**Resolving non-stationarity problems**

Differencing

Differencing helps to create the constant mean needed for stationarity. We will use differencing a lot!

1st differences: xt – xt-1 = ∇xt

2nd differences: (xt – xt-1) – (xt-1 – xt-2) = ∇xt – ∇xt-1 = ∇2xt

Taking “differences” between successive data values in the time series helps to remove trend. Specifically, 1st differences help remove linear trend and 2nd differences help remove quadratic trend.

Why does this work? Consider the linear trend model xt = β0 + β1t where t = time and β1 ≠ 0. Then

xt – xt-1 = β0 + β1t – [β0 + β1(t – 1)] = β1

which is not dependent on t.

Example: Nonstationarity in the mean (nonstat.mean.R, nonstat.mean.csv)

| **t** | **xt** | **xt** – **xt-1** |
| --- | --- | --- |
| 1 | 1.31 | NA |
| 2 | 13.67 | 13.67 - 1.31 = 12.36 |
| 3 | 6.29 | 6.29 - 13.67 = -7.38 |
| 4 | -0.95 | -7.24 |
| 5 | 9.59 | 10.53 |
| 6 | -0.45 | -10.03 |
|  |  |  |
| 98 | 102.80 | 2.80 |
| 99 | 93.82 | -8.98 |
| 100 | 108.72 | 14.91 |

Below is the code for a plot of xt vs. t and the ACF for xt.

> nonstat.mean <- read.csv(file = "nonstat.mean.csv")

> head(nonstat.mean)

time x

1 1 1.31

2 2 13.67

3 3 6.29

4 4 -0.95

5 5 9.59

6 6 -0.45

> tail(nonstat.mean)

time x

95 95 92.69

96 96 91.22

97 97 100.00

98 98 102.80

99 99 93.82

100 100 108.72

> dev.new(width = 8, height = 6, pointsize = 10)

> plot(x = nonstat.mean$x, ylab = expression(x[t]), xlab =

"t (time)", type = "l", col = "red", main =

"Nonstationary time series", panel.first=grid(col =

"gray", lty = "dotted"))

> points(x = nonstat.mean$x, pch = 20, col = "blue")



> acf(x = nonstat.mean$x, type = "correlation", main =

"Plot of the ACF")



> x.ts <- ts(nonstat.mean[,2])

> set1 <- ts.intersect(x.ts, x.ts1 = lag(x = x.ts, k = -1))

> head(set1)

x.ts x.ts1

[1,] 13.67 1.31

[2,] 6.29 13.67

[3,] -0.95 6.29

[4,] 9.59 -0.95

[5,] -0.45 9.59

[6,] 17.22 -0.45

> cor(set1)

x.ts x.ts1

x.ts 1.000000 0.857779

x.ts1 0.857779 1.000000

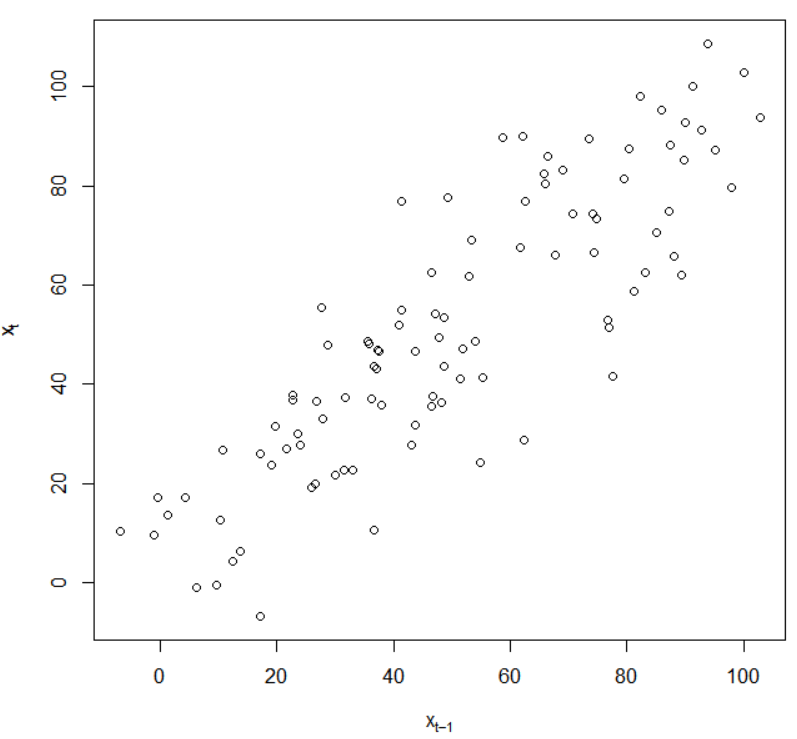
> # Need as.numeric() so that plot.ts() is not run, want

plot.default()

> plot(y = as.numeric(set1[,1]), x = as.numeric(set1[,2]),

ylab = expression(x[t]), type = "p", xlab =

expression(x[t-1]))



Notes:

* This data is said to have “nonstationarity in the mean” because the mean of xt, μt, appears to be changing as a function of time.
* Why is there large positive autocorrelation at lag = 1, 2, … ?

