IV. Introduction to Time Series – AR(1) Model and Forecasting

a. Introduction to Dependent Observations
b. Checking for Independence
c. Autocorrelation
d. The AR(1) Model
e. Random Walks
f. Trend Models and US GDP
g. Google Trend Modeling

a. Introduction to Dependent Observations

Consider observations taken over time.

To denote this, we will index the observations with the letter t rather than the letter i.

Our data will be observations on Y_1, Y_2, ...Y_t, ...where t indexes the day, month, year, or any time interval.

Key new idea:

Exploit the dependence in the series

Time series analysis is about uncovering, modeling, and exploiting dependence

a. Introduction to Dependent Observations

We will NOT assume that Y_{t-1} is *independent* of Y_t

Example: Is tomorrow's temperature independent of today's?

Suppose $y_1 \ldots y_T$ are the temperatures measured daily for several years. Which of the following two predictors would work better:

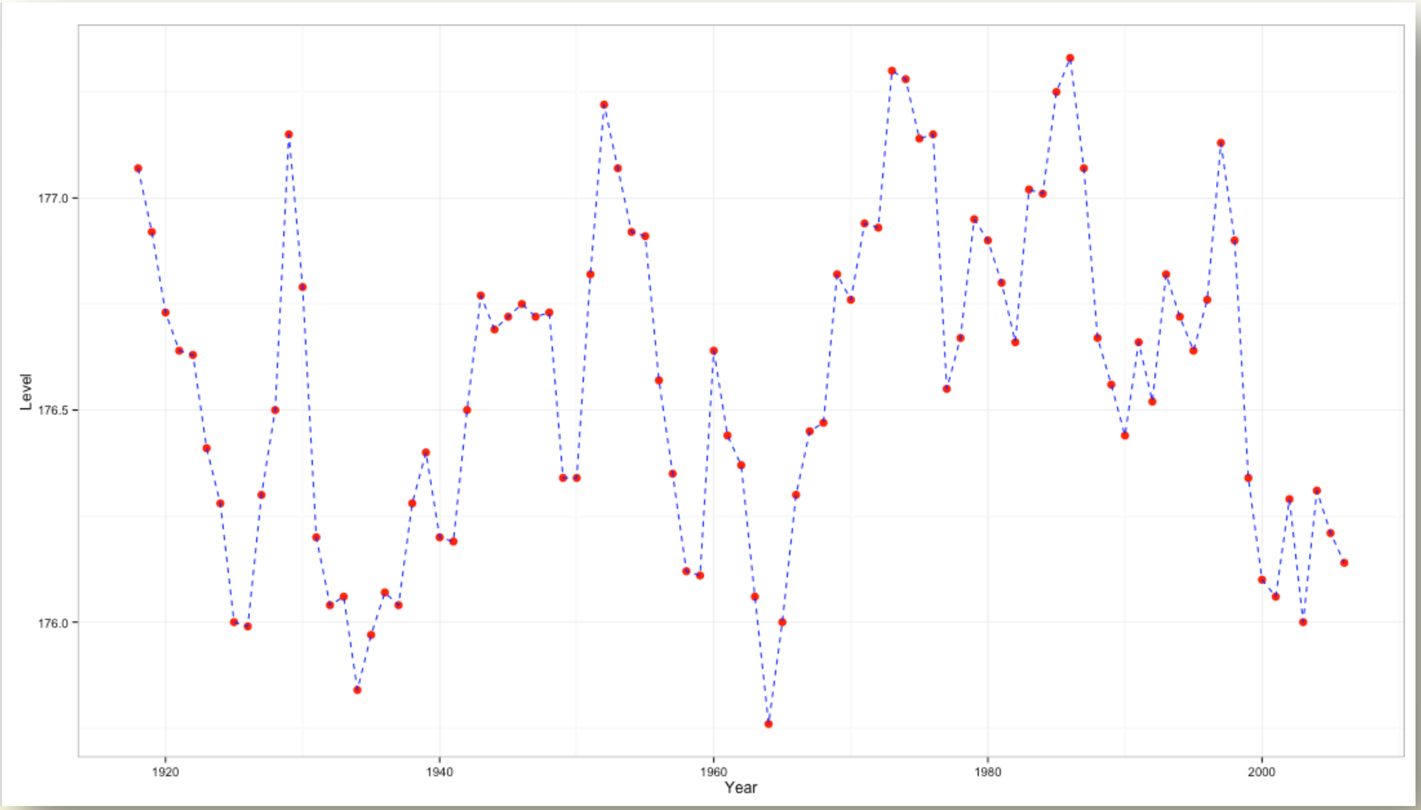
i. the average of the temperatures from the previous year
ii. the temperature on the previous day?

