

MATH 545: INTRODUCTION TO TIME SERIES ANALYSIS

WINTER 2025

NON-STATIONARY MODELLING AND SIMULATION

ARIMA Models: An $\text{ARIMA}(p, d, q)$ model is a model for process $\{X_t\}$ specified as

$$\Phi(B)(1 - B)^d X_t = \Theta(B)Z_t$$

where $\Phi(\cdot)$ and $\Theta(\cdot)$ are the p - and q -order AR and MA polynomials, that is, the process $\{Y_t\}$ defined by differencing $\{X_t\}$ d times at lag 1,

$$Y_t = (1 - B)^d X_t$$

is a (stationary, causal) $\text{ARMA}(p, q)$ process. Analysis of data via an $\text{ARIMA}(p, d, q)$ model proceeds by first differencing d times, and then proceeding as for an $\text{ARMA}(p, q)$ process. Such processes are easily handled using R. For example, for the $\text{ARIMA}(2, 1, 1)$ model given by the polynomials

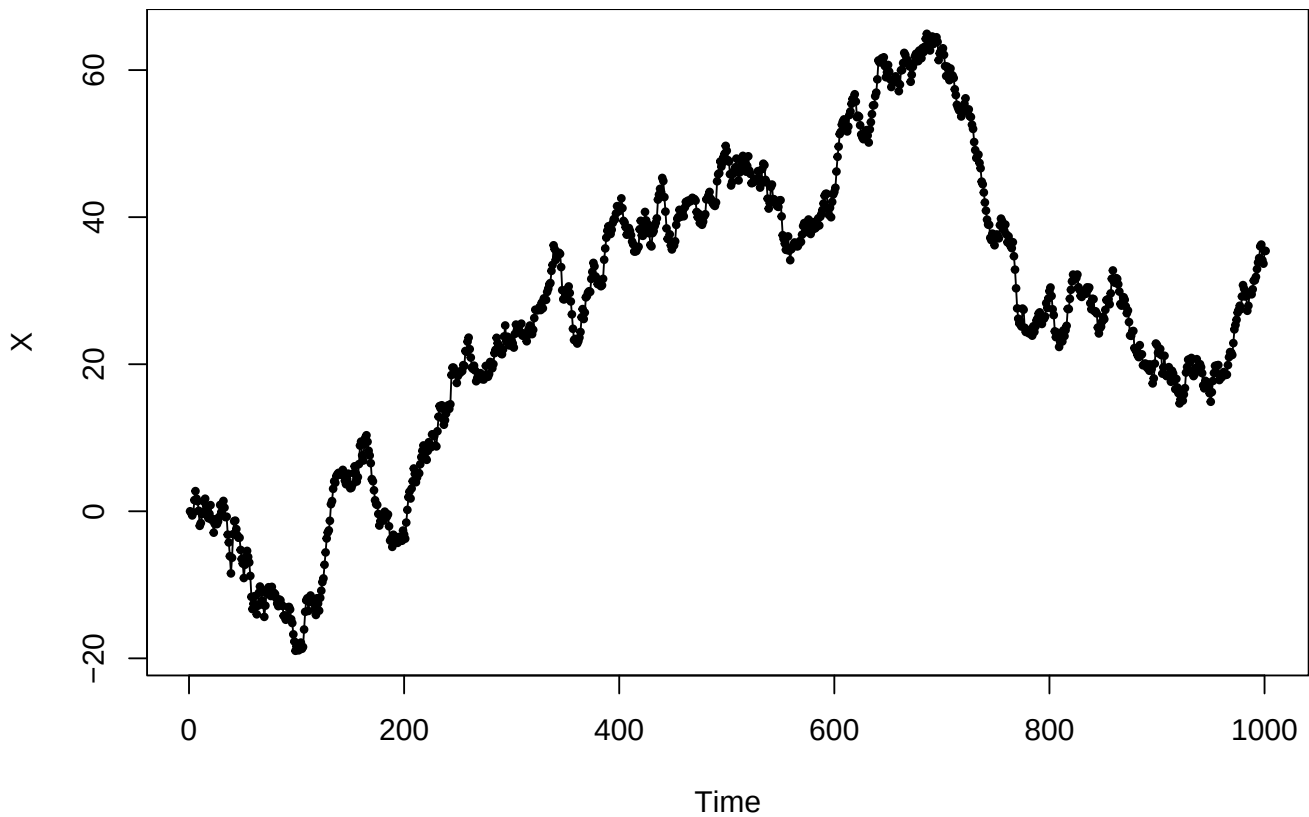
$$\Phi(z) = (1 - 0.2z)(1 + 0.8z) = (1 + 0.6z - 0.16z^2) \quad \Theta(z) = 1 + 0.9z$$

