**Deep understanding of the ARIMA model**

**Explore the features of time series — stationarity, stability, autocorrelation**



Image from [Unsplash](https://unsplash.com/photos/rmM8V7L1BhM)

**A quick introduction to time series**

Generally, a model for time-series forecasting can be written as

https://miro.medium.com/max/318/1*6xXgZwP-9NUD70j02RQonQ.png

Eq 0.2 Definition of the time-series forecasting model

where yₜ is the variables to be forecasted ([dependent variable](https://en.wikipedia.org/wiki/Dependent_and_independent_variables), or response variable), t is the time at which the forecast is made, h is the forecast horizon, Xₜ is the variables used at time t to make forecast ([independent variable](https://en.wikipedia.org/wiki/Dependent_and_independent_variables)), θ is a vector of parameters in function *g*, and εₜ₊ₕ denotes errors. It is worth noting that the observed data is *uniquely orderly according to the time of observation*, but it doesn’t have to be dependent on time, i.e. time (index of the observations) doesn’t have to be one of the independent variables.

**Some possible properties of time series**

**Stationarity**: a stationary process is *a stochastic process, whose mean, variance and autocorrelation structure do not change over time.* It can also be defined formally using mathematical terms, but in this article, it’s not necessary. Intuitively, if a time series is stationary, we look at some parts of them, they should be very similar — the time series is flat looking and the shape doesn’t depend on the shift of time. (A quick check of knowledge: is f(t) = sin(t) a stationary process? It surely isn’t, since it’s not stochastic, stationarity is not one of its properties)

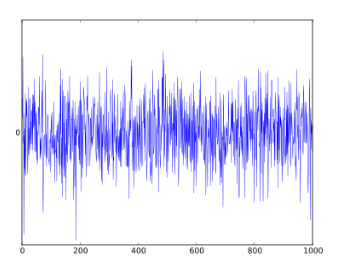


Fig 1.1 Example of stationarity. Image from [Wikipedia(White noise)](https://www.wikiwand.com/en/White_noise)

Figure 1.1 shows the simplest example of a stationary process — white noise.

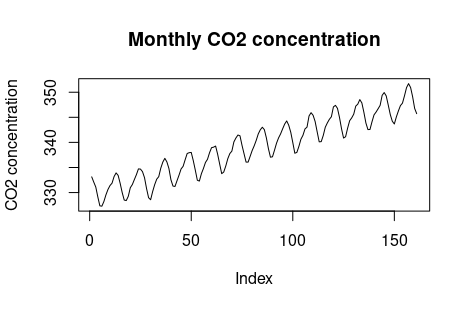


Fig 1.2 Example of non-stationarity time series. Graph made by the author.

The above image Figure 1.2 shows a non-stationary time series. Why is it so? We can see the obvious trend, it means that the variance changes over time. But if we use linear regression to fit a line to it (to capture the trend) and remove the trend the data now has a constant location and variance, but it’s still not stationary because of periodic behavior, which is not stochastic.

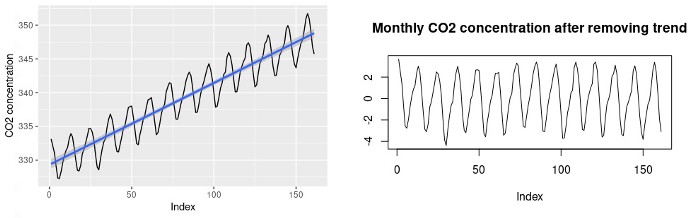


Fig 1.3 (left) Fit a line to the original data, Fig1.4 (right) Result after removing the trend. Graph made by the author

When using ARMA to model a time series, one of the assumptions is that the data is stationary.

**Seasonality**: Seasonality is *the property of showing certain variations in a specific time interval that is shorter than a year* (it can be over a different period of course. If we are observing the hourly temperature in a day and collect data over a couple of days, then the period is a day and it can also have seasonality — peaks might appear at 2 or 3 pm. This means we don’t have to interpret *season* in the context of seasonality using common sense), monthly, quarterly, etc. A very typical example of seasonal time series is electricity usage, in summer the electricity usage is usually higher because of, for instance, air conditioners.