If the readings are iid $N(\mu, \sigma^2)$, what would be your prediction for Y_{T+1}?

This example demonstrates that we should handle dependent time series quite differently from independent series.

a. Introduction to Dependent Observations

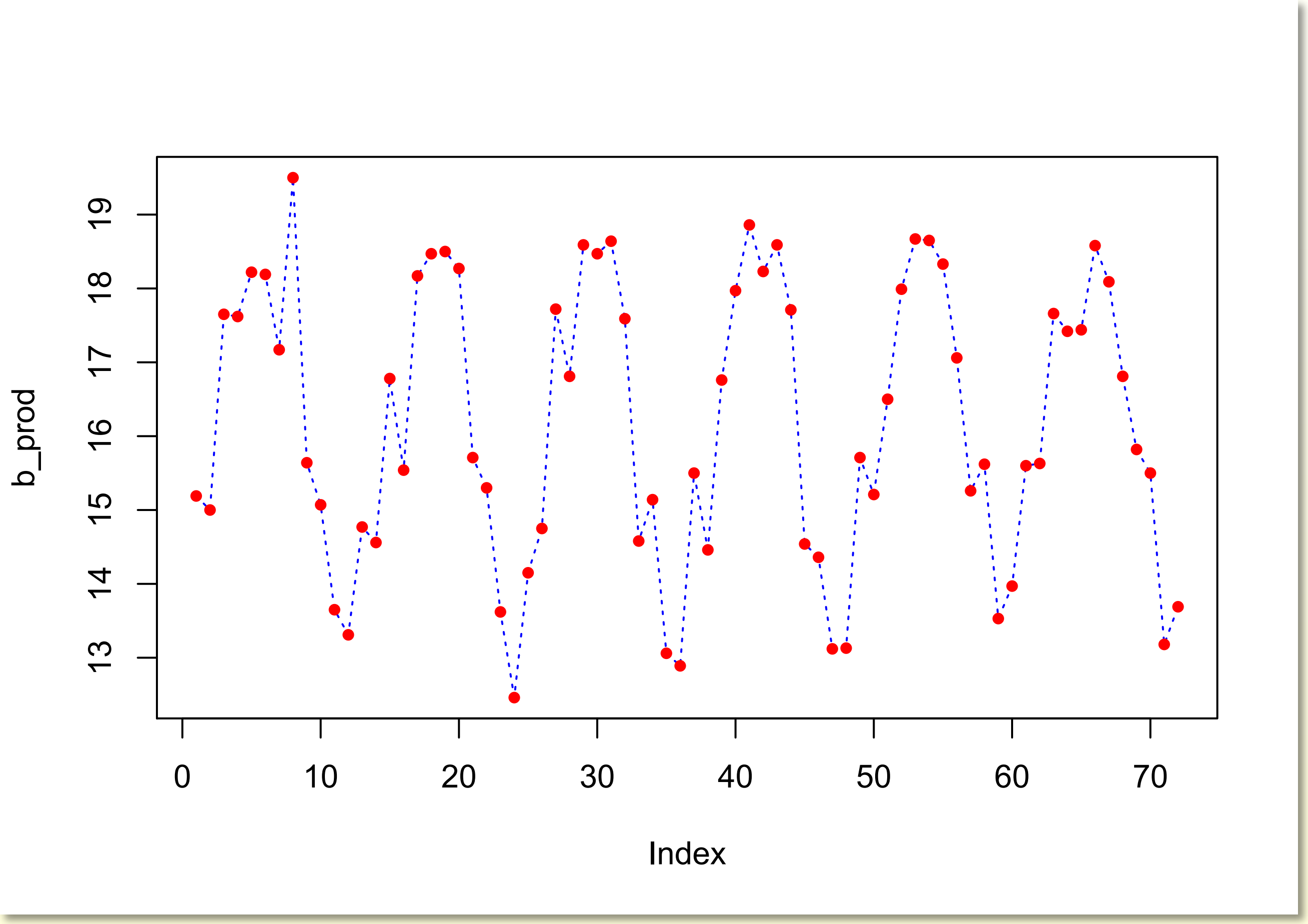
The Lake Michigan Time Series