that is

$$(1 + 0.6B - 0.16B^2)(1 - B)X_t = (1 + 0.9B)Z_t$$

is simulated and analyzed as follows:

```
set.seed(3834)
n<-1000
X<-arima.sim(n,model=list(order=c(2,1,1),ar=c(-0.6,0.16),ma=0.9))
par(mar=c(4,4,0,0));plot(X,type='l');points(X,pch=19,cex=0.5)
```



```

fit1<-arima(X,order=c(2,1,1))
fit1

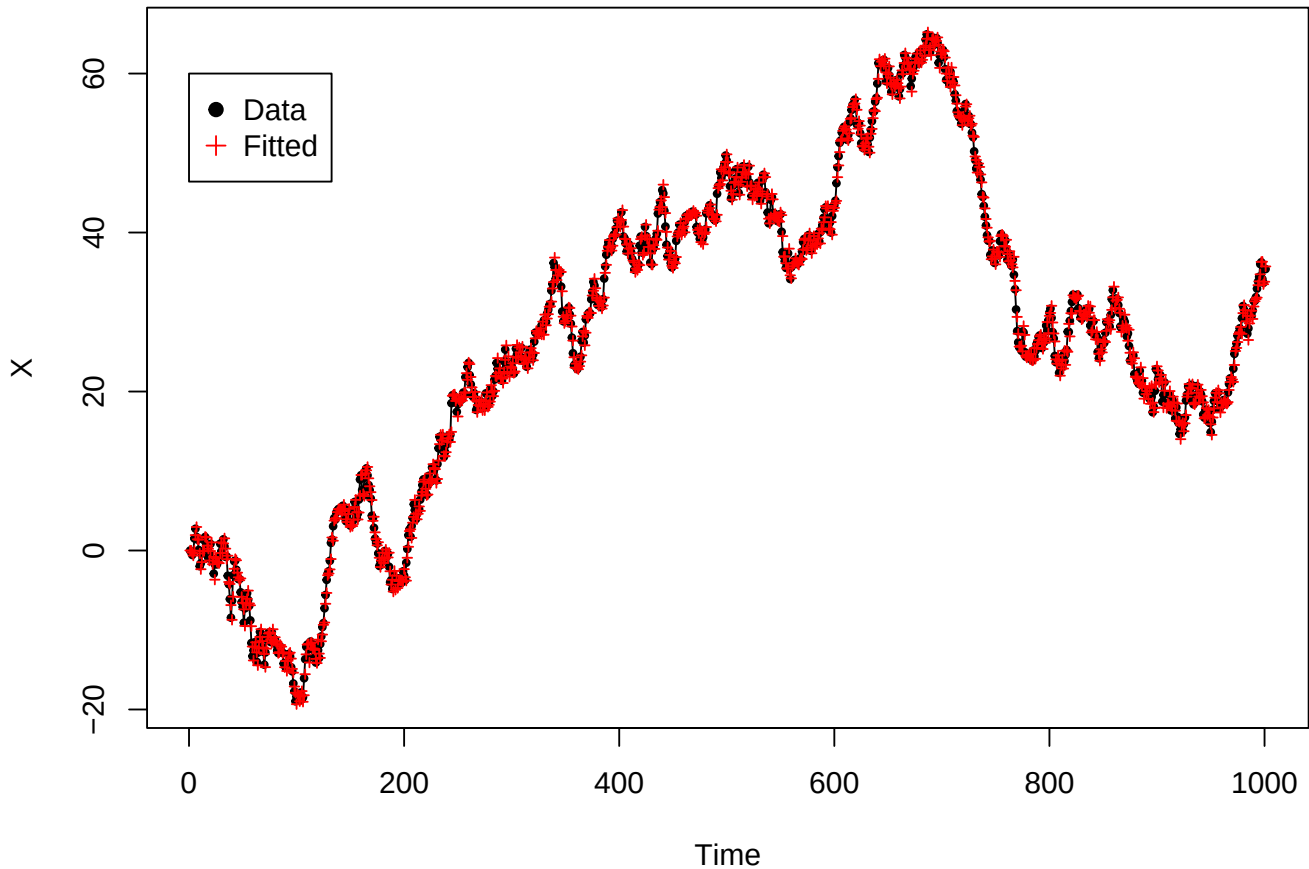
+
+ Call:
+ arima(x = X, order = c(2, 1, 1))
+
+ Coefficients:
+      ar1      ar2      ma1
+    -0.6488  0.1609  0.9456
+ s.e.   0.0383  0.0343  0.0206
+
+ sigma^2 estimated as 1.029:  log likelihood = -1433.21,  aic = 2874.42

```

```

fit1.resids<-residuals(fit1)
fit1.fitted<-X-fit1.resids
par(mar=c(4,4,1,0));plot(X,type='l');points(X,pch=19,cex=0.5)
points(fit1.fitted,pch=3,cex=0.5,col='red')
legend(0,60,c('Data','Fitted'),pch=c(19,3),col=c('black','red'))

```



```

round(acf(fit1.resids,lag.max=20,main='ACF for residuals',plot=F)$acf,4)[,1]

+ [1] 1.0000 0.0024 -0.0171 -0.0050 0.0581 0.0149 -0.0001 0.0317 -0.0111
+ [10] -0.0015 -0.0152 -0.0030 0.0424 0.0204 -0.0127 0.0361 0.0268 -0.0439
+ [19] -0.0330 -0.0184 0.0184

