357466 0.1354001 0.1368083 0.1374243 0.1376508 0.1441940 0.1347115
##  [8] 0.1269779 0.1347789 0.1373465 0.1398588 0.1436175 0.1313779 0.1315448
## [15] 0.1435805 0.1355649 0.1421277 0.1335153 0.1316765 0.1333955 0.1400838
## [22] 0.1427467 0.1473227 0.1347447 0.1320856 0.1333025 0.1354734 0.1341742
## [29] 0.1380676 0.1357880 0.1346228 0.1382810 0.1319484 0.1308446 0.1382417
## [36] 0.1327046
```

```
optimal_index <- which.min(mse)

cv_model <- fit[[optimal_index]]

summary(cv_model)
```

```
## Series: df$value[1:i]
## ARIMA(1,0,1) with non-zero mean
##
## Coefficients:
##          ar1      ma1    mean
##       0.8253  -0.4888  5.9125
## s.e.  0.0814   0.1118  0.0815
##
## sigma^2 estimated as 0.08209:  log likelihood=-14.72
## AIC=37.43   AICc=37.88   BIC=47.65
##
## Training set error measures:
##                        ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.002725722 0.2819456 0.2228183 -0.2756862 3.787114 0.7600473
##                    ACF1
## Training set -0.1279409
```

The cross-validation method constructed above yielded an ARMA(1,1):

$$y_t = c + \phi_1 y_{t-1} + +\theta_1 \varepsilon_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```
cv_fc <- forecast(cv_model, h = h)

autoplot(cv_fc)
```

7.3 Bootstrapping

Now, let's proceed to *bootstrapping*. It envolves the following steps:

1. - Estimate ARMA(p,q)

$$Y_t = c + \sum_{j=1}^{p} \phi_j Y_{t-j} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t$$

   - Calculate the residuals of the regression:

$$\hat{\varepsilon}_t := Y_t - (\hat{c} + \sum_{j=1}^{p} \hat{\phi}_j Y_{t-j} + \sum_{j=1}^{q} \hat{\theta}_j \varepsilon_{t-j})$$

   - If the residuals do not have mean 0, create the centered residuals:

$$\tilde{\varepsilon}_t = \hat{\varepsilon}_t - \frac{1}{t} \sum_{t=1}^{T} \hat{\varepsilon}_t$$

2. - Select at random, with replacement, a sample with $T + m$ elements, $m >> 0$:

$$\{\varepsilon_1^*, ..., \varepsilon_{T+m}^*\}$$

   - Create a series $\{Y_t^*\}_{t=1}^{T+m}$:

$$Y_t^* = Y_t, 1 \leq t \leq max(p, q)$$

$$Y_t^* = \hat{c} + \sum_{j=1}^{p} \hat{\phi}_j Y_{t-j} + \sum_{j=1}^{q} \hat{\theta}_j \varepsilon_{t-j}^* + \varepsilon_t^*, max(p, q) < t \leq T + m$$

3. - Using the simulated sample $\{Y_t^*\}_{t=1}^{T+m}$, create a forecast for $h > 0$ periods using the estimated coefficients *obtained with the real sample.*
   - This yields a vector of dimension h containing the forecasts in the form:

$$(\hat{Y}_{T+1}^*, ..., \hat{Y}_{T+h}^*)$$

   - Repeat steps 2 and 3 for S times. Create a matrix with the results.
   - This yields a S x h matrix where each row is equal to the aforementioned vector.