Water level in Lake Michigan measured in June, `data(lmich_yr)`.

a. Introduction to Dependent Observations

Monthly US Beer Production (millions of barrels)

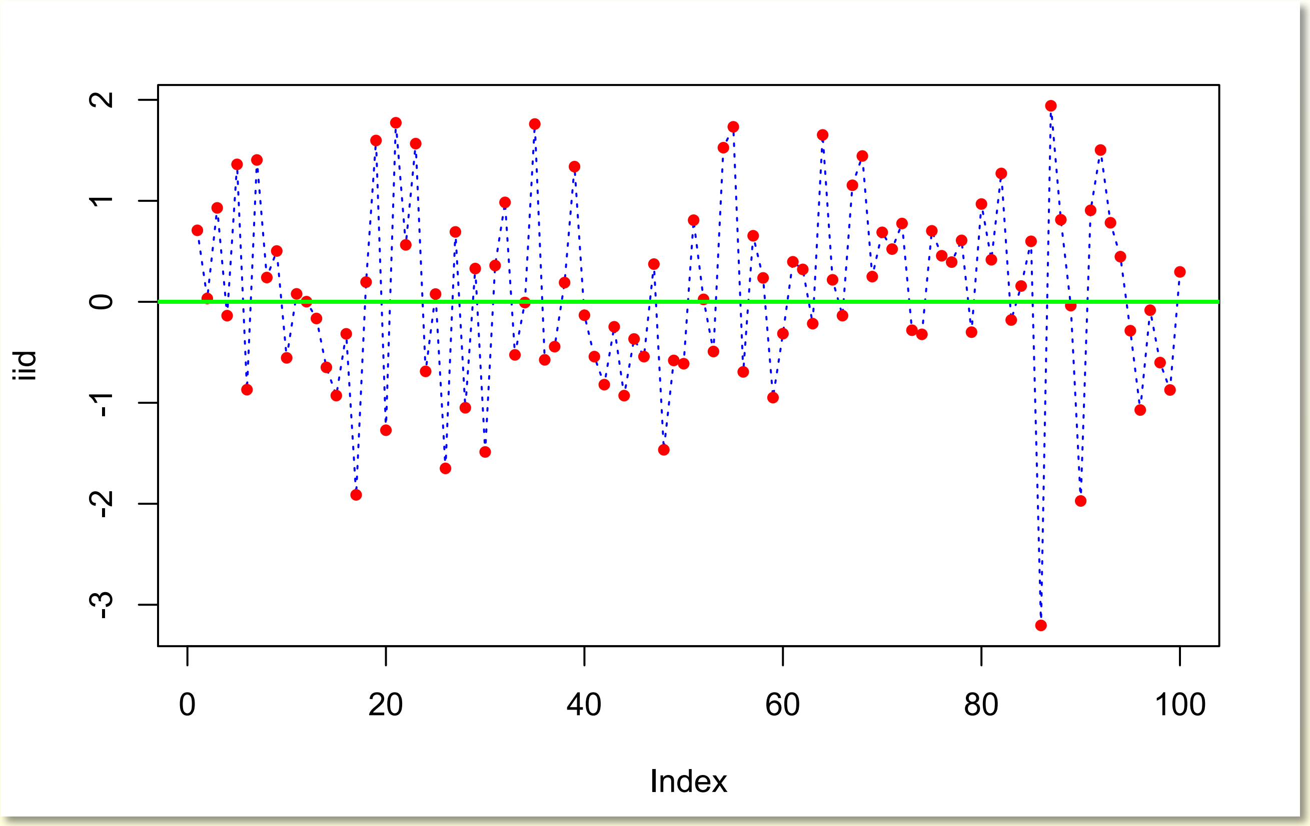


Strong Seasonality

`data(beerprod)`

a. Introduction to Dependent Observations

What Does IID Data Look Like?