```

The residuals are uncorrelated, and plausibly white noise. The data can also be analyzed after differencing:

```

fit2<-arima(diff(X),order=c(2,0,1))
fit2

+
+ Call:
+ arima(x = diff(X), order = c(2, 0, 1))
+
+ Coefficients:
+      ar1      ar2      ma1  intercept
+    -0.6495  0.1600  0.9455     0.0354
+ s.e.    0.0383  0.0343  0.0206     0.0419
+
+ sigma^2 estimated as 1.028:  log likelihood = -1432.85,  aic = 2875.71

```

The parameter estimates are similar, and differ only because of the loss of a single data point that happens due to the differencing.

Seasonal ARIMA Models: A seasonal ARIMA (or SARIMA) model is an ARIMA model with an additional stochastic seasonal structure that can allow for seasonal differencing. The $ARIMA(p, d, q) \times (P, D, Q)_S$ model is a model for process $\{X_t\}$ specified as

$$\Phi_S(B^S)\Phi(B)(1 - B^S)^D(1 - B)^dX_t = \Theta_S(B^S)\Theta(B)Z_t$$

where the seasonal polynomials

$$\Phi_S(z) = 1 - \phi_{S1}z - \dots - \phi_{SP}z^P \quad \Theta_S(z) = 1 - \theta_{S1}z - \dots - \theta_{SQ}z^Q$$

allow for a seasonal ARMA structure, that is

$$\Phi_S(B^S) = 1 - \phi_{S1}B^S - \phi_{S2}B^{2S} - \dots - \phi_{SP}B^{SP}$$

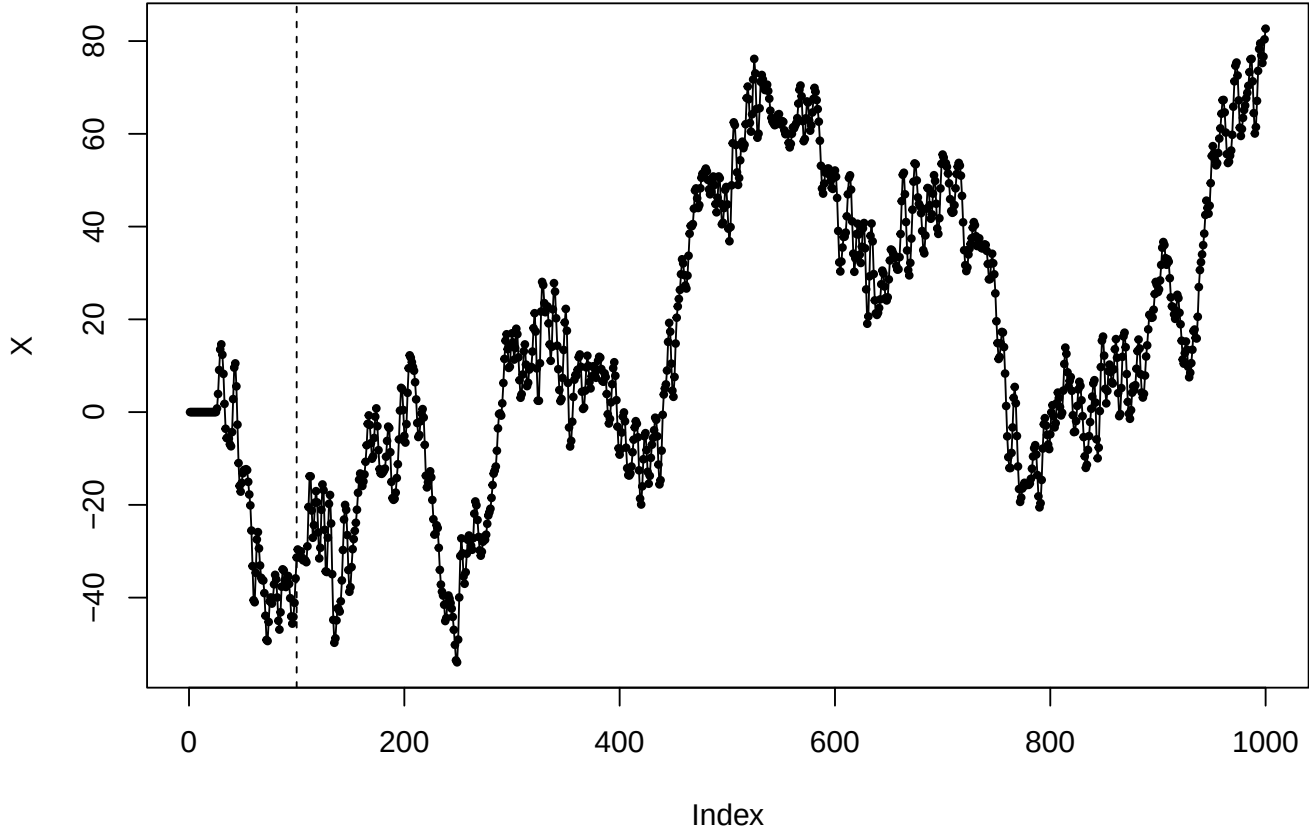
and so on, as well as seasonal differencing, at lag S . Such models can be handled using the `sarima` package for simulation, and the `arima` function for fitting. For example, suppose we wanted to simulate and analyze the $ARIMA(2, 1, 1) \times (1, 0, 2)_{12}$ model with

