To Difference or Not To Difference? Likelihood Ratio Test for Autoregressive Time Series with a Unit Root

12/05/2021

Set up

To read real-world data in the Simulation and Application part, please download the .csv file and store it under the same directory as this file.

Introduction

In time-series analysis, stationarity is the most important assumption about data collected over time. In traditional statistics, data is collected in an independent and identical way and all inferences are based on this fundamental assumption. In contrast, in time-series analysis, the independent assumption no longer holds. Since we get only one observation at each timestamp, we need stationarity in order to make inference on time-series data:

Definition of stationary time series:

```
   A stationary time series is a finite variance process such that
   1) the mean value is constant and does not depend on time t
   2) the covariance between any two observations, Y(s) and Y(t), depends on
s and t only through their difference |s-t|
```

Autoregressive (AR) model is widely used because of temporal correlation of time-series data. An AR(p) model can be written as

$$Y_t = \rho_1 Y_{t-1} + \rho_2 Y_{t-2} + \ldots + \rho_p Y_{t-p} + \epsilon_t$$

where ϵ_t is an IID noise process.

In practice, AR(1) model, a special case of AR(p) process, can well estiamte most time-indexed observations. An AR(1) process is represented as $Y_t = \rho Y_{t-1} + a_t$, or equivalently, $(1 - \rho B)Y(t) = a_t$, where $\phi(B) = 1 - \rho B$ is called **backshift function**, i.e., $(1 - \rho B)Y(t) = Y(t) - \rho Y(t-1)$.

If $\rho < 1$, then the AR(1) proess safiesties all the conditions of a stationary time series above. If $\rho > 1$, the AR(1) process has moving means and explosive variance. If $\rho = 1$, the AR(1) process is called a **random walk** and a **unit root** exists in this case.

Motivation

In this blog we are most interested in studying two underlying models: random walk with drift

$$Y_t = \alpha + \rho Y_{t-1} + \epsilon_t$$

and time-series with only a deterministic trend:

$$Y_t = \beta_0 + \beta_1 t + \epsilon_t$$

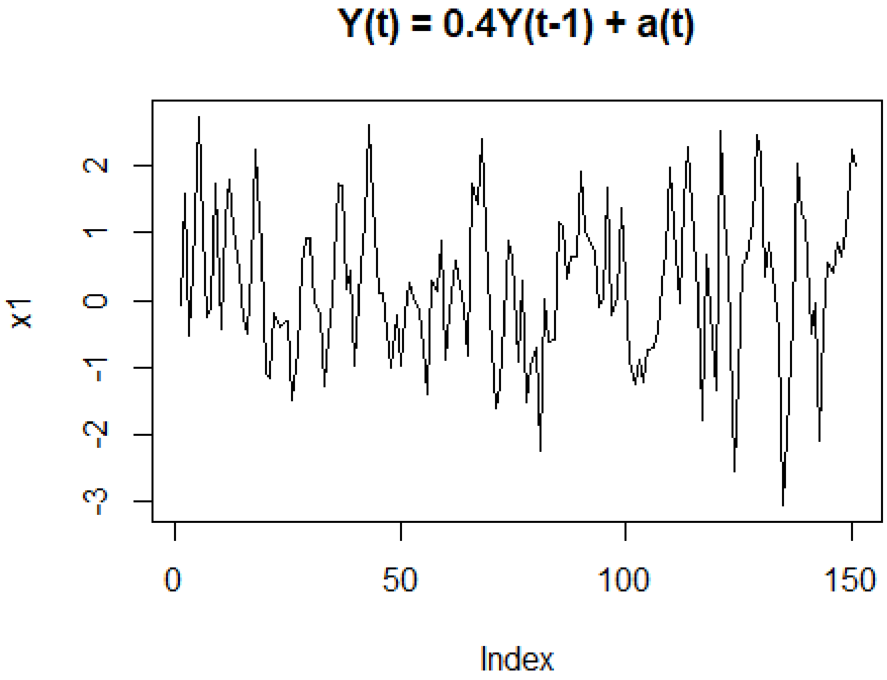
Note that the second model is just a linear regression model with time t as the covariate.

Let's start with why we are interested in these two models.

Stationary AR(1) process

Firstly, if data comes from stationary auto-regressive (1) (AR(1)) process ($Y(t) = 0.4 * Y_{t-1} + \epsilon_t$), it looks like this:

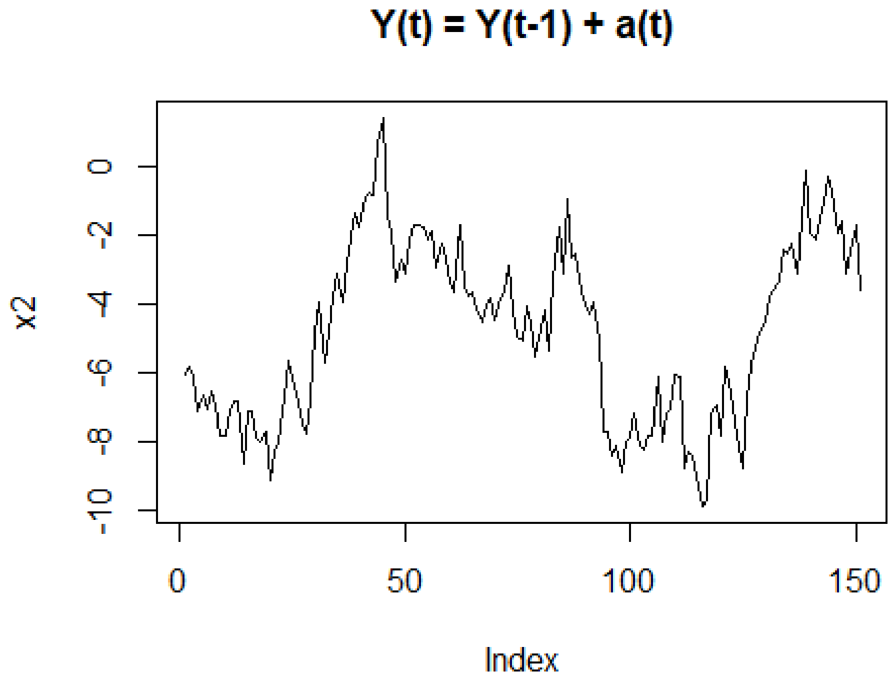
This time-series data oscillates around 0 and has stationary covariance. We can make all kinds of inference, like forecasting.