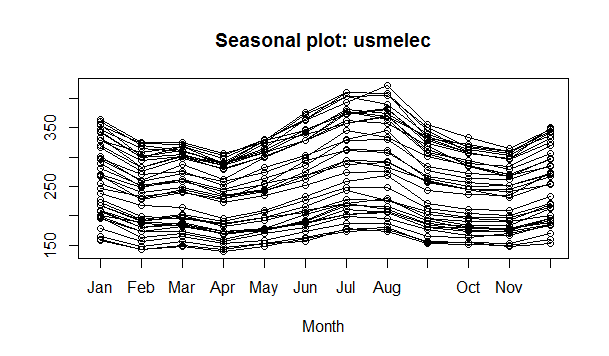


Fig. 1.5 A seasonality plot of US electricity usage. Image from [Wikipedia(Seasonality)](https://en.wikipedia.org/wiki/Seasonality?oldformat=true)

Figure 1.5 shows the data with seasonality properties, we can easily see that there is a peak of electricity usage in July and August, and a smaller peak in January and December. The data plotted in figure 1.5 is the electricity usage in the US from the year 1973 to 2010. usmelec is built-in data set in R. The electricity usage throughout the whole period looks like this

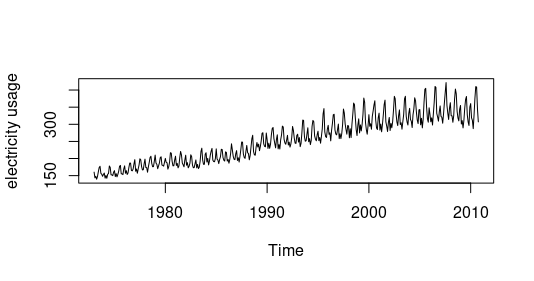


Fig 1.6 US electricity usage from 1973 to 2010. Graph made by the author

The presence of seasonality requires us to adjust the way of making predictions. For example, if we want to sell air conditioners, we need to predict the monthly sales in the future using the sales in the same season in the past years, instead of the closest months.

**Definition of ARIMA step by step**

**A technical note: Backshift operator**

*What it is?* It is an operator which shifts a variable xₜ backward, which is denoted sometimes as B (Backshift) and sometimes L (Lag), in this article, we will adopt the notation B. It’s defined as

https://miro.medium.com/max/115/1*H1vggHd0FIao6393GJJTwA.png

Eq 2.1 Definition of the backshift operator

and the forward-shift operator B⁻¹, which satisfies B⁻¹B = 1.

*Why do we need this?* It’s because it enables us to expressly backshifting in a succinct way — in terms of polynomials, which also helps us with defining more complicated models. When we use something to represent something in math, it’s always important to look at what operations it supports. In this case, we can easily see that the back-shift operator allows all the arithmetic operations on it: addition, subtraction, multiplication, division, exponentiation, etc.

It means, e.g. using the *operator* (1-B) on xₜ gets us the difference xₜ-xₜ₋₁, if we want to take the difference of xₜ and xₜ₋₁ again, we use *operator* (1-B)² on xₜ, which gives us (1–2B+B²)xₜ = xₜ-2xₜ₋₁+ xₜ₋₂. This is the same as (xₜ-xₜ₋₁)-(xₜ₋₁-xₜ₋₂). In this example, we encountered first-order and second-order differencing (Equation 2.12, 2.13), which will be explained later.

**Autoregressive (AR) model**

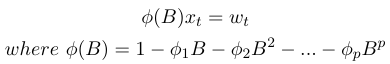
An autoregressive model of order p, abbreviated **AP(p), models the current value as a linear combination of the previous p values**. That’s how this model gets its name, it is a linear regression with itself. We see a lot of terms in the definition, but the model is still concerned to be *univariate* since the current value depends on its past values. Those values are of the same variable taken at different time points, so after all only one variable is concerned. A more generalized model is *VAR* (Vector autoregression), which allows multivariate times series. Formally, a (univariant) autoregressive model is defined as

https://miro.medium.com/max/433/1*UOJnJvmpf-AsfKDuvlvR_g.png

Eq. 2.2 The formal definition of the autoregressive model

where wₜ ~ wn(0, σᵥᵥ²), and ϕ₁, ϕ₂,…, ϕp (ϕp ≠ 0) are parameters. wn denotes “white noise”, it has normal distribution with mean 0 and variance σᵥᵥ². Sometimes there is also a constant on the right side of Equation 2.2, denoted by c ([*Forecasting: Principles and Practice chap 8.3*](https://otexts.com/fpp2/AR.html)). The role that the constant plays will be explained in the subsection of **ARIMA**.