many (but not too many) crossings of the mean

b. Checking for Independence

Independence:

$\boxed{\text{Knowing } Y_t \text{ does not help you predict } Y_{t+1}}$

It is not always easy just to look at the data and decide whether a time series is independent.

So how can we tell?

Plot Y_t vs. Y_{t-1} to check for a relationship

or

Plot Y_t vs. Y_{t-s} for s = 1, 2, …

b. Checking for Independence

How do we do this in R? – Use the “back” command

```
> back(b_prod)
```

	b_prod	b_prod(t-1)
1	15.19	*
2	15.00	15.19
3	17.65	15.00
4	17.62	17.65
5	18.22	17.62
6	18.19	18.22

Now each row has Y at time t, and Y one period ago

Y

Y lagged once

b. Checking for Independence

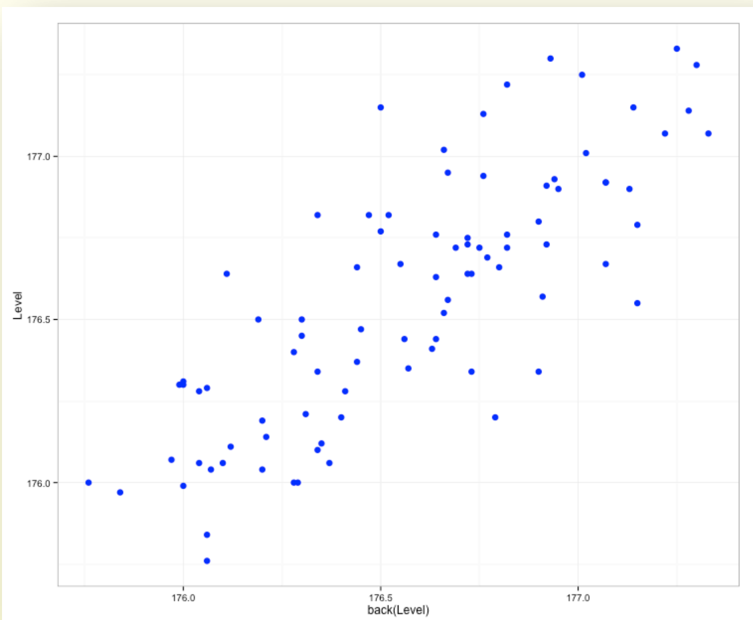
Now let's return to the lake data…

Each point is a pair of adjacent years.
e.g. ($Level_{1929}$, $Level_{1930}$)

```
back(back(Level)
```

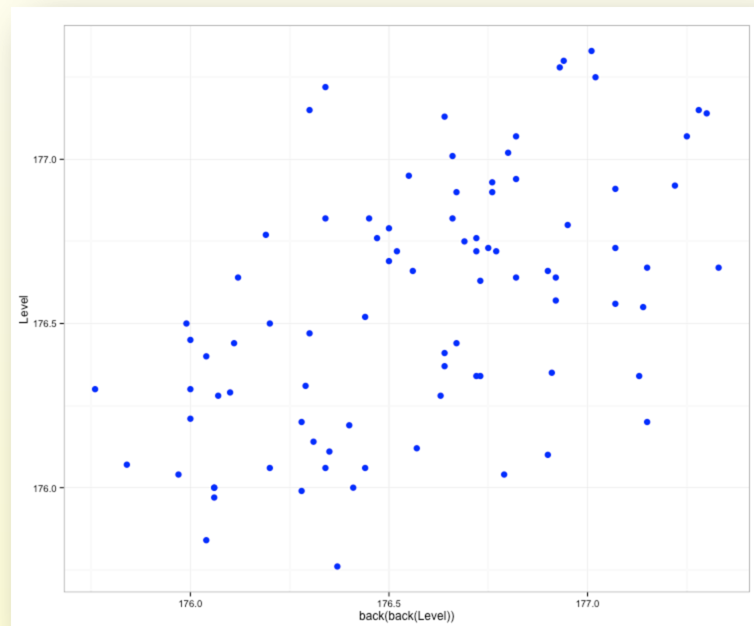
First, let's plot $Level_t$ vs. $Level_{t-1}$