Random walk process

Now, if the ρ parameter above is 1, i.e., $Y(t) = Y_{t-1} + \epsilon_t$, the plot looks like this:

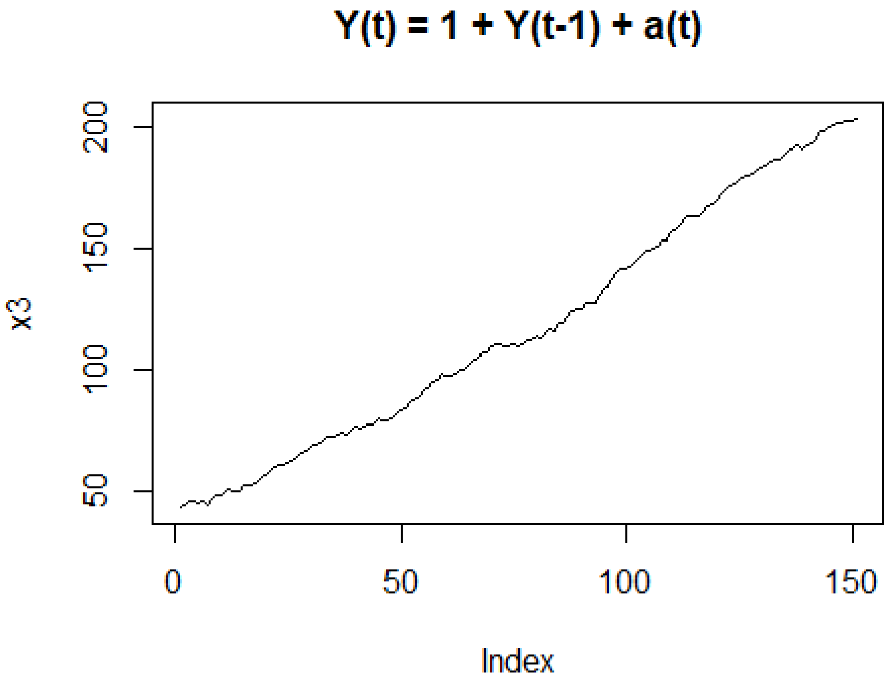
In this scenario, stationary variance no longer holds and it seems like there are some trends present in the data.



Random walk with drift

Next, if the underlying model is a random walk with drift ($Y_t = 1 + Y_{t-1} + \epsilon_t$), it looks like this:

Even though both models are AR(1) process and ρ parameter is the same, their behavior is totally different.



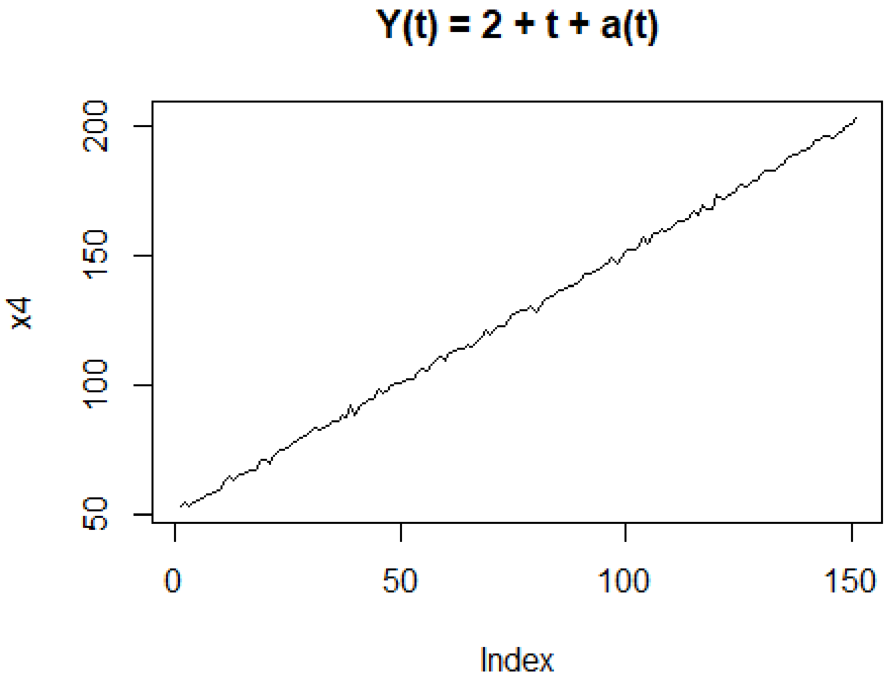
Deterministic trend

Finally and most interestingly, if the underlying model only have a deterministic trend

$$Y_t = \beta_0 + \beta_1 t + \epsilon_t$$

it looks like this:

Even though the plot looks very similar to the last one, the underlying model is totally different: it doesn't even have an autoregressive component.