In terms of the back-shift operator, the autoregressive model is defined as



Eq 2.3 Definition of AR model using back-shift operator

**Moving-average (MV) model**

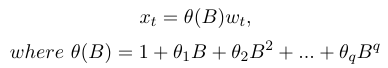
The moving average model of order q, abbreviated as **MA(q), models the current value as a linear combination of the previous q error terms** (unlike the autoregression model, in moving average, it is the *error term*, which is concerned). Formally, it’s defined as

https://miro.medium.com/max/482/1*1-BZnlwqwFKEcV7pRYNN_Q.png

Eq. 2.4 The formal definition of the moving average model

where wₜ ~ wn(0, σᵥᵥ²), and θ₁, θ₂,…, θq (θq ≠ 0) are parameters. wn denotes “white noise”, it has normal distribution with mean 0 and variance σᵥᵥ², like in the definition of the autoregressive model. This is a univariate model as well. A constant can also be added to the definition ([*Forecasting: Principles and Practice chap 8.4*](https://otexts.com/fpp2/MA.html)).

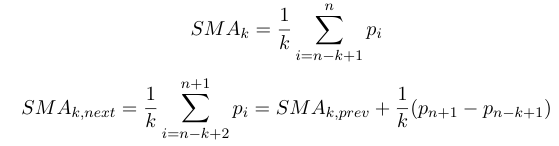
In terms of the back-shift operator, the moving average model can be defined as



Eq 2.5 Definition of MA model using back-shift operator

Unlike the autoregressive model, the name of this moving-average model is not so obvious. According to the footnote on page 48 in *Forecasting with Univariate Box‐Jenkins Models (2009)*: Concepts and Cases by Pankratz: *“the label ‘moving average’ is technically incorrect since the MA coefficients may be negative and may not sum to unity. This label is used by convention.”*

Another confusing thing is a concept that looks like the moving-average model, the *moving average*, a.k.a. rolling average or running average, which is used to *smooth out a time series*. In fact, it is completely a different tool — it is not used for prediction. We use an example of the simple moving average to make this clear



Eq 2.6, 2.7 Simple moving average

After observing the formulas, we can see that we need the k-th value to compute the k-th moving average.

**Autoregressive-moving-average (ARMA) model**

ARMA model is **the combination of AR and MA**, which is quite self-explanatory. ARMA takes into consideration both the past values and past error terms and describes a (weakly) stationary stochastic process in terms of two polynomials. Formally a time series is **ARMA(p, q)** if it is stationary and

https://miro.medium.com/max/596/1*Jz3Eu9lzv5WqGikbLv9Tkw.png

Eq 2.8 Formal definitional of ARMA

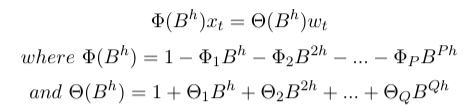
where ϕp ≠ 0, θq ≠ 0, wₜ ~ wn(0, σᵥᵥ²), and σᵥᵥ² > 0. The parameters p and q are the autoregressive and the moving average orders respectively, as mentioned before. In terms of the back-shift operator, the ARIMA model can be written as

https://miro.medium.com/max/161/1*2Nz1SL74FZpvRHW8xC4ChQ.png

Eq 2.9 The ARIMA model defined in terms of back-shift operator

This rewriting is not trivial. It reveals a serious problem that can occur in the model — *the redundancy of parameters*. If the polynomials ϕ(B) = 0 and θ(B) = 0 have common factors, then the models will contain redundant parameters. It will make the model uselessly complicated. When can this situation occur? When we try to fit a white noise series (xₜ = wₜ) using an ARMA(1,1) model, the program will do it, but the model we get will have superfluous parameters. Therefore, we need to remove the redundancy to simplify the model, and the redundancy can be removed using covariance analysis.