Below is the code to find the first differences:

> #Find first differences

> first.diff <- diff(x = nonstat.mean$x, lag = 1,

differences = 1)

> first.diff[1:5]

[1] 12.36 -7.38 -7.24 10.54 -10.04

> nonstat.mean$x[2] - nonstat.mean$x[1]

[1] 12.36

> nonstat.mean$x[3] - nonstat.mean$x[2]

[1] -7.38

> plot(x = first.diff, ylab = expression(x[t]-x[t-1]), xlab

= "t (time)", type = "l", col = "red", main = "First

differences", panel.first = grid(col = "gray", lty =

"dotted"))

> points(x = first.diff, pch = 20, col = "blue")



> acf(x = first.diff, type = "correlation", main = "Plot of

the ACF for first differences")



If you want xt and xt − xt-1 in the same data frame, use the ts.intersect() function:

> x <- ts(data = nonstat.mean$x)

> x.diff1 <- ts(data = first.diff, start = 2)

> ts.intersect(x, x.diff1)

Time Series:

Start = 2

End = 100

Frequency = 1

x x.diff1

2 13.67 12.36

3 6.29 -7.38

4 -0.95 -7.24

5 9.59 10.54

EDITED

96 91.22 -1.47

97 100.00 8.78

98 102.80 2.80

99 93.82 -8.98

100 108.72 14.90

Why does the data set start at 2?

Other types of differencing:

* 2nd differences: diff(x, lag = 1, differences = 2)
* xt – xt-2: diff(x, lag = 2, differences = 1); this can be useful when there is a “seasonal” trend

Note: There are formal hypothesis tests to determine if differencing is needed. This corresponds to an area of time series known as “unit root” testing. The name will be clear once we examine autoregressive models in detail.

Backshift operator

A convenient way to represent differencing in time series models is to use the “backshift operator”. It is denoted by “B” and defined as follows:

Bxt = xt-1

Notice that xt moved back one-time period when the backshift operator was applied to it.

In general, B2xt = xt-2, B3xt = xt-3, …, and Bkxt = xt-k.

Notes:

* + Let C be a constant not indexed by time. Then BC = C.
  + (1-B)xt = xt – xt-1 = ∇xt
  + B×B = B2
  + (1-B)2xt = (1 - 2B + B2)xt   
    = xt – 2Bxt + B2xt   
    = xt – 2xt-1 + xt-2   
    = xt – xt-1 – xt-1 + xt-2

= (xt – xt-1) – (xt-1 – xt-2)  
= ∇2xt

* + (1-B)0xt = xt
  + (1-B)xt can be thought of as a “linear filter” since the linear trend is being filtered out of the time series.

Example: Moving Average

mt = (wt + wt-1 + wt-2)/3 where wt ~ independent N(0,) for t = 1, …, n can be represented by (1 + B + B2)wt/3

Example: Autoregression

xt = 0.7xt-1 + wt where wt ~ independent N(0,) for t = 1, …, n can be represented by (1 – 0.7B)xt = wt

Example: 1st differencing needed example (first\_diff.R)

Consider the following model:

(1-0.7B)(1-B)xt = wt where wt ~ ind. N(0,) for t = 1, …, n. This simplifies to

(1-0.7B)(xt – xt-1) = wt

⇔ xt – xt-1 – 0.7Bxt + 0.7Bxt-1 = wt

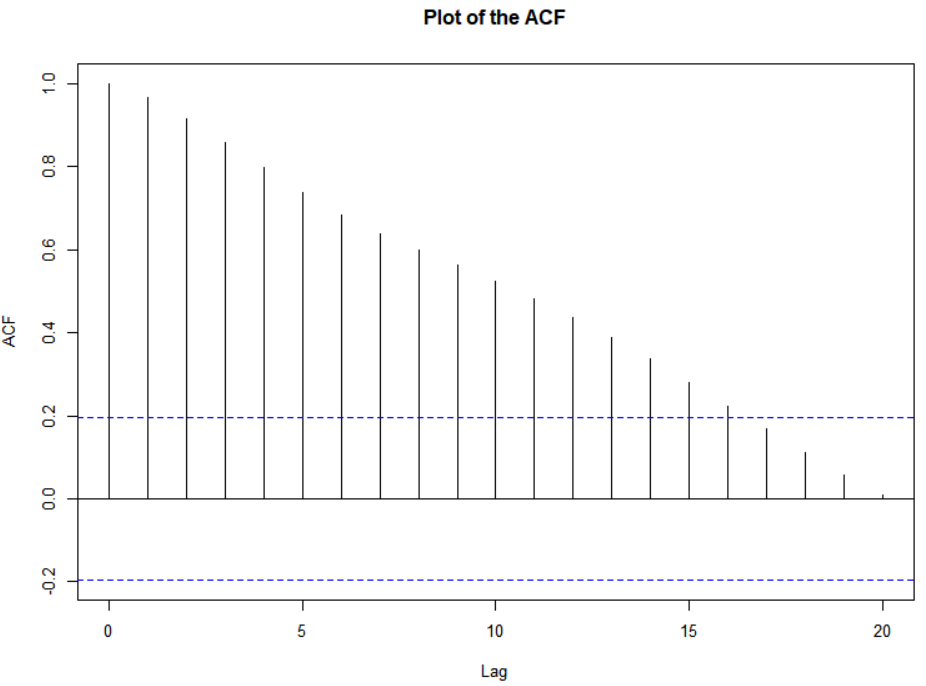
⇔ xt = xt-1 + 0.7xt-1 - 0.7xt-2 + wt

⇔ xt = 1.7xt-1 - 0.7xt-2 + wt

Later in the course, we will identify this as a ARIMA(1,1,0) model.

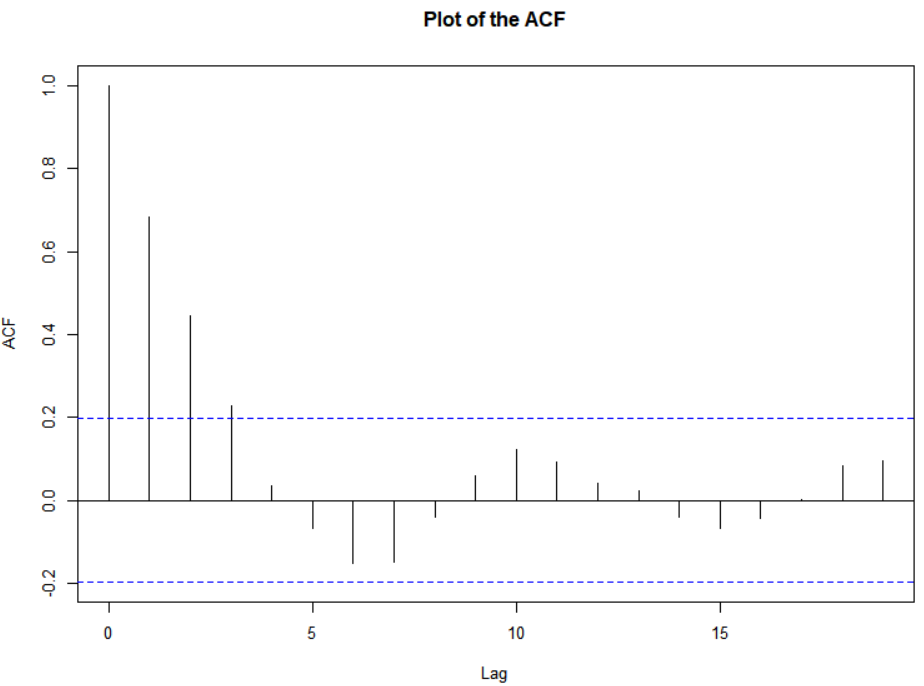
Suppose a realization of a time series is simulated from this model. Below is a plot of the data.