In time-series analysis, if the underlying model is $Y(t) = Y(t-1) + \epsilon_t$, the first step is *differencing* the data, i.e., $Y(t)^* = Y(t) - Y(t-1)$. If data comes from $Y_t = \beta_0 + \beta_1 t + \epsilon_t$, we only need to do normal regression analysis without considering further temporal correlation. But how can we distinguish the true models? To difference or not to difference? This is where unit root test comes in. In the most basic version of unit root test, for an AR(1) model $Y(t) = \rho Y(t-1) + \epsilon_t$, the hypothesis testing is $H_0: \rho = 1$ v.s. $H_A: \rho \neq 1$. An augmented version of this test, called Augmented Dickey-Fuller test, allows for richer testing ideas such as unit root with a drift, but the basic idea holds the same.

Derivation of parameter estimate and testing statistics

In linear model setting $Y = X\beta + U$ where $U \sim N(0, \sigma^2 I)$, n is the number of observations, p is the number of predictors and $r = rank(X) \leq p$.

The convex function $Q(\beta)$ reaches its minimum when $\nabla_\beta Q(\beta) = 0$ which yields to the normal equation: $X^T X\beta = X^T Y$. Solving this equation gives us:

Simple linear regression model is defined as : $y_i = \beta_0 + \beta_1 x_i + u_i$ where $i = 1, \cdots, n$.

Then we have :

$$X^T X\beta = \begin{pmatrix} n & \sum_{i=1} n\, x_i \\ \sum_{i=1} n\, x_i & \sum_{i=1} n\, {x_i}^2 \end{pmatrix} \times \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$$

$$X^T Y = \begin{pmatrix} \sum_{i=1} n\, y_i \\ \sum_{i=1} n\, x_i y_i \end{pmatrix}$$

We can solve the normal equations: $X^T X\beta = X^T Y$

$$\begin{cases} \widehat{\beta_0} = \overline{y_n} - \overline{x_n}\widehat{\beta_1} \\ \widehat{\beta_1} = \dfrac{\sum_{i=1}^n (x_i - \overline{x_n}) y_i}{\sum_{i=1}^n (x_i - \overline{x_n})^2} = \dfrac{\sum_{i=1}^n (x_i - \overline{x_n})(y_i - \overline{y_n})}{\sum_{i=1}^n (x_i - \overline{x_n})^2} \end{cases} (3)$$

We can find the variance of the estimates:

$$\begin{cases} \mathrm{Var}(\widehat{\beta_1}) = \mathrm{Var}(\dfrac{\sum_{i=1}^n (x_i - \overline{x_n}) y_i}{\sum_{i=1}^n (x_i - \overline{x_n})^2}) = \dfrac{\sigma^2}{\sum_{i=1}^n (x_i - \overline{x_n})^2} \\ \mathrm{Var}(\widehat{\beta_0}) = \mathrm{Var}(\overline{y_n} - \overline{x_n}\widehat{\beta_1}) = \sigma^2 (\dfrac{1}{n} + \dfrac{\overline{{x_n}^2}}{\sum_{i=1}^n (x_i - \overline{x_n})^2}) \end{cases} \quad (4)$$

In general,

$$\mathrm{Var}(\hat{\beta}) = \mathrm{Var}((X^T X)^{-1} X^T Y) = (X^T X)^{-1} X^T \mathrm{Var}(y) X (X^T X)^{-1} = \sigma^2 (X^T X)^{-1} \ (5)$$

t-test statistics for $H_0: \beta_0 = 0$ is $\frac{\widehat{\beta} - \beta}{\mathrm{se}(\widehat{\beta})}$ (6). Notice that alternative hypothesis H_a is generalized as H_0 is not true for all the hypothesis testing from here on out as described in the paper from section 2.

We define the projection matrix $P = X(X^T X)^{-1} X$ and we can decompose y into two orthogonal pieces:

$$Y = X\hat{\beta} + Y - X\hat{\beta} = P_x Y + (I - P_x) Y$$

with $P_x Y = X\hat{\beta}$ and $(I - P_x) Y = Y - X\hat{\beta} = \hat{e}$, then the estimate of variance will be defined as follows:

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^n (Y - X\hat{\beta})^2}{n - r} = \frac{X^T (I - P_x) Y}{n - r} \ (7)$$

In AR(1) model, the equation 1.1 from the paper: $Y_t = \alpha + \rho Y_{t-1} + e_t$ where $t = 2,3,\cdots,n$ with fixed Y_1 and $e_t \sim N(0, \sigma^2 I)$. The matrix notation will be look like this:

where we have $n-1$ observations and rank(X)=r=2.

Because of (3), we have the parameter estimates:

$$\widehat{\alpha_\mu} = \overline{y}_{(0)} - \widehat{\rho_\mu}\overline{y}_{-1}$$

$$\widehat{\rho_\mu} = \frac{\sum_{t=2}^{n}(y_t - \overline{y}_{(0)})(y_t - \overline{y}_{(-1)})}{\sum_{t=2}^{n}(y_{t-1} - \overline{y}_{(-1)})^2}$$

where $\overline{y}_{(i)} = \frac{\sum_{t2}^{n} y_{t+i}}{n-1}$ for $i = -1, 0$.