The definition shown in Equation 2.8 is the *non-seasonal* ARMA. But it often happens that the data is seasonal. What should we do, if we want to remove seasonality? The answer is to introduce a lag *h (length of the seasonal period)*, which brings us to the seasonal ARMA (SARMA), denoted as ARMA(P, Q)ₕ, and is of the form



Eq 2.10 definition of the seasonal ARMA

This method of removing the seasonal effect corresponds to what we have described before: *using Augusts’ data to predict August’s sales*. What should h equal to? This depends on the frequency of the seasonality, for example, if the seasonal variation appears once in a year in some specific months, then h = 12, if it appears once in each quarter of the year, then h = 4. If we combine seasonal ARMA with non-seasonal ARMA (multiply the seasonal and non-season operators together), we will get one layer of generalization — mixed season ARMA.

https://miro.medium.com/max/277/1*1T4ZS5Deojjg6qDC-t-Lkw.png

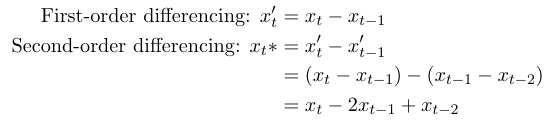
Eq 2.11 The model of mixed seasonal ARMA

https://miro.medium.com/max/211/1*Qp7jDyf0srRxvXEZUBcQvg.png

Notation of mixed seasonal ARMA

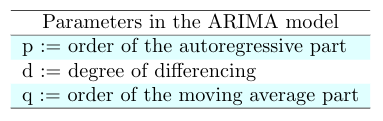
**Autoregressive integrated moving average (ARIMA) model**

ARIMA model is **ARMA modeled on a differenced series,** the **differencing** is sometimes denoted as **▽**. What is differencing then? It is a technique of removing the non-stationary of a series (this removes the non-constant trend, which means it only makes the mean stationery, but not variance). It takes the difference between two observations.



Eq 2.12, Eq 2.13 differencing

Of course, we can difference the observations multiple times. Eq 2.12 and Eq 2.13 show the example of first-order differencing and second-order differencing. It is obvious how the differencing differs from differentiation — differencing just *takes the difference*, meanwhile, differentiation *calculates the rate of change*. ARIMA model is usually denoted by ARIMA(p, d, q), the meaning of the parameters are summarized in the following table



Tab. 2.1 Parameters in the ARIMA model. (made by the author)

Now it’s time to introduce the formal definition of ARIMA model,

https://miro.medium.com/max/491/1*CAgvezF7cfwJE-5-1R2rJw.png

Eq 2.14 The definition of the ARIMA model

where yₜ’ denoted the difference series, other parameters are defined in the same way as those in the ARMA model. As before, a constant can be added to the model, which denotes the *drift*. It can be easily understood via an example with an ARIMA(0, 1, 0) model (no autoregressive nor moving-average terms, modeled using first-degree difference) involved:

Without parameter: the model is xₜ = xₜ₋₁ + εₜ, which is a random walk.

With parameter: the model is xₜ = c+ xₜ₋₁ + εₜ. This is a random walk with drift.

*The constant adds a non-constant trend to the process.* (This sentence looks very weird, but we need to be aware that the formula xₜ = c+ xₜ₋₁+ εₜ is recursive and in each unwinding a constant in the formula with stack up) How much influence does this constant have? A lot. We can see this from the simulation of two random walks, one with and one without drift

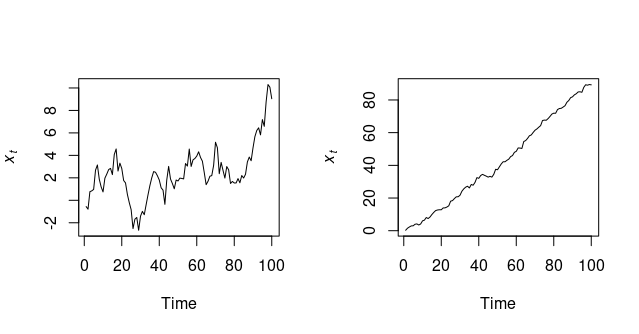


Fig 2.1 Simulation of random walk without drift (left) and with drift (right). Figures made by the author.

However, this differencing doesn’t take care of seasonality. To remove seasonality, we need to take the seasonal difference (the difference between xₜ and xₜ₋ₕ), and this brings us to the seasonal autoregressive integrated moving average (SARIMA) model. The relation between SARIMA and SARMA is very similar to the relation between ARIMA and ARMA — SARIMA is SARMA with differencing, but now we need to take not only the non-seasonal difference but also seasonal difference. We use **▽**ᴰ and **▽**ᵈ to denote the seasonal and non-seasonal differences, respectively. *D* is the degree of seasonal differencing.

https://miro.medium.com/max/304/1*-zw9lV61EqUAxcgTooB3Mg.png

Eq. 2.15 The model of seasonal ARIMA

https://miro.medium.com/max/265/1*kAb8DmWxRCsO3Xqi_n3uBg.png

Notation of seasonal ARIMA

**ARIMA in practice**

As stated in the bible book *Forecasting: Principles and Practices*, there is a [general approach](https://otexts.com/fpp2/arima-r.html#modelling-procedure) of fitting an ARIMA model: preprocess, until the data become stationary; feed to a function, which computes ARIMA model; compare the models; check the results (the residuals); if not good enough, iterate, otherwise use the result model to do forecast.