After the first differences:





R code used for this example.

set.seed(7328)

w <- rnorm(n = 200, mean = 0, sd = 1)

x <- numeric(length = 200)

x.1 <- 0

x.2 <- 0

for (i in 1:length(x)) {

x[i] <- 1.7\*x.1 - 0.7\*x.2 + w[i]

x.2 <- x.1

x.1 <- x[i]

}

#Do not use first 100

X <- x[101:200]

Dev.new(width = 8, height = 6, pointsize = 10)

plot(x = x, ylab = expression(x[t]), xlab = "t", type =

"l", col = "red", lwd = 1 , main =

expression(paste("Data simulated from ", (1-0.7\*B)\*(1-

B)\*x[t] == w[t], " where ", w[t], "~N(0,1)")),

panel.first=grid(col = "gray", lty = "dotted"))

points(x = x, pch = 20, col = "blue")

acf(x = x, type = "correlation", main = "Plot of the ACF")

#Find first differences

plot(x = diff(x = x, lag = 1, differences = 1), ylab =

expression(x[t]-x[t-1]), xlab = "t (time)", type = "l",

col = "red", main = expression(paste("1st diff. for data

simulated from ", (1-0.7\*B)\*(1-B)\*x[t] == w[t], " where

", w[t], "~N(0,1)")), panel.first=grid(col = "gray", lty

= "dotted"))

points(x = diff(x = x, lag = 1, differences = 1), pch =

20, col = "blue")

acf(x = diff(x = x, lag = 1, differences = 1), type =

"correlation", main = "Plot of the ACF")

See the program for an easier way to use arima.sim() to simulate the data.

Fractional Differencing

Use fractional powers of B between –0.5 to 0.5 to do the differencing. This is used with long-memory time series.

Notes:

* Differencing is often used to help make a nonstationary in the mean time series stationary. Unfortunately in real applications, we do not know what exact level of differencing is needed (we can approximate it). If a too high of level of differencing is done, this can hurt a time series model. As a compromise between differencing and not differencing at all, fractional differencing can be used.
* The reason why these are called a “long memory” time series can be seen from a “short memory” time series. A short memory stationary process will have ρ(h)→0 “quickly” as h→∞. A long memory time series does not and has ρ(h)→0 “slowly”. More on this later in the course.
* The fractional difference series can be represented as



where the πj’s are found through a Taylor series expansion of (1-B)d.

#### Transformations

In regression analysis, transformations of the response variable are taken to induce approximate constant variance. In a similar manner, we can take transformations of xt to help make a nonstationary in the variance time series be approximately stationary in the variance.

Example: Johnson & Johnson earnings per share data (jj.R)

This data comes from Shumway and Stoffer’s book

> library(astsa)

> x <- jj

> x

Qtr1 Qtr2 Qtr3 Qtr4

1960 0.710000 0.630000 0.850000 0.440000

1961 0.610000 0.690000 0.920000 0.550000

1962 0.720000 0.770000 0.920000 0.600000

1963 0.830000 0.800000 1.000000 0.770000

1964 0.920000 1.000000 1.240000 1.000000

1965 1.160000 1.300000 1.450000 1.250000

1966 1.260000 1.380000 1.860000 1.560000

1967 1.530000 1.590000 1.830000 1.860000

1968 1.530000 2.070000 2.340000 2.250000

1969 2.160000 2.430000 2.700000 2.250000

1970 2.790000 3.420000 3.690000 3.600000

1971 3.600000 4.320000 4.320000 4.050000

1972 4.860000 5.040000 5.040000 4.410000

1973 5.580000 5.850000 6.570000 5.310000

1974 6.030000 6.390000 6.930000 5.850000

1975 6.930000 7.740000 7.830000 6.120000

1976 7.740000 8.910000 8.280000 6.840000

1977 9.540000 10.260000 9.540000 8.729999

1978 11.880000 12.060000 12.150000 8.910000

1979 14.040000 12.960000 14.850000 9.990000

1980 16.200000 14.670000 16.020000 11.610000

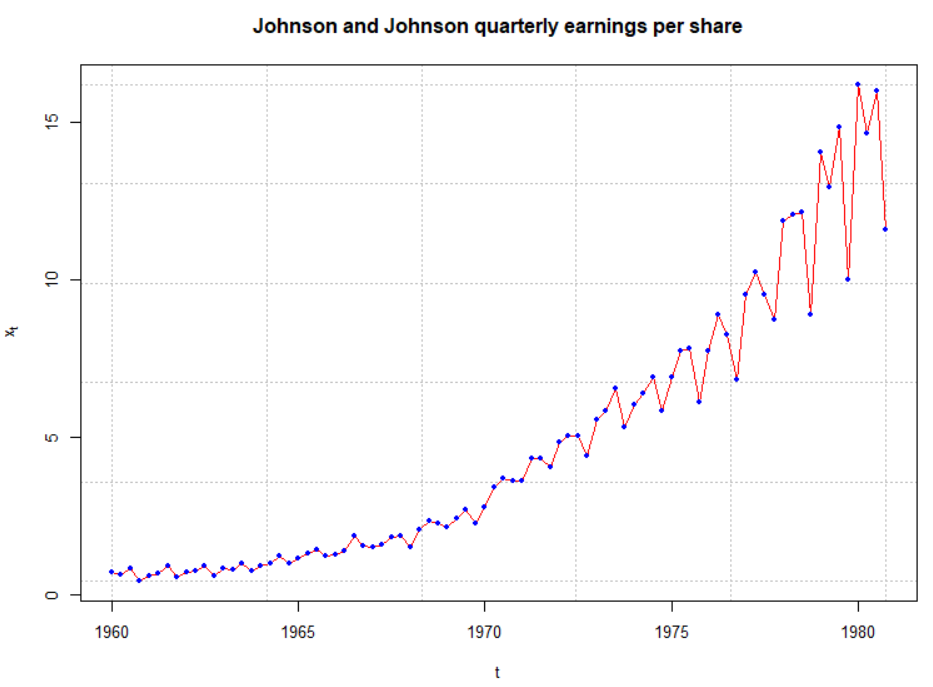
> plot(x = x, ylab = expression(x[t]), xlab = "t", type =