We'll use, again, the optimal model from *auto.arima*, MA(3):

$$y_t = c + +\theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \theta_3 \varepsilon_{t-3} + \varepsilon_t, \quad \varepsilon_t \sim wn(0, \sigma^2)$$

```
S <- 1000

m <- 100



optimal_p <- aa_model$arma[1]

optimal_q <- aa_model$arma[2]

e_sample <- data.frame(matrix(NA, nrow = S, ncol = (length(df$value) +
    m)))

y_star <- data.frame(matrix(NA, nrow = S, ncol = (length(df$value) +
    m + max(aa_model$arma[1], aa_model$arma[2]))))

arima_star <- data.frame(matrix(NA, nrow = S, ncol = (length(df$value) +
    m + max(aa_model$arma[1], aa_model$arma[2]))))


for (i in 1:S) {

    e_sample[i] <- sample(aa_model$residuals, replace = T, size = (length(df$value) +
        m))

}

for (i in 1:S) {

for (j in ((aa_model$arma[1] + aa_model$arma[2] + 1):(length(df$value) +
        m))) {

        arima_star[i, j] <- (aa_model$coef[4] + (aa_model$coef[1] *
            e_sample[i, j - 1]) + (aa_model$coef[2] * e_sample[i,
            j - 2]) + (aa_model$coef[3] * e_sample[i, j - 3]) +
            e_sample[i, j])

    }

}


y_fixed <- data.frame(matrix(NA, nrow = S, ncol = (aa_model$arma[1] +
    aa_model$arma[2])))

for (i in 1:S) {
    y_fixed[i, 1] <- data.frame(df$value[1])
    y_fixed[i, 2] <- data.frame(df$value[2])
    y_fixed[i, 3] <- data.frame(df$value[3])
}


y_star <- data.frame(y_fixed, arima_star[, -(1:3)])

y_m <- y_star[, -(1:100)]

y_m <- y_m[, -(101:103)]

y_mt <- t(y_m)

y_matrix <- as.matrix(y_m)
```

```
fc_list <- vector("list", S)

for (i in 1:S) {

    fc_list[[i]] <- forecast(ts(y_matrix[i, ]), model = aa_model,
h = 5)
```

```
}

fc_list[[1]]
```

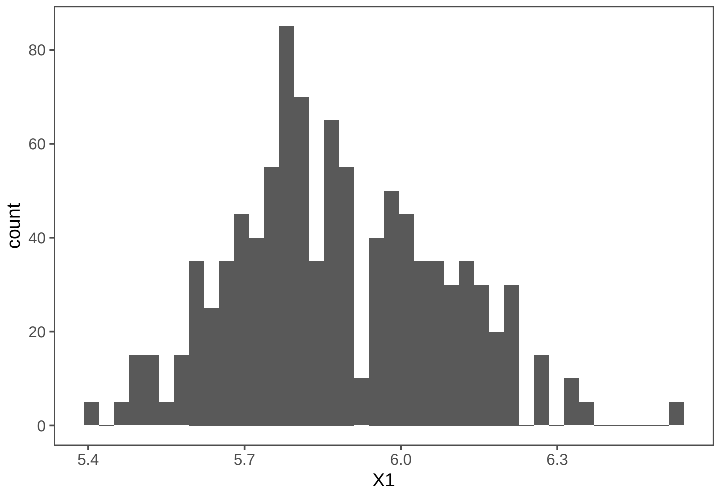
```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 101       5.880900 5.549926 6.211875 5.374719 6.387082
## 102       5.869038 5.532663 6.205413 5.354597 6.383479
## 103       5.841494 5.439567 6.243421 5.226800 6.456189
## 104       5.898184 5.475000 6.321368 5.250980 6.545387
## 105       5.898184 5.475000 6.321368 5.250980 6.545387
```

```
fc_mean <- data.frame(matrix(NA, nrow = S, ncol = 5))

for (i in 1:S) {

    fc_mean[i, ] <- fc_list[[i]]$mean

}
```