Because of (7),

$$S_{e\mu}^2 = \hat{\sigma}^2 = \frac{\sum_{t=2}^{n}(y_t - \hat{\alpha}_\mu - \hat{\rho}_\mu y_{t-1})^2}{n-3}$$

Because of (4),

$$\text{Var}(\hat{\alpha}_\mu) = \hat{\sigma}^2(\frac{1}{n-1} + \frac{\overline{y_n}_{(-1)}^{\;2}}{\sum_{t=2}^{n}(y_{t-1} - \overline{y}_{(-1)})^2}) = S_{e\mu}{}^2(\frac{1}{n-1} + \frac{\overline{y_n}_{(-1)}^{\;2}}{\sum_{t=2}^{n}(y_{t-1} - \overline{y}_{(-1)})^2}) = S_{\alpha\mu}{}^2$$

Because of (6), “t-statistic” to test $\alpha = 0$ is: $\hat{\tau}_{\alpha\mu} = \frac{\hat{\alpha}_\mu - 0}{\text{se}(\hat{\alpha}_\mu)} = S_{\alpha\mu}^{-1}\hat{\alpha}_\mu$

For model 1.3 from the paper: $Y_t = \alpha + \beta(t - 1 - \frac{1}{2}n) + \rho Y_{t-1} + e_t$ where $t = 2,3,\cdots,n$ with fixed Y_1 and $e_t \sim N(0, \sigma^2 I)$. The matrix notation will be look like this:

where we have $n-1$ observations and rank(X)=r=3. Because of (1), estimates from 1.4 in the paper can be obtained as follows:

$$\hat{\theta}_\tau = \begin{pmatrix} \hat{\alpha}_\tau \\ \hat{\beta}_\tau \\ \hat{\rho}_\tau \end{pmatrix} = (X^T X)^{-1} X^T Y$$

Because of (5), $\text{Var}(\hat{\theta}_\tau) = \sigma^2 (X^T X)^{-1}$. We can use $\hat{\sigma}^2$ to estimate σ^2 if it’s unknown.

$$\text{Var}(\hat{\alpha}_\tau) = \hat{\sigma}^2 (X^T X)^{-1} = C_{11}\hat{\sigma}^2$$

and

$$\text{Var}(\hat{\beta}_\tau) = \hat{\sigma}^2 (X^T X)^{-1} = C_{22}\hat{\sigma}^2$$

where C_{11} and C_{22} represent the first two diagonal entries of the symmetric matrix $(X^TX)^{-1}$ and $\hat{\sigma}^2 = \frac{Y^T(I-P_x)Y}{(n-1)-3} = \frac{Y^T(I-X(X^TX)^{-1}X^T)Y}{n-4}$. So the "regression t statistic" when testing $\alpha = 0, \beta = 0$ from 1.5 can be found from the estimates below:

$$\hat{t}_{\alpha\tau} = \frac{\hat{\alpha}_\tau - 0}{\text{se}(\hat{\alpha}_\tau)} = (C_{11}S_{e\tau})^{-\frac{1}{2}}\hat{\alpha}_\tau$$

$$\hat{t}_{\beta\tau} = \frac{\hat{\beta}_\tau - 0}{\text{se}(\hat{\beta}_\tau)} = (C_{22}S_{e\tau})^{-\frac{1}{2}}\hat{\beta}_\tau$$

Then the log likelihood function ends up with the following form:

Under H_0: $(\alpha, \beta, \rho) = (0,0,1)$ the log likelihood is reduced to :

${\hat{\sigma}_0}^2$ can be found by setting the first derivative of convex function log likelihood to zero.

$$\frac{\partial l}{\partial \sigma^2} = -\frac{n-1}{2\sigma^2} + \frac{\sum_{t=2}^{n}(y_t - y_{t-1})^2}{-2\sigma^4}$$

$$(n-1)\sigma^2 = \sum_{t=2}^{n}(y_t - y_{t-1})^2$$

$${\hat{\sigma}_0}^2 = \frac{\sum_{t=2}^{n}(y_t - y_{t-1})^2}{n-1}$$

Under the alternative, the estimate of variance ${\hat{\sigma}_1}^2$ is just the average sum of squares (SSE):

$$\begin{aligned}
{\hat{\sigma}_1}^2 &= \frac{Y^T(I-P_x)y}{n-1} \\
&= (n-4)\frac{Y^T(I-P_x)y}{n-4}\frac{1}{n-1} \\
&= (n-4)\frac{{S_{e\tau}}^2}{n-1}
\end{aligned}$$

We know from linear model (2) that if we only test partial parameter $\beta_0 = 0$ then $T_{LR} = 2\{l(\hat{\beta}) - l(\tilde{\beta})\}$ will have the following form if σ^2 is unknown:

$$\begin{aligned}
T_{LR} &= 2\{-\frac{n}{2}\log{\hat{\sigma}_{full}}^2 - \frac{{\| Y - X\hat{\beta} \|_2}^2}{2{\hat{\sigma}_{full}}^2} + \frac{n}{2}\log{\hat{\sigma}_{reduced}}^2 - \frac{{\| Y - X\tilde{\beta} \|_2}^2}{2{\hat{\sigma}_{reduced}}^2}\} \\
&= 2\{-\frac{n}{2}\log{\hat{\sigma}_{full}}^2 - \frac{n}{2} + \frac{n}{2}\log{\hat{\sigma}_{reduced}}^2 + \frac{n}{2}\} \\
&= n\log\frac{{\hat{\sigma}_0}^2}{{\hat{\sigma}_1}^2}
\end{aligned}$$

where $\hat{\beta}$is the regular maximum likelihood estimator (MLE) and $\tilde{\beta}$ is the constrained MLE under the null hypothesis. The T_{LR} is just comparing two SSE. Therefore for 1.4:

$$\begin{aligned}
(\frac{\hat{\sigma}_0^2}{\hat{\sigma}_1^2})^{n-1} &= \left(\sqrt{\frac{\frac{\sum_{t=2}^{n}(y_t - y_{t-1})^2}{n-1}}{(n-4)\frac{S_{e\tau}{}^2}{n-1}}}\right)^{n-1} \\
&= \left(\frac{(n-1)\hat{\sigma}_0{}^2}{(n-4)S_{e\tau}{}^2}\right)^{\frac{n-1}{2}} \\
&= \left(\frac{3(n-4)S_{e\tau}{}^2 + 3((n-1)\hat{\sigma}_0{}^2 - (n-4)S_{e\tau}{}^2)}{3(n-4)S_{e\tau}{}^2}\right)^{\frac{n-1}{2}} \\
&= \left(1 + \frac{3}{n-4}\frac{(n-1)\hat{\sigma}_0{}^2 - (n-4)S_{e\tau}{}^2}{3S_{e\tau}{}^2}\right)^{\frac{n-1}{2}} \\
&= \left(1 + \frac{3}{n-4}\Phi_2\right)^{\frac{n-1}{2}}
\end{aligned}$$

where $\Phi_2 = \frac{(n-1)\hat{\sigma}_0{}^2 - (n-4)S_{e\tau}{}^2}{3S_{e\tau}{}^2}$.

For 1.3, The T_{LR} is equivalent to the following ratio:

$$\begin{aligned}
(\frac{\hat{\sigma}_0^2}{\hat{\sigma}_1^2})^{n-1} &= \left(\sqrt{\frac{\frac{\sum_{t=2}^{n}(y_t - y_{t-1})^2}{n-1}}{(n-3)\frac{S_{e\mu}{}^2}{n-1}}}\right)^{n-1} \\
&= \left(\frac{(n-1)\hat{\sigma}_0{}^2}{(n-3)S_{e\mu}{}^2}\right)^{\frac{n-1}{2}} \\
&= \left(\frac{2(n-3)S_{e\mu}{}^2 + 2((n-1)\hat{\sigma}_0{}^2 - (n-3)S_{e\mu}{}^2)}{2(n-3)S_{e\mu}{}^2}\right)^{\frac{n-1}{2}} \\
&= \left(1 + \frac{2}{n-3}\frac{(n-1)\hat{\sigma}_0{}^2 - (n-3)S_{e\mu}{}^2}{2S_{e\mu}{}^2}\right)^{\frac{n-1}{2}} \\
&= \left(1 + \frac{2}{n-3}\Phi_1\right)^{\frac{n-1}{2}}
\end{aligned}$$

where $\Phi_1 = \frac{(n-1)\hat{\sigma}_0{}^2 - (n-3)S_{e\mu}{}^2}{2S_{e\mu}{}^2}$.

Simulation and Application

We will apply the unit root test to real world data to demonstrate its power in time-series data analysis and to simulated data for completeness of reasoning.

SPY Stock Price

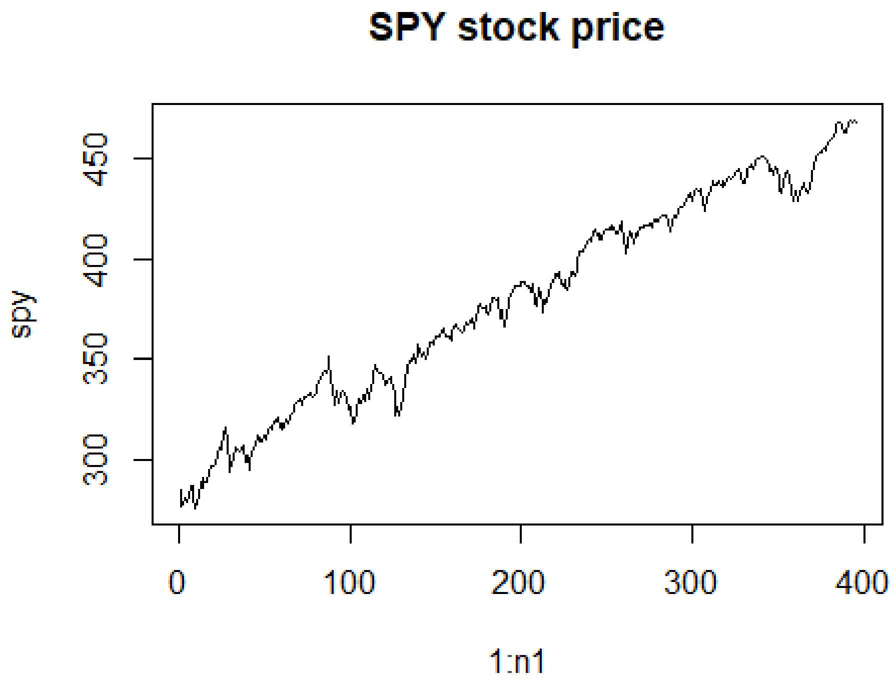
Stock price data is a typical time-series data. In this example we will look at price history of *SPY*, which tracks the performance of *Standard and Poor's 500*, whose components include the largest 500 U.S. equities.

```r
library(tseries)
library(normwhn.test)
library(forecast)
library(astsa)
library(Hmisc)

price <- read.csv('price_series.csv')

n1 = nrow(price)
start = 110
price <- price[start:n1, ]

## EDA
spy <- price$SPY
n1 = length(spy)
# plot
plot(x = 1:n1, y = spy, type = 'l', main = 'SPY stock price')
```