In R, the parameters are estimated using *maximum likelihood estimation* (MLE), which maximizes a likelihood function. The likelihood is equal to the probability that an observation x is produced, given the parameters (ϕ, θ, …) of our model. To compare the models, the *Akaike information criterion* (AIC) is used, which evaluates the loss of information and also penalizes the complication of models (amount of estimated parameters). We choose the model with the least AIC value.

**Automated**

If we want to use an automated process to build to model, the function auto.arima is at our disposal. It uses a step-wise procedure (Hyndman-Khandakar algorithm) to traverse the space of models efficiently. The function auto.arima takes care of differencing the data to make the data stationary (whether d = 0), choosing hyperparameters, and selecting the best model according to AIC.

We use oil prices from the 16th of August last year to 26th August this year to show the automated ARIMA process. What we want to achieve is to use the data from 16 Aug. 2020 to 16 Aug.2021 to predict the oil price in the following 10 days, then compare the result against the real values. (In this article the purpose is mainly to introduce the reader to the principle of the ARIMA model, so this is really a rough stock prediction, and things like back-testing are not included)

# read data  
# Datasource: <https://finance.yahoo.com/quote/OIL/history?p=OIL>  
dataOil <- read.csv2(file = "data/OIL.csv", sep=",")  
head(dataOil)# split into training and test data  
lenOil <- nrow(dataOil)  
trainSize <- ceiling(lenOil-10)# take the first 12 month   
train.oil <- dataOil[1:trainSize, ]  
test.oil <- slice\_tail(dataOil, n=lenOil-trainSize)# convert to time series  
train.close <- train.oil$Close  
train.close <- as.numeric(train.close)# frequency is set to be one because we have only one year of data  
train.close.ts <- ts(data=train.close, frequency=1)# plot the training data   
plot(train.close, type='l', ylab='Close price', main="One year oil price")

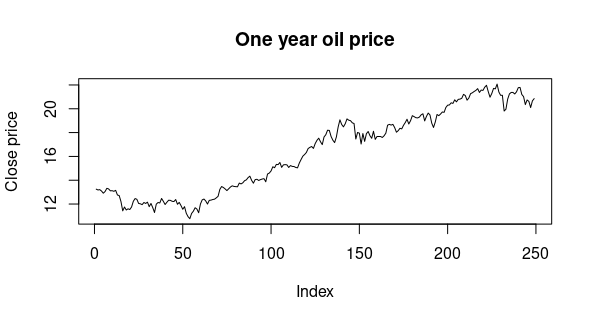


Fig 3.1 Oil price from last 26 Aug. 2020 to 26 Aug. 2021

Obviously, we can see a trend, and the time series is not stationary, but there is no evidence of varying variance, so we are not going to do any transformation. The rest of the model building is taken care of by auto.arima. But what we need to do is to check the residuals, to make sure that they look like random noise, this part is done by checkresiduals, which applies Ljung-Box test. If the p-value of *the Ljung-Box test* is larger than 0.05, we **can’t reject** the null hypothesis that *the residuals are distributed independently*. Therefore, this model is admissible, otherwise, we would need to choose a new one.

start <- length(train.close.ts)  
test.index <- seq(start, start+9)  
test.close <- as.numeric(test.oil$Close)aa.train <- auto.arima(train.close.ts)   
# --> AIC = 157.11  
checkresiduals(aa.train)  
# --> p-value = 0.07454plot(forecast(aa.train, h=10))  
lines(test.index, test.close)

The result looks like this, compares with the real values of the oil price from 17 Aug. to 26 Aug (the last 10 days).