$$(1 - 0.3B^{12})(1 - 1.2B + 0.8B^2)(1 - B)X_t = (1 + 0.7B^{12} - 0.3B^{24})(1 + 0.4B)Z_t$$

```

library(sarima)
set.seed(384)
p<-2;d<-1;q<-1
P<-1;D<-0;Q<-2
S<-12
n<-1000
nvals<-n
X<-sim_sarima(n=nvals,
  model=list(ar=c(1.2, -0.8), ma=0.4, #Non-seasonal
            sar=0.3, sma=c(0.7,-0.3 ), #Seasonal
            iorder=d, siorder=D, nseasons=S))
par(mar=c(4,4,2,0));plot(X,type='l');points(X,pch=19,cex=0.5)
abline(v=100,lty=2)

```



The arguments to `sim_sarima` named `iorder` and `siorder` specify the parameters d and D respectively.

```
fit0<-arima(X,order=c(2,1,1),seasonal=list(order=c(1,0,2),period=12),
           optim.control=list(maxit=1000))
fit0
+
+ Call:
+ arima(x = X, order = c(2, 1, 1), seasonal = list(order = c(1, 0, 2), period = 12),
+       optim.control = list(maxit = 1000))
+
+ Coefficients:
+      ar1      ar2      ma1      sar1      sma1      sma2
+      1.2041  -0.7927  0.4158  -0.3369  1.2848  0.2935
+ s.e.  0.0220   0.0213  0.0310   0.6171  0.6292  0.6153
+
+ sigma^2 estimated as 1.027:  log likelihood = -1452.85,  aic = 2919.7
```

Note that we can combine the $\Phi_S(\cdot)$ and $\Phi(\cdot)$ filters, and the $\Theta_S(\cdot)$ and $\Theta(\cdot)$ filters, that is

$$\Phi_S(z^S)\Phi(z) = 1 - \phi_1^*z - \dots - \phi_{p+PS}^*z^{p+PS} = \Phi^*(z)$$

$$\Theta_S(z^S)\Theta(z) = 1 - \theta_1^*z - \dots - \theta_{q+QS}^*z^{q+QS} = \Theta^*(z)$$

to obtain an $\text{ARMA}(p + PS, q + QS)$ model before seasonal and lag one differencing. That is