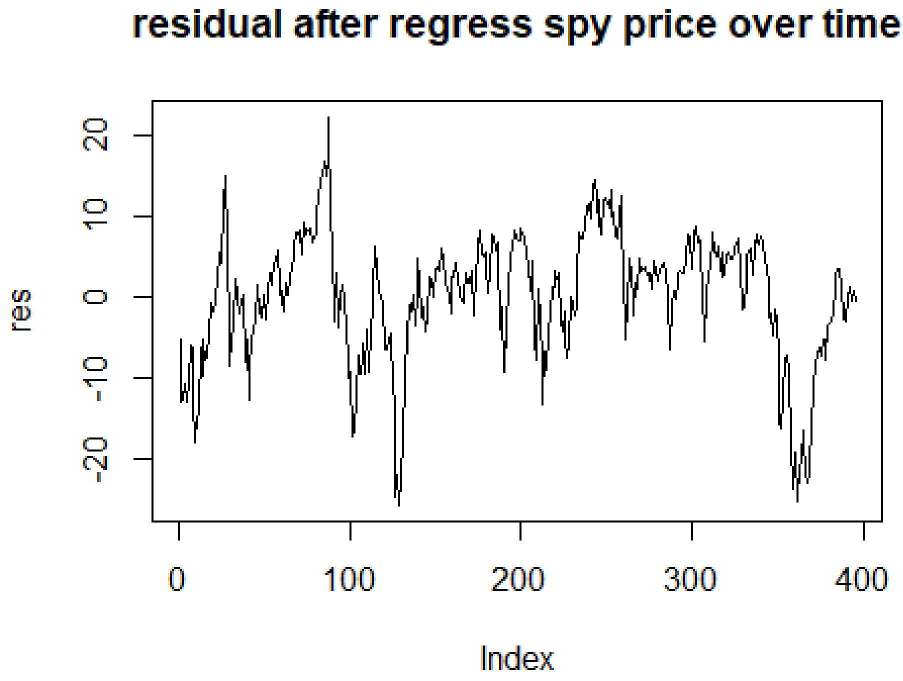
As we can see from the plot above, SPY stock price shows a upward linear trend and it is apparently non-stationary. But we don't know if it should be modeled by random walk with drift or a deterministic trend model. So the first step in this analysis is to apply the likelihood ratio test:

```
adf.test(spy)

## 
##  Augmented Dickey-Fuller Test
## 
## data:  spy
## Dickey-Fuller = -3.757, Lag order = 7, p-value = 0.02136
## alternative hypothesis: stationary
```

adf.test() function is the Augmented Dickey-Fuller test. Please see Appendix for more details. The null hypothesis of presence of unit root is rejected, which tells us that there might be a linear trend or intercept. Therefore, our first step is to de-trend data to make it stationary:

```
# Regress spy over time
x1 = 1:n1
fit1 <- lm(spy ~ x1)
res <- fit1$residuals
plot(res, type = 'l', main = 'residual after regress spy price over time') #
This looks stationary
```



Now the residual left over looks much better, similar to the stationary time-series plot in the Motivation section above. Note that by this regression analysis, we haven't considered the time-indexed nature of the stock data. So our next step is to perform analysis over possibly temporal-correlated residual terms. We use the term "possibly" since the residual terms can be independent just like normal regression setting. Statistical testing, including unit root test, is required for further analysis.

```
whitenoise.test(res) # p-value almost 0. White noise rejected
adf.test(res) # p-value = 0.02136. Unit root is rejected, but I would say
this is a stationary process
```

The first test is a *white noise test* testing for independence of data. With a small p-value, we reject the white noise process and concludes that residuals are temporal correlated.

The second test is the unit root test over residuals. This time, the null hypothesis of prescence of unit root is rejected again. However, the interpretation here would be different. Since the plot of residuals looks stationary, without a unit root we can conclude that it does come from a stationary process, and it fits under ARMA model setting

```
# Look at acf and pacf of residual
acf(res)
pacf(res) # Strong indication of AR process. I will try AR(4), AR(1)
ar1 = sarima(res, 1, 0, 0) # rho = 0.9, AIC = 5.28, BIC = 5.31
# ACF looks ok
ar4 = sarima(res, 4, 0, 0) # AIC = 5.27, BIC = 5.33
```

Finally, by looking at Autocorrelation and Partial Autocorrelation functions of the data, we believe AR(1) process is a better fit for the data with $\rho = 0.9$. Therefore, The final model for *SPY* price would be

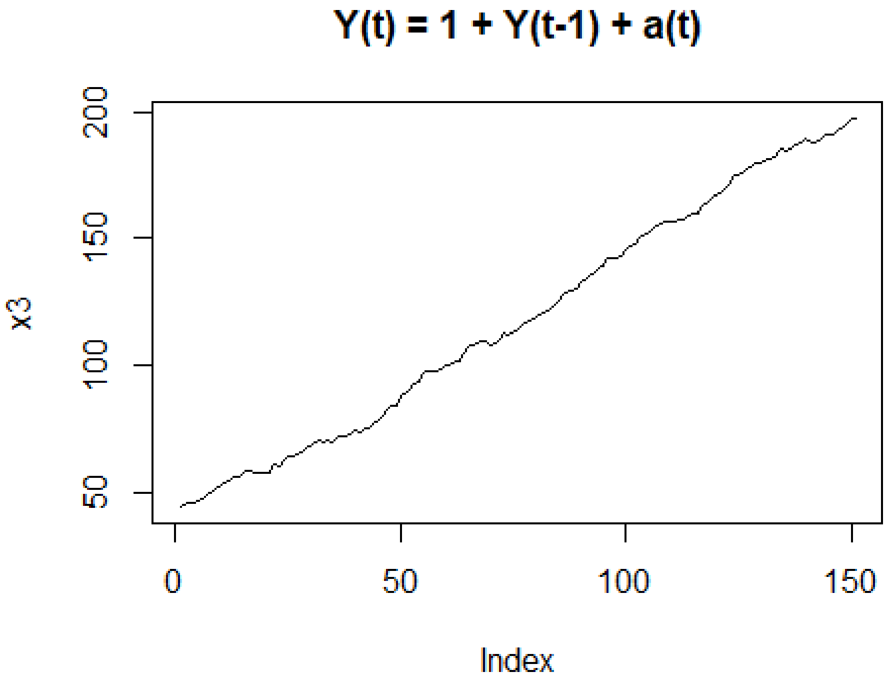
$$Y(t) = 288.79 + 0.458 * time + 0.9 * Y(t-1) + \epsilon(t)$$