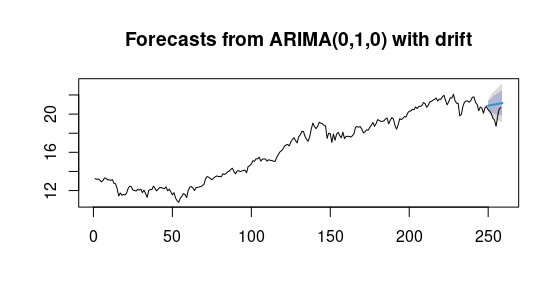


Fig 3.2 The result of the forecasting

**Customer model**

The automated process can fail to find a suitable model for us. If it fails the Ljung-Box test, we need to choose the hyperparameters ourselves. Therefore, sometimes we might need to choose our own model (using Arima). Compared to the automated process, what do we need to do *in addition* to use our own model? Firstly, after observing the data, we need to investigate the partial autocorrelation of the data. (to know more about partial correlation [click here](https://towardsdatascience.com/partial-correlation-508353cd8b5)).

# plot the partial difference   
dif.train <- difference(train.close.ts)  
ggtsdisplay(dif.train, plot.type='partial')

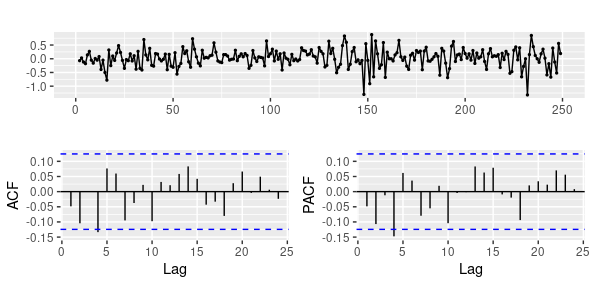


Fig 3.3 First-order difference, autocorrelation, and partial autocorrelation

The difference shown in the graph looks quite stationary, therefore we will just use the first-order difference. But we can still perform a unit root test to check the stationarity. We have several alternatives for this, here we use the Augmented Dickey-Fuller test (ADF, in R it’s adf.test, if we set k=0, then it becomes simply Dickey-Fuller test).

# ADF test  
dif.train <- na.remove(dif.train)  
adf.test(dif.train)  
# --> p-value smaller than 0.01

According to the result, we can **reject** the null hypothesis that *the first-order difference is* ***non-stationary***. Therefore we set **d=1**. According to the partial autocovariance, which shows *the correlation between a stationary time series and its own lagged values with the influence of shorter lags removed*, variable xₜ is correlated with xₜ₊₄, then we can try to set **p = 4**. The rest is **q**, we can try from 0 and gradually increase it. As to seasonality, it’s not really shown in the graph, therefore we can leave the parameters of **seasonality** (P, D, Q) as default zeros. Should we add drift? From the increasing trend in Figure 3.2, seems to be yes, we can set include.drift to be true.

# manually try the different parameters  
fit.m1 <- Arima(train.close.ts, order=c(4,1,0), include.drift=TRUE)  
fit.m1  
# --> AIC = 156.15fit.m2 <- Arima(train.close.ts, order=c(4,1,1), include.drift=TRUE)  
fit.m2  
# --> AIC = 157.61   
# after trying some more other values of q, we will find out  
# that increasing q doesn't really decrease the AICcheckresiduals(fit.m1)  
# --> p-value = 0.4187plot(forecast(fit.m1, h=10))  
lines(test.index, test.close)

Our custom model has a slightly lower AIC value, yet the plot looks almost the same as the one shown in Figure 3.2.

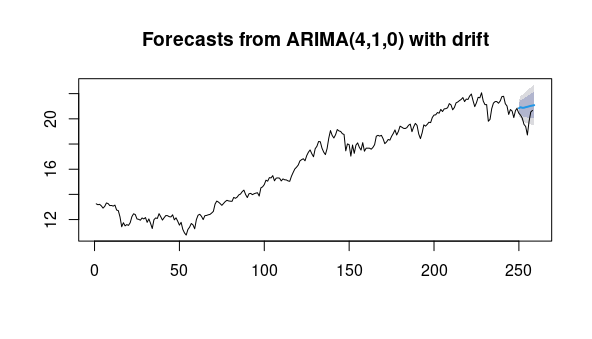


Fig 3.4 The result of the forecasting-custom model

*Summary*

In this article, at first, we quickly introduced the formal definition of time series and some typical properties which can occur in a time series. After that, we get familiar with the definition of ARIMA gradually, along with the extension of ARMA and ARIMA — seasonal ARMA (SARMA) and seasonal ARIMA (SARIMA). At last, we build a model to do a short-term prediction of oil prices both automatically and manually.