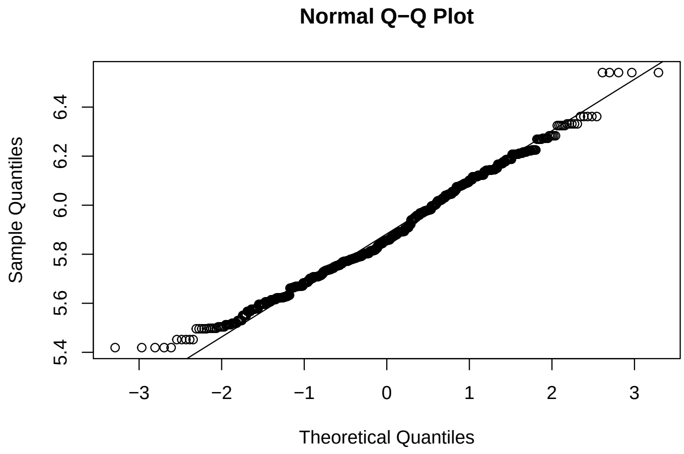
```
head(fc_mean)
```

```
##         X1       X2       X3       X4       X5
## 1 5.880900 5.869038 5.841494 5.898184 5.898184
## 2 5.734428 5.543250 5.786803 5.898184 5.898184
## 3 5.728049 5.722659 5.832225 5.898184 5.898184
## 4 5.844103 5.943213 5.958997 5.898184 5.898184
## 5 5.550780 5.518844 5.732659 5.898184 5.898184
## 6 5.742853 5.863462 5.848949 5.898184 5.898184
```

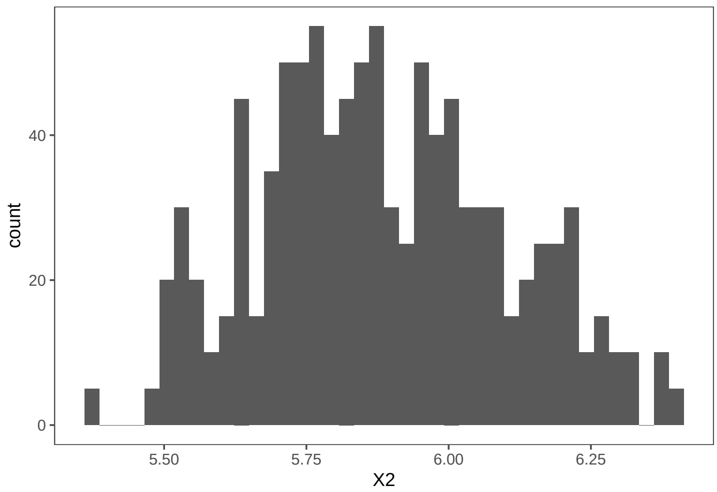
```
hist_x1 <- ggplot(data = fc_mean, aes(x = X1)) + geom_histogram(bins = 40) +
    theme_few()
hist_x1
```



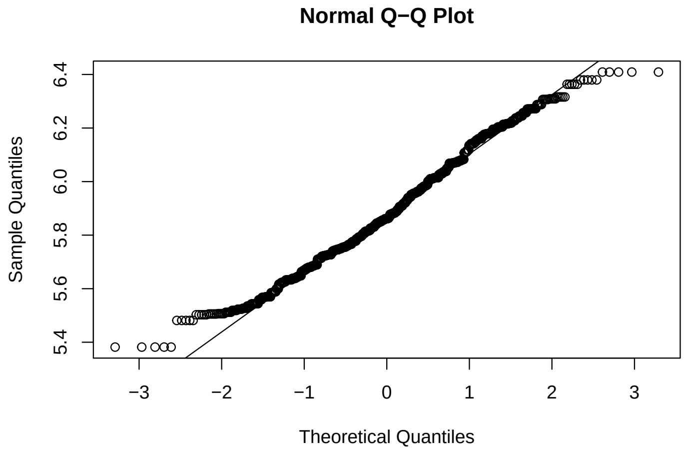

```
qq_x1 <- qqnorm(fc_mean$X1)
qqline(fc_mean$X1)
```