Simulation: unit root with drift

Now we turn out eyes to the second case: unit root with drift. Due to lack of real-world data, we use simulated values with n = 150.

```
# Generate data from AR(1) process with drift
set.seed(1234)
obs = 200
burnin = 50
time = 1:obs
x3 = 1 * time + arima.sim(model = list(order = c(1, 0, 0), ar = 0.99), n =
obs)
x3 = x3[burnin: obs]

plot(x3, type = 'l', main = 'Y(t) = 1 + Y(t-1) + a(t)')
```



The pattern of this plot looks the same as *SPY* stock price. However, if we recklessly de-trended the analysis like we did before, the result of analysis would be confusing since the true model doesn't have a trend.

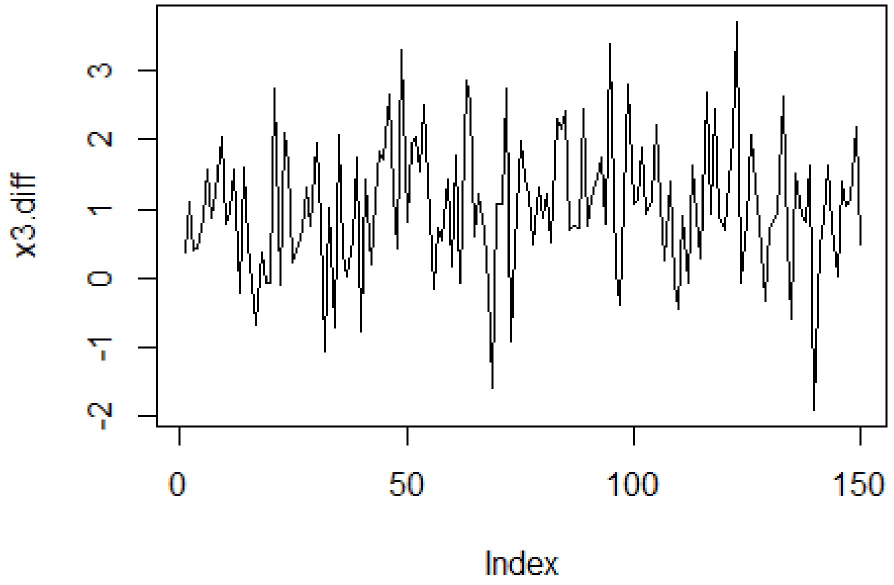
Unit root test would be a good starting point for this analysis:

```
adf.test(x3)

##
##  Augmented Dickey-Fuller Test
##
## data:  x3
## Dickey-Fuller = -2.5167, Lag order = 5, p-value = 0.3611
## alternative hypothesis: stationary
```

This time we get a large p-value, which suggests prescence of unit root. Then the next step would be to difference the data, i.e., $Y(t)^* = Y(t) - Y(t-1)$, and perform normal time-series analysis with differenced stationary data:

```
x3.diff <- diff(x3)
plot(x3.diff, type = 'l')
```



```
whitenoise.test(x3.diff)
```

After differencing, the only thing left in the model is random error, so the white noise test is not rejected, and we can conclude that our model is a random walk with drift.

Monte Carlo (MC) simulation

Next we'll use MC method to check the reliability of adf unit root testing with AR(1) model. We will generate data with different sample sizes and repeat the same procedure 1000 times. Then we will compare the true coverage rate with norminal coverage rate when $\alpha =$

0.05 with true $\rho = 0.5$. We would expect 95% of the time that adf test will be rejected with p-value less than α for large sample sizes.

```
# Generate data from AR(1) process with drift N times
fn.stats=function(data){

   pvalue=apply(data, 2, function(x) {adf.test(x)$p.value})
   proportion=sum(pvalue<0.05)/length(pvalue)
   avg_rej=mean(pvalue)
   se_rej=sd(pvalue)
   list (probability=proportion, average_rejction=avg_rej,
standarderror=se_rej)

}


fn.sample=function(N,n,seed=1234){
   set.seed(seed)
# AR(1) without drift
datand <- matrix(rep(1,N*n), nrow=n)
for(i in 1:N){
   datand[,i]=arima.sim(model = list(order = c(1, 0, 0), ar = 0.5), n = n)
}

out_wod <- fn.stats(datand)
# AR(1) with drift
time = 1:n
datawd <-matrix(rep(1,N*n), nrow=n)

for(i in 1:N){
   datawd[,i]=1 * time+arima.sim(model = list(order = c(1, 0, 0), ar = 0.5),
n = n)
}

out_wd <- fn.stats(datawd)

list(p_wod = out_wod, p_wd=out_wd )
}

# Simulation size
N<-1000
out.50<- fn.sample(N=N, n=50, seed=346)
out.100<- fn.sample(N=N, n=100, seed=346)
out.150<- fn.sample(N=N, n=150, seed=346)

################################
#Pleae un comment the following three statements when checking the results
from the table
################################
```

```
#out.50
#out.100
#out.150
```