Corr = .794



Now, let's plot $Level_t$ vs. $Level_{t-2}$

Corr = .531



c. Autocorrelation

Time series is about dependence. We use correlation as a measure of dependence.

Although we have only one variable, we can compute the correlation between Y_t and Y_{t-1} or between Y_t and Y_{t-2}.

The correlations between Y's at different times are called **autocorrelations**.

However, we must assume that all the Y's have:

- same mean (no upward or downward trends)
- same variances

c. Autocorrelation

We will assume what is known as **stationarity.**

Roughly speaking this means:

- The time series varies about a fixed mean and has constant variance
- The dependence between successive observations does not change over time

Let's define the autocorrelations for a stationary time series.

$$\boxed{\rho_s = \frac{\operatorname{cov}\left(Y_t, Y_{t-s}\right)}{\sqrt{\operatorname{Var}\left(Y_t\right) \times \operatorname{Var}\left(Y_{t-s}\right)}} = \frac{\operatorname{cov}\left(Y_t, Y_{t-s}\right)}{\operatorname{Var}\left(Y_t\right)}}$$

Note that the autocorrelation does not depend on t because we have assumed stationarity .

c. Autocorrelation

We estimate the theoretical quantities by using sample averages (as always).

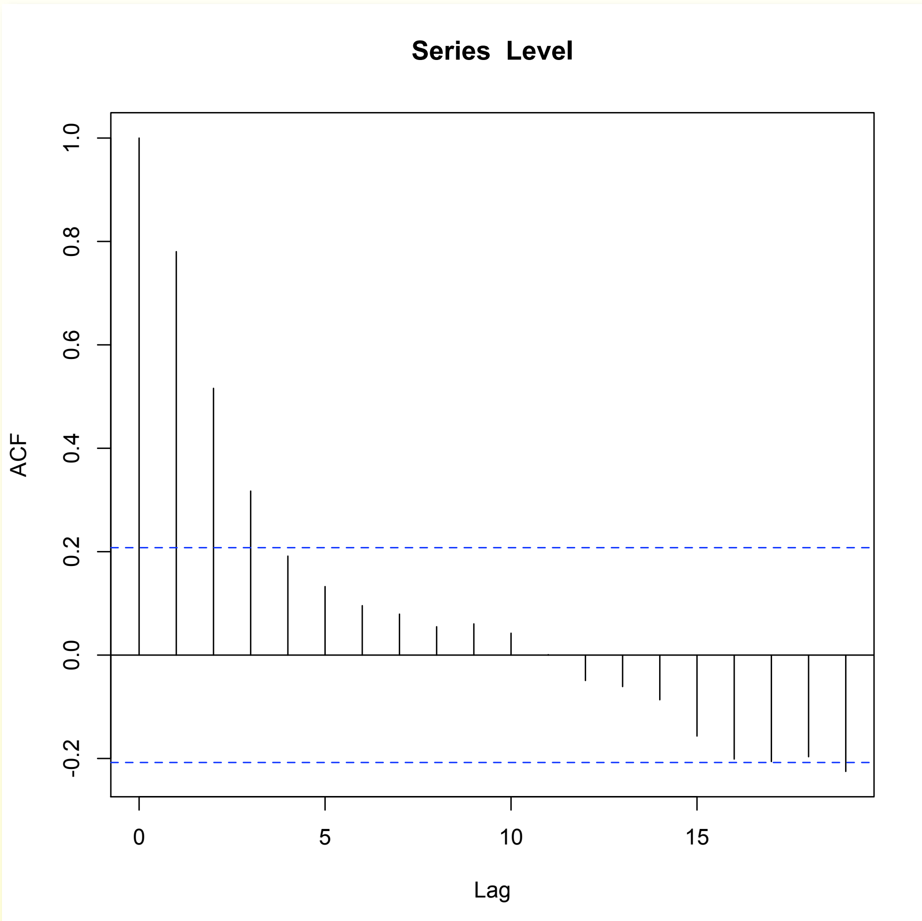
The estimated or **sample autocorrelations** are:

$$\boxed{r_s = \frac{\sum_{t=s}^{T}(Y_t - \bar{Y})(Y_{t-s} - \bar{Y})}{\sum_{t=1}^{T}(Y_t - \bar{Y})^2}}$$

c. Autocorrelation

The ACF command in R computes the autocorrelations

```
> acf(Level)
```

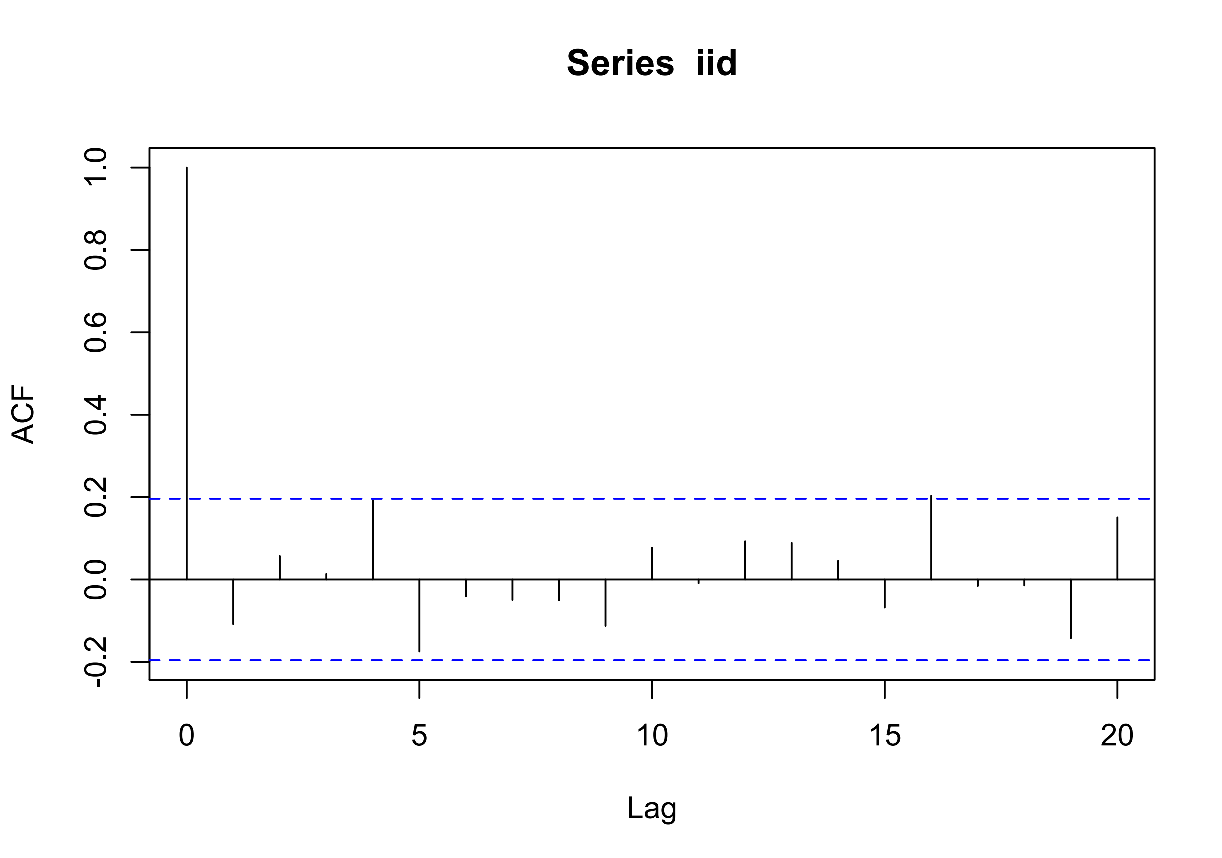


There is a strong dependence between observations spaced close together in time (e.g only one or two years apart). As time passes, the dependence diminishes in strength.