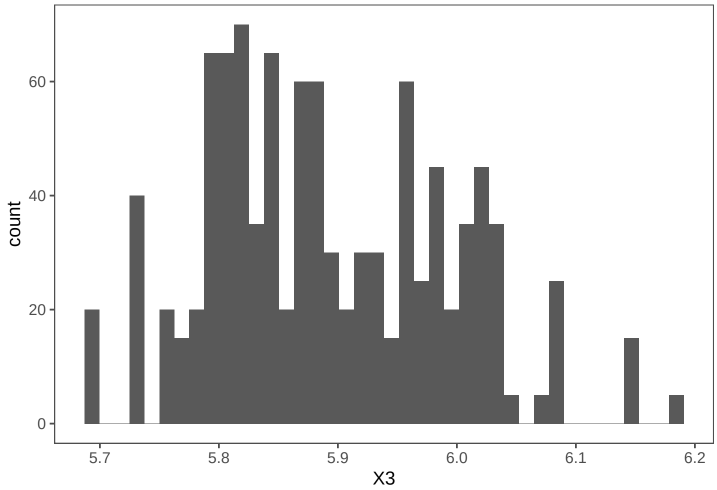

```r
hist_x2 <- ggplot(data = fc_mean, aes(x = X2)) + geom_histogram(bins = 40) +
    theme_few()
hist_x2
```



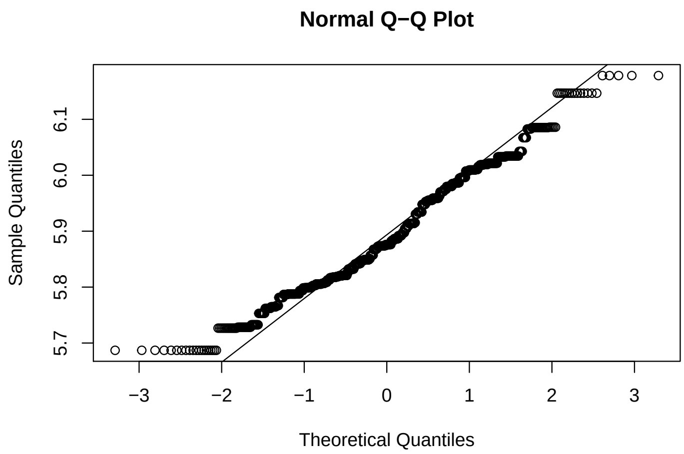

```r
qq_x2 <- qqnorm(fc_mean$X2)
qqline(fc_mean$X2)
```



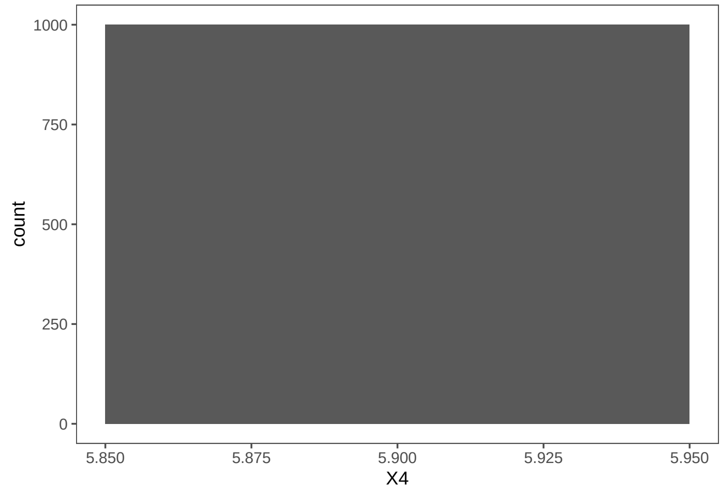
```r
hist_x3 <- ggplot(data = fc_mean, aes(x = X3)) + geom_histogram(bins = 40) +
    theme_few()
hist_x3
```