"l", col = "red", lwd = 1, main = "Johnson and Johnson

quarterly earnings per share", panel.first = grid(col =

"gray", lty = "dotted"))

> points(x = x, pch = 20, col = "blue")



> plot(x = log(x), ylab = expression(log(x[t])), xlab =

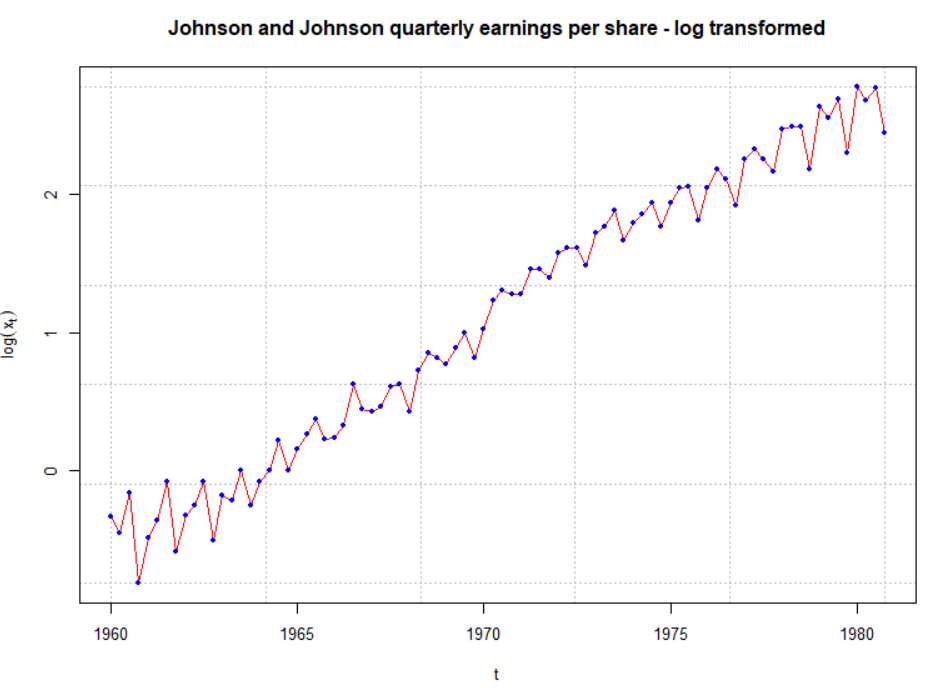
"t", type = "l", col = "red", lwd = 1, main = "Johnson

and Johnson quarterly earnings per share - log

transformed", panel.first = grid(col = "gray", lty =

"dotted"))

> points(x = log(x), pch = 20, col = "blue")



Notes:

* Typical transformations include log(xt), , and . There are a few different ways to deciding between the appropriate transformations. Usually, I will try all of these transformations and examine a plot of the transformed data over time to determine if the transformation worked. Constants may need to be added to xt if xt can be less than 0.
* Regression courses sometimes teach the Box-Cox family of transformations to determine an appropriate transformation. The process involves finding the “best” λ to transform xt in the following manner:



I have not found it used often in time series analysis. For example, Shumway and Stoffer suggest it could be used here, but do not explore it further. The BoxCox() and BoxCox.lambda() functions from the forecast package provide ways to obtain it. Please see my program.

* If the variance stabilizing transformation is needed, do this before differencing (see Wei’s time series book for a discussion). For example, suppose differencing and a natural log variance stabilizing transformation is needed. Then examine log(xt) – log(xt-1) instead of log(xt – xt-1).
* Variance stabilizing transformations often help with normality assumption of wt.