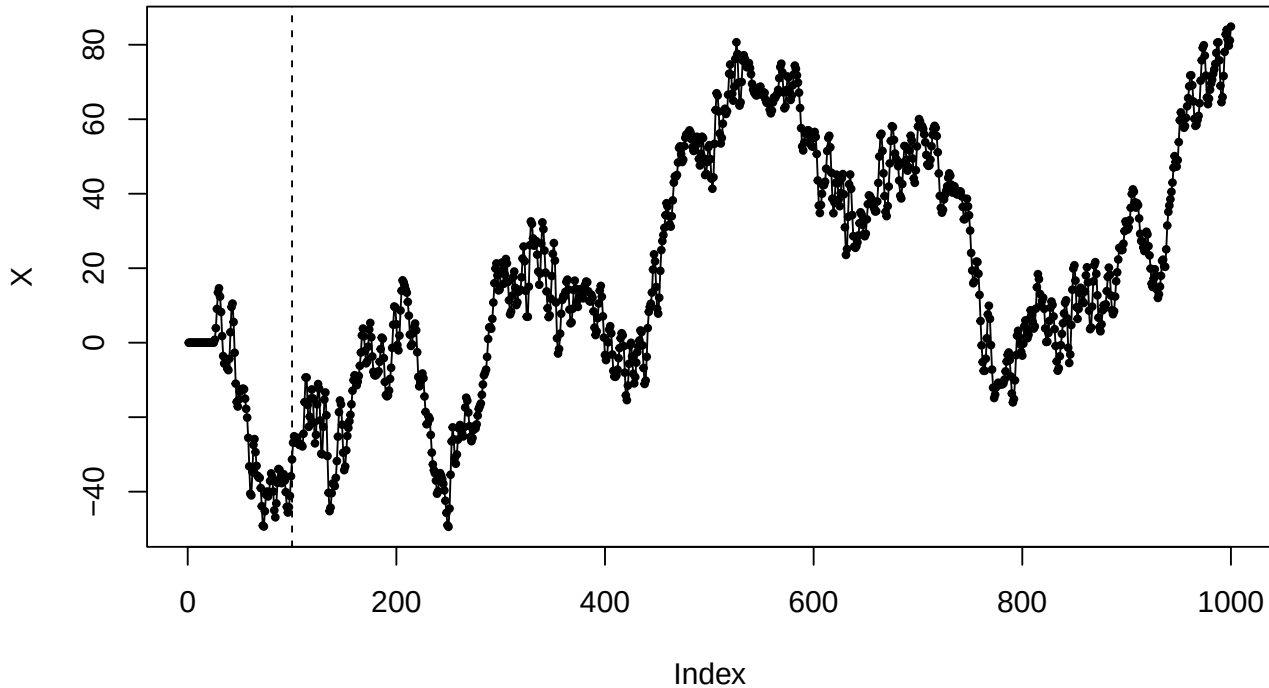
$$Y_t = (1 - B^S)^D(1 - B)^d X_t$$

is such that $\{Y_t\} \sim \text{ARMA}(p + PS, q + QS)$. Accordingly, we can use the `diffinv` function to generate $\{X_t\}$ from $\{Y_t\}$ (here recall $D = 0$).

```

phi.vec<-c(1.2,-0.8,rep(0,9),0.3,-0.3*1.2,0.3*0.8)
theta.vec<-c(0.4,rep(0,10),0.7,0.7*0.4,rep(0,10),-0.3,-0.3*0.4)
set.seed(384)
Y<-arima.sim(n,model=list(order=c(p+S*P,0,q+S*Q),ar=phi.vec,ma=theta.vec))
Y<-c(diff(X[1:100]),Y);X<-diffinv(Y,lag=1);X<-X[1:n]
par(mar=c(4,4,1,1));plot(X,type='l');points(X,pch=19,cex=0.5);abline(v=100,lty=2)

```



Notice that simulating the data in this way cannot recover the original series without the correct ‘initialization’.

Unit Roots: In the simplest case, consider an AR(1) model

$$X_t = \phi X_{t-1} + Z_t$$

from which it follows that

$$X_t - X_{t-1} = \phi X_{t-1} - X_{t-1} + Z_t = (\phi - 1)X_{t-1} + Z_t$$

that is

$$(1 - B)X_t = \phi^* X_{t-1} + Z_t$$

where $\phi^* = \phi - 1$. This suggests an intercept-free regression relationship between the differenced data $(1 - B)X_t$ and the lagged data X_{t-1} . We can test the hypotheses

$$H_0 : \phi^* = 0 \quad (\text{i.e. } \phi = 1)$$

$$H_1 : \phi^* < 0 \quad (\text{i.e. } \phi < 1)$$

by using the test statistic

$$t_{DF} = \frac{\hat{\phi}^*}{\text{e.s.e}(\hat{\phi}^*)}$$

which can be computed using the `lm` function in R. The precise form of the test statistic is

$$\hat{\phi}^* = \frac{\sum_{t=2}^n (x_t - x_{t-1})x_{t-1}}{\sum_{t=2}^n x_{t-1}^2}$$