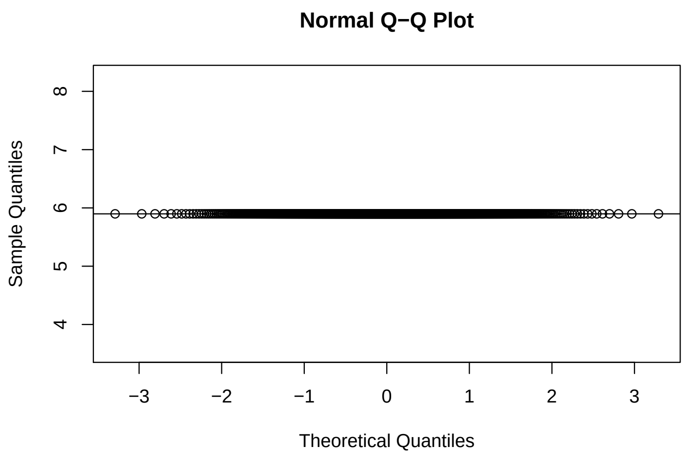

```r
qq_x3 <- qqnorm(fc_mean$X3)
qqline(fc_mean$X3)
```



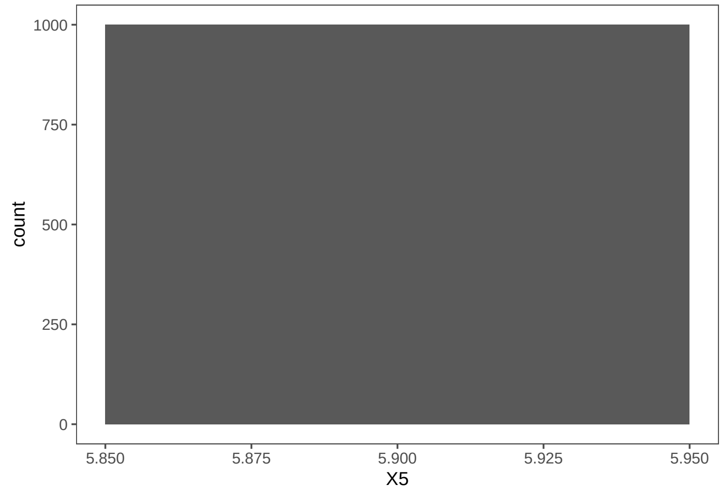

```r
hist_x4 <- ggplot(data = fc_mean, aes(x = X4)) + geom_histogram(bins = 40) +
    theme_few()
hist_x4
```



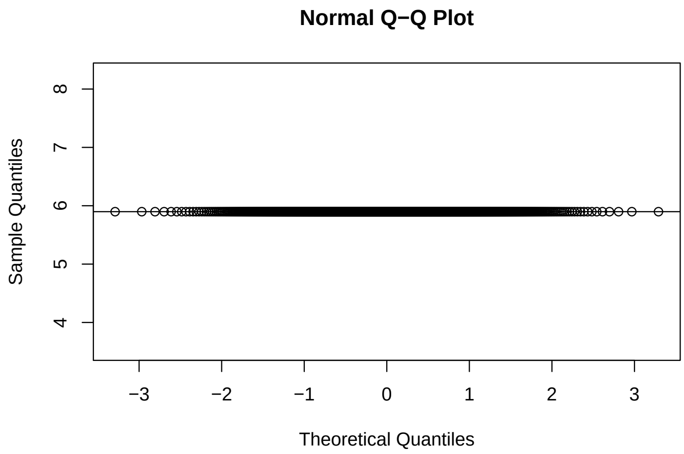

```
qq_x4 <- qqnorm(fc_mean$X4)
qqline(fc_mean$X4)
```




```
hist_x5 <- ggplot(data = fc_mean, aes(x = X5)) + geom_histogram(bins = 100) +
    theme_few()
hist_x5
```