THe table below summarize the rejection rates after we repeat the same procedure for 1000 times while varying the sample sizes. We can see that the the true rejection rate is very close to norminal level 95 percent when sample size increases to 150 for true $\rho = 0.5$. This simmple simulation is only designed to show that MC simulation can be a extremely valuable tool to check the expected results. Readers can also vary the value of true ρ or any other parameters of the interest that is pertinant to their study.

```
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse
1.3.1 --

## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x dplyr::src()       masks Hmisc::src()
## x dplyr::summarize() masks Hmisc::summarize()

library(gridExtra)

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine

library(knitr)
datap=c(out.50$p_wod$probability,out.100$p_wod$probability,out.150$p_wod$prob
ability)
datamean=c(out.50$p_wod$average_rejction,out.100$p_wod$average_rejction,out.1
50$p_wod$average_rejction)
datase=c(out.50$p_wod$standarderror,out.100$p_wod$standarderror,out.150$p_wod
$standarderror)
df_table=rbind(datap,datamean,datase)

colnames(df_table) <- c("Sample size 50", "Sample size 100", "Sample size
150")
rownames(df_table) <- c("Rejection Rate", "Average p-value",
                        "Standard Error of p-value")
```

```r
kable(round(df_table, 3),caption = "Table 1: Rejection rates for AR(1)
without Drift")
```

Table 1: Rejection rates for AR(1) without Drift

	Sample size 50	Sample size 100	Sample size 150
Rejection Rate	0.296	0.764	0.951
Average p-value	0.189	0.047	0.017
Standard Error of p-value	0.180	0.073	0.027

```r
datapd=c(out.50$p_wd$probability,out.100$p_wd$probability,out.150$p_wd$probab
ility)
datameand=c(out.50$p_wd$average_rejction,out.100$p_wd$average_rejction,out.15
0$p_wd$average_rejction)
datased=c(out.50$p_wd$standarderror,out.100$p_wd$standarderror,out.150$p_wd$s
tandarderror)
df_tabled=rbind(datapd,datameand,datased)

colnames(df_tabled) <- c("Sample size 50", "Sample size 100", "Sample size
150")
rownames(df_tabled) <- c("Rejection Rate", "Average p-value",
                        "Standard Error of p-value")
kable(round(df_tabled, 3),caption = "Table 2: Rejection rates for AR(1) with
Drift")
```

Table 2: Rejection rates for AR(1) with Drift

	Sample size 50	Sample size 100	Sample size 150
Rejection Rate	0.321	0.757	0.960
Average p-value	0.185	0.046	0.016
Standard Error of p-value	0.177	0.071	0.022

Conclusion

As demonstrated by the real-world example and simulated data, in time-series analysis, when we are presented with data with some trends in it, unit root test can help us to decide whether we should first difference the time series, or de-trend it. Considering that most time series data has an inherent trend, this likelihood ratio unit root test is indispensable in time-series analysis.

Appendix

Augmented Dickey Fuller Test (ADF Test)

Augmented Dickey Fuller Test (ADF Test) is the augmented version of Dickey-Fuller unit root test which takes into account of more complex AR models with unknown orders. This test will check the presence of a unit root with H_0 that a unit root exists.

The adf.test incorporates three types of linear regression models. The first type (type1) is a linear model with no drift and linear trend with respect to time:

$$dx_t = \rho * x_{t-1} + \beta_1 * dx_{t-1} + \ldots + \beta_{nlag-1} * dx_{t-nlag+1} + e_t,$$

where d is an operator of first order difference, i.e., $dx_t = x_t - xt - 1$, and e_t is an error term.

The second type (type2) is a linear model with drift but no linear trend:

$$dx_t = \mu + \rho * x_{t-1} + \beta_1 * dx_{t-1} + \ldots + \beta_{nlag-1} * dx_{t-nlag+1} + e_t.$$

The third type (type3) is a linear model with both drift and linear trend:

$$dx_t = \mu + \beta * t + \rho * x_{t-1} + \beta_1 * dx_{t-1} + \ldots + \beta_{nlag-1} * dx_{t-nlag+1} + e_t.$$

Source: https://www.rdocumentation.org/packages/aTSA/versions/3.1.2/topics/adf.test

When we call adf.test() in R, it runs all three tests. The p-value reported by this function is the smallest p-value from these three test cases. If one of these tests shows significance, we can conclude that there is unit root in the model and further diagnosis can be performed based on the visualization and regression analysis.

Codes

```
library(tseries)
library(normwhn.test)
library(forecast)
library(astsa)
library(Hmisc)

price <- read.csv('price_series.csv')

n1 = nrow(price)
start = 110
price <- price[start:n1, ]

## EDA
spy <- price$SPY
n1 = length(spy)
# plot
```

```r
plot(x = 1:n1, y = spy, type = 'l', main = 'SPY stock price')

adf.test(spy)

# Regress spy over time
x1 = 1:n1
fit1 <- lm(spy ~ x1)
res <- fit1$residuals
plot(res, type = 'l', main = 'residual after regress spy price over time') # This looks stationary

whitenoise.test(res) # p-value almost 0. White noise rejected
adf.test(res) # p-value = 0.02136. Unit root is rejected, but I would say this is a stationary process

# Look at acf and pacf of residual
acf(res)
pacf(res) # Strong indication of AR process. I will try AR(4), AR(1)
ar1 = sarima(res, 1, 0, 0) # rho = 0.9, AIC = 5.28, BIC = 5.31
# ACF looks ok
ar4 = sarima(res, 4, 0, 0) # AIC = 5.27, BIC = 5.33

# Generate data from AR(1) process with drift
set.seed(1234)
obs = 200
burnin = 50
time = 1:obs
x3 = 1 * time + arima.sim(model = list(order = c(1, 0, 0), ar = 0.99), n = obs)
x3 = x3[burnin: obs]

plot(x3, type = 'l', main = 'Y(t) = 1 + Y(t-1) + a(t)')

adf.test(x3)

x3.diff <- diff(x3)
plot(x3.diff, type = 'l')

whitenoise.test(x3.diff)
```