with estimated standard error

$$\text{e.s.e}(\hat{\phi}^*) = \sqrt{\frac{s^2}{\sum_{t=2}^n x_{t-1}^2}}$$

where

$$s^2 = \frac{1}{n-2} \sum_{t=2}^n (x_t - x_{t-1} - \hat{\phi}^* x_{t-1})^2.$$

This test statistic is a form of the **Dickey-Fuller** test statistic. It can be computed using standard tools.

```
set.seed(234)
n<-1000
Z<-rnorm(n)           #Simulate the whote noise
X<-cumsum(Z)[-n]       #Create the random walk
dX<-Z[-1]             #Create the lagged series
fit0<-lm(dX~-1+X)      #Fit the regression mode
coef(summary(fit0))

+      Estimate Std. Error   t value Pr(>|t|)
+ X -0.002145914  0.00287926 -0.7453003 0.4562657

(phi.hat<-sum(dX*X)/sum(X^2)) #Check the calculation
+ [1] -0.002145914

s<-sqrt(sum((dX-phi.hat*X)^2)/(n-2))
(ese.phi.hat<-s/sqrt(sum(X^2)))

+ [1] 0.00287926

(t.df<-phi.hat/ese.phi.hat)

+ [1] -0.7453003

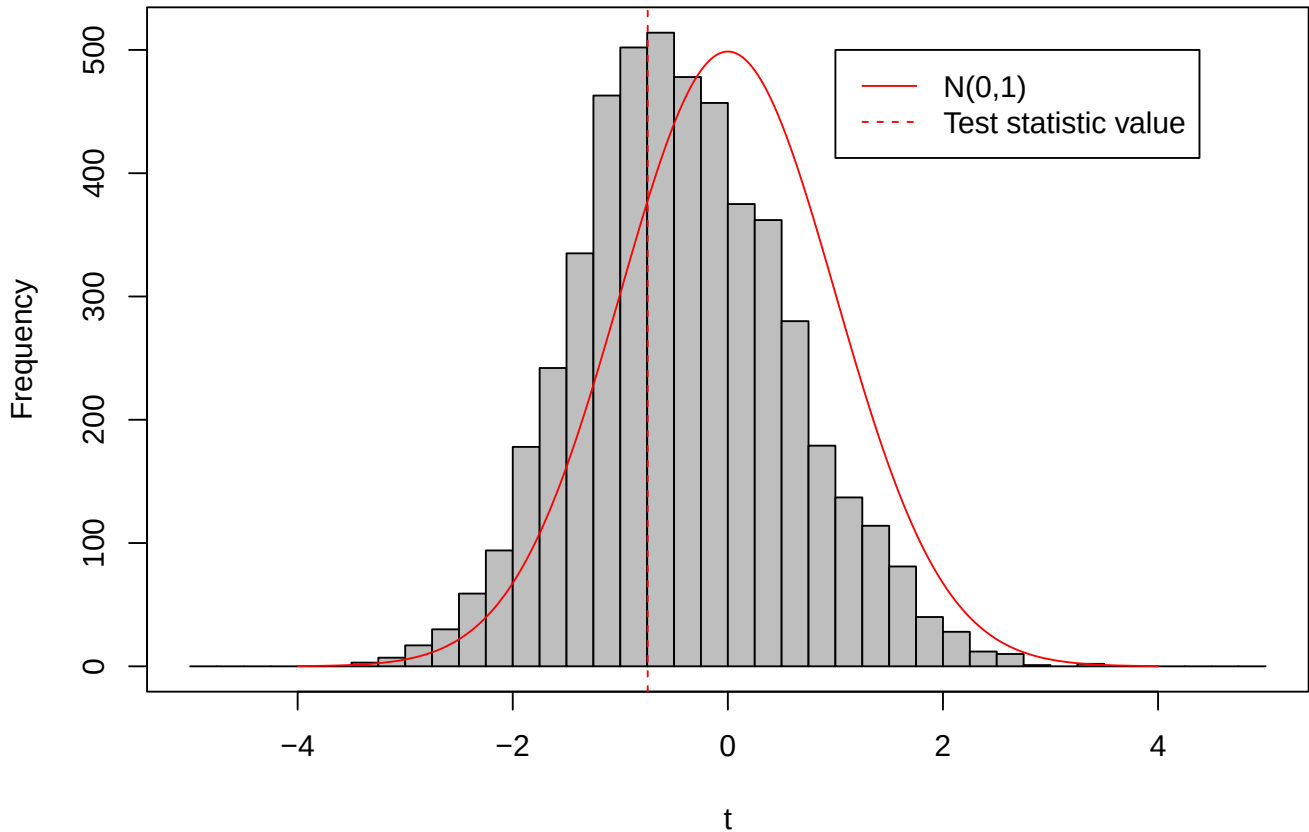
(p.val<-pnorm(t.df))

+ [1] 0.2280451
```

The test here has non-stationarity as the null hypothesis, and the p -value is relatively large (i.e. 0.2280451). However, the difficulty here is that *the test statistic is not asymptotically distributed as a standard Normal random variable* under the null hypothesis; in fact, it is not centered at zero. This is demonstrated in the following simulation study: we simulate $N = 5000$ times from a random walk and compute the test statistic each time.

```
set.seed(234)
n<-1000;N<-5000
ur.test0<-function(nv){ #Function to compute the D-F test statistic.
  Z<-rnorm(nv);dX<-Z[-1];X<-cumsum(Z)[-n]
  fit0<-lm(dX~-1+X)
  tst<-coef(summary(fit0))[1,3]
  return(tst)
}
t.stat<-replicate(N, ur.test0(n)) #Use the replicate function for speed of computing.
par(mar=c(4,4,2,0))
hist(t.stat,main='Dickey-Fuller test statistic null distribution',
     col='gray',xlab='t',br=seq(-5,5,by=0.25));box() #Null distribution computed via Monte Carlo
abline(v=t.df,col='red',lty=2)
xv<-seq(-4,4,length=1000);yv<-dnorm(xv);lines(xv,yv*0.25*N,col='red')
legend(1,500,c('N(0,1)', 'Test statistic value'),col='red',lty=c(1,2))
```

Dickey-Fuller test statistic null distribution



Using this (Monte Carlo) simulation study, we estimate the p -value in this one-sided test by the proportion of test statistic values generated under the null hypothesis that are more negative than the observed version.

```
(p.mc<-sum(t.df > t.stat)/N)
+ [1] 0.388
```

Thus the estimated p -value is 0.388. This is accurate to approximately $\sqrt{\hat{p}(1-\hat{p})/N} = 0.0068914$ (this is the Monte Carlo standard error), so we can conclude result is much less significant than that given initially. The empirical critical values at 0.01, 0.05 and 0.10 are

```
quantile(t.stat,probs=c(0.01,0.05,0.1))
+      1%      5%      10%
+ -2.519584 -1.918737 -1.634016
```