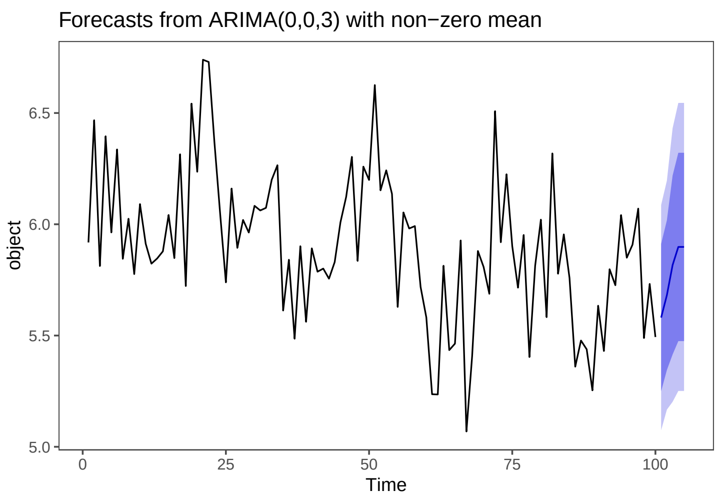
```
qq_x5 <- qqnorm(fc_mean$X5)
qqline(fc_mean$X5)
```



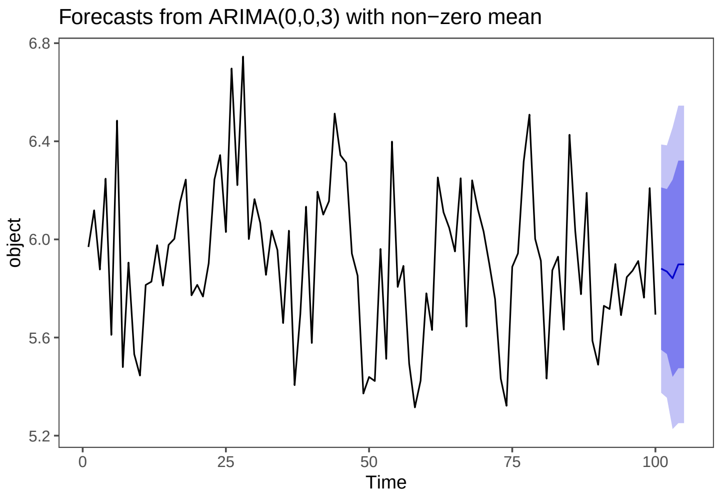
The results show that, from $h \geq 4$, the predicted value is the mean of the series.

Now, some forecasting plots:

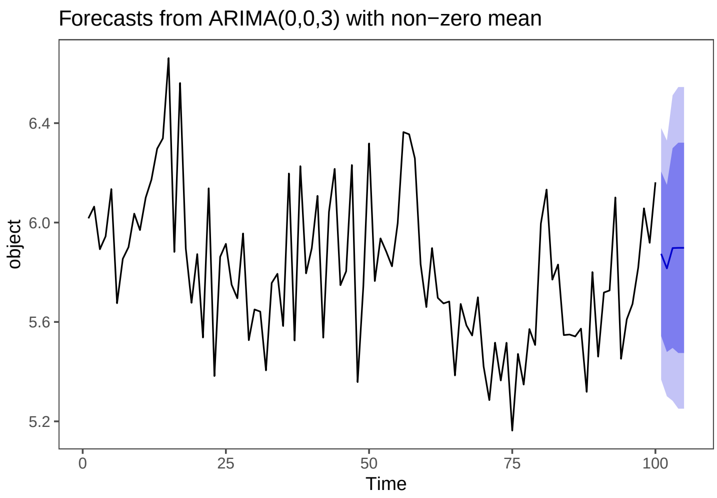
```
fc <- forecast(df$value, model = aa_model, h = h)

autoplot(fc) + theme_few()
```




```
autoplot(fc_list[[1]]) + theme_few()
```




```
autoplot(fc_list[[66]]) + theme_few()
```




```
autoplot(fc_list[[796]]) + theme_few()
```