We can also use the `ur.df` function in the `urca` library:

```
library(urca)
summary(ur.df(cumsum(Z),type='none',lags=0))
+
+ #####
+ # Augmented Dickey-Fuller Test Unit Root Test #
+ #####
+
+ Test regression none
+
+
+ Call:
```

```

+ lm(formula = z.diff ~ z.lag.1 - 1)
+
+ Residuals:
+      Min       1Q   Median       3Q      Max
+ -3.03526 -0.68164  0.01244  0.61487  3.07803
+
+ Coefficients:
+           Estimate Std. Error t value Pr(>|t|)
+ z.lag.1 -0.002146   0.002879  -0.745   0.456
+
+ Residual standard error: 0.9852 on 998 degrees of freedom
+ Multiple R-squared:  0.0005563, Adjusted R-squared:  -0.0004452
+ F-statistic: 0.5555 on 1 and 998 DF,  p-value: 0.4563
+
+
+ Value of test-statistic is: -0.7453
+
+ Critical values for test statistics:
+      1pct   5pct  10pct
+ tau1 -2.58 -1.95 -1.62

```

Here the critical values are computed by more extensive computation and analytical approximations.

For illustration, we now consider the result for a simulated stationary process, an AR(1) model with $\phi = 0.95$.

```

n<-1000
X<-arima.sim(n,model=list(ar=0.95))
dX<-diff(X)                                #Create the lagged series
X<-X[-n]                                    #Create the random walk
fit0<-lm(dX~1+X)                            #Fit the regression mode
coef(summary(fit0))

+      Estimate Std. Error t value      Pr(>|t|)
+ X -0.04800701  0.009667528 -4.9658 0.0000008044032

(phi.hat<-sum(dX*X)/sum(X^2)) #Check the calculation
+ [1] -0.04800701

s<-sqrt(sum((dX-phi.hat*X)^2)/(n-2))
(ese.phi.hat<-s/sqrt(sum(X^2)))

+ [1] 0.009667528

(t.df<-phi.hat/ese.phi.hat)

+ [1] -4.9658

(p.val<-pnorm(t.df))

+ [1] 0.0000003420914

(p.mc<-sum(t.df > t.stat)/N)

+ [1] 0

```

The test statistic is much more negative, and the Monte Carlo estimated p -value is now 0. Using `ur.df` we obtain

```

summary(ur.df(X,type='none',lags=0))

+
+ #####
+ # Augmented Dickey-Fuller Test Unit Root Test #

```

```

+ #####
+
+ Test regression none
+
+ Call:
+ lm(formula = z.diff ~ z.lag.1 - 1)
+
+ Residuals:
+      Min       1Q   Median       3Q      Max
+ -3.7212 -0.6433  0.0074  0.6480  2.9812
+
+ Coefficients:
+             Estimate Std. Error t value    Pr(>|t|)
+ z.lag.1 -0.047942    0.009678  -4.954 0.000000856 ***
+ ---
+ Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
+
+ Residual standard error: 0.9947 on 997 degrees of freedom
+ Multiple R-squared:  0.02402, Adjusted R-squared:  0.02304
+ F-statistic: 24.54 on 1 and 997 DF, p-value: 0.0000008557
+
+ Value of test-statistic is: -4.9535
+
+ Critical values for test statistics:
+      1pct  5pct 10pct
+ tau1 -2.58 -1.95 -1.62

```

Here the test statistic is -4.9658004, which is much more extreme than the 0.01 critical value (which is -2.58) so we reject the null hypothesis, and conclude that the process is stationary.

The Dickey-Fuller procedure can be extended to allow for a ‘drift’ term, that is

$$X_t = \phi_0^* + \phi X_{t-1} + Z_t$$

say, so that

$$(1 - B)X_t = \phi_0^* + \phi^* X_{t-1} + Z_t$$

where we need to include an intercept in the regression model.

```

set.seed(234)
library(urca)
Z<-rnorm(n);X<-cumsum(Z)[-n]
dX<-Z[-1]
fit0<-lm(dX~X)
coef(summary(fit0))

+             Estimate Std. Error  t value    Pr(>|t|)
+ (Intercept) -0.081438495 0.044957816 -1.811442 0.07037321
+ X           -0.007573077 0.004153013 -1.823514 0.06852491

summary(ur.df(cumsum(Z),type='drift',lags=0))

+
+ #####
+ # Augmented Dickey-Fuller Test Unit Root Test #
+ #####
+
+ Test regression drift
+
+

```

```

+ Call:
+ lm(formula = z.diff ~ z.lag.1 + 1)
+
+ Residuals:
+      Min       1Q   Median       3Q      Max
+ -2.95171 -0.63992  0.03264  0.64069  3.11275
+
+ Coefficients:
+              Estimate Std. Error t value Pr(>|t|)
+ (Intercept) -0.081438   0.044958  -1.811   0.0704 .
+ z.lag.1      -0.007573   0.004153  -1.824   0.0685 .
+ ---
+ Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
+
+ Residual standard error: 0.984 on 997 degrees of freedom
+ Multiple R-squared:  0.003324, Adjusted R-squared:  0.002324
+ F-statistic: 3.325 on 1 and 997 DF,  p-value: 0.06852
+
+ Value of test-statistic is: -1.8235 1.919
+
+ Critical values for test statistics:
+      1pct   5pct 10pct
+ tau2 -3.43 -2.86 -2.57
+ phi1  6.43  4.59  3.78

```

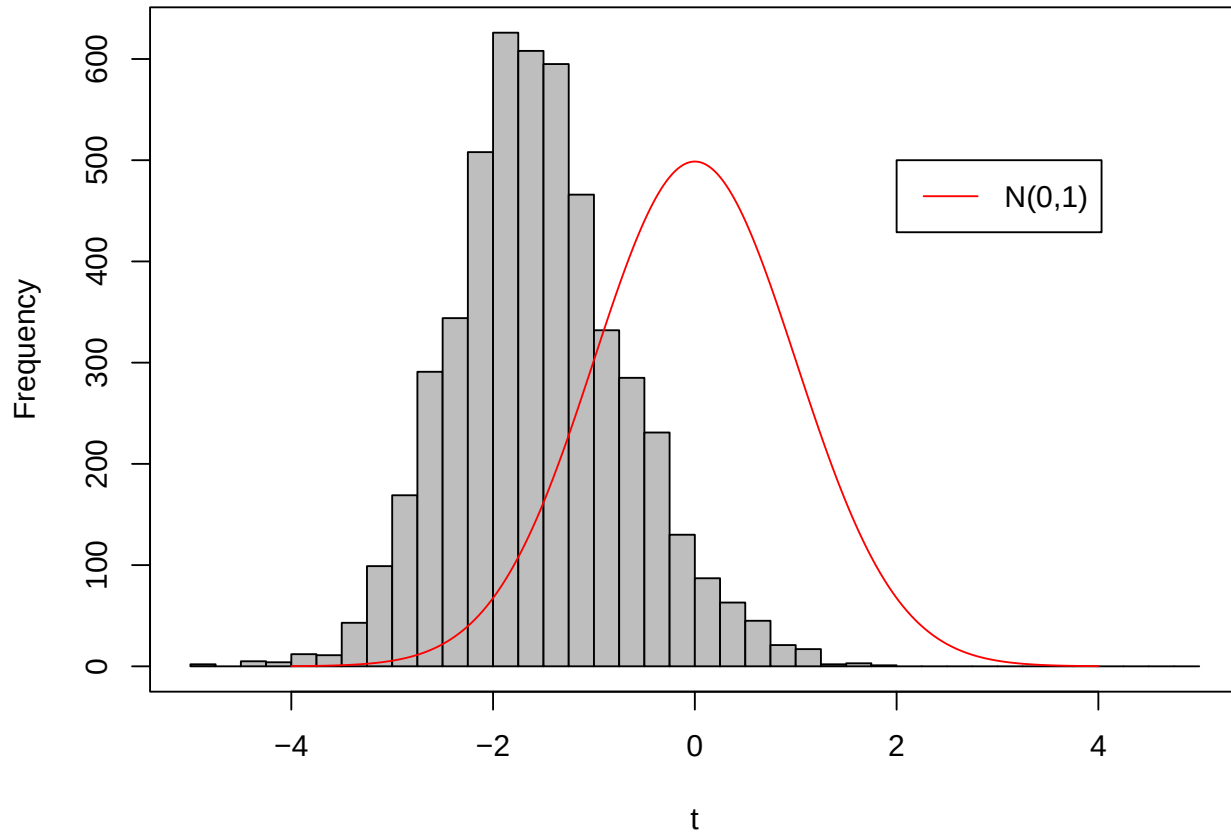
Here we inspect the tau2 statistic. The null distribution in this case is different again:

```

set.seed(234)
n<-1000
ur.test1<-function(nv){
  Z<-rnorm(nv);dX<-Z[-1];X<-cumsum(Z)[-n]
  fit0<-lm(dX~X)
  tst<-coef(summary(fit0))[2,3]
  return(tst)
}
t.stat<-replicate(N, ur.test1(n))
par(mar=c(4,4,2,2))
hist(t.stat,main='Augmented Dickey-Fuller Test Statistic',
     col='gray',xlab='t',br=seq(-5,5,by=0.25));box()
xv<-seq(-4,4,length=1000);yv<-dnorm(xv);lines(xv,yv*0.25*N,col='red')
legend(2,500,c('N(0,1)'),col='red',lty=1)

```

Augmented Dickey–Fuller Test Statistic



```
t.df<-coef(summary(fit0))[2,3]
(p.mc<-sum(t.df > t.stat)/N)

+ [1] 0.387

quantile(t.stat,probs=c(0.01,0.05,0.1))

+      1%      5%      10%
+ -3.423264 -2.878236 -2.617482
```