c. Autocorrelation

Let's look at the autocorrelations for the IID series.

```
> acf(iid)
```



In contrast to the ACF for the 'level' series, the sample autocorrelations are much smaller.

c. Autocorrelation

How do we know if the sample autocorrelations are good estimates of the underlying theoretical autocorrelations?

and

How do we know if we have enough sample information to reach definitive conclusions?

If all the true autocorrelations are 0, then the standard deviation of the sample autocorrelations is about 1/sqrt(T).

$$\boxed{\mathrm{Std\,Err}(r_s) = \frac{1}{\sqrt{T}}}$$

c. Autocorrelation

The Box-Ljung test can be used to test the hypothesis that the first L (defined by `lag`) autocorrelations are zero.

```
> Box.test(rnorm(100),type="Ljung",lag=20)

 Box-Ljung test

data:  rnorm(100)
X-squared = 13.9207, df = 20, p-value = 0.8345

> Box.test(lmich_yr$Level,type="Ljung",lag=20)

 Box-Ljung test

data:  lmich_yr$Level
X-squared = 128.698, df = 20, p-value < 2.2e-16
```

d. The AR(1) Model

A simple way to model dependence over time is with the "autoregressive model of order 1."

This is a SLR model of Y_t regressed on lagged Y_{t-1}.

$$\boxed{\text{AR(1)}: \; Y_t = \beta_0 + \beta_1 Y_{t-1} + \varepsilon_t}$$

What does the model say for the T+1 st observation?

$$\boxed{Y_{T+1} = \beta_0 + \beta_1 Y_T + \varepsilon_{T+1}}$$

The AR(1) model expresses what we don't know in terms of what we do know at time T.

d. The AR(1) Model

If we subtract μ from both sides of the AR(1) model equation, we can write the model in terms of deviations from the mean.

$$\boxed{Y_t - \mu = \beta_1 (Y_{t-1} - \mu) + \varepsilon_t}$$

Thus, β_1 governs the rate at which you “revert” to the mean level of the series.

On average, Y_t is closer to the mean than Y_{t-1}.

If there is no mean reversion, then we have a **random walk**.

d. The AR(1) Model

Some Intuition on Mean Reversion

We have seen that the slope parameter governs the rate at which the AR(1) model "returns" or "reverts" to the mean level of the series.

Fact for the AR(1) model:

$$\boxed{E[Y_t] = \mu = \frac{\beta_0}{(1-\beta_1)}}$$

d. The AR(1) Model

How should we predict $Y_{T+1}, Y_{T+2}, \ldots, Y_{T+s}$ given Y_T?

$$E\left[Y_{T+1} | Y_T\right] = \beta_0 + \beta_1 Y_T + E\left[\varepsilon_{T+1} | Y_T\right] = \beta_0 + \beta_1 Y_T$$

$$\hat{Y}_{T+1} = \beta_0 + \beta_1 Y_T$$

$$E\left[Y_{T+2} | Y_T\right] = \beta_0 + \beta_1 E\left[Y_{T+1} | Y_T\right] + E\left[\varepsilon_{T+2} | Y_T\right] = \beta_0 + \beta_1 \hat{Y}_{T+1}$$

$$\hat{Y}_{T+2} = \beta_0 + \beta_1 \hat{Y}_{T+1}$$

$$\vdots$$

$$\hat{Y}_{T+s} = \beta_0 + \beta_1 \hat{Y}_{T+s-1}$$

d. The AR(1) Model

How do we use the AR(1) model? We simply regress Y on lagged Y.

If our model successfully captures the dependence structure in the data then the residuals should look iid. There should be no dependence in the residuals!

So to check the AR(1) model, we can check the residuals from the regression for any “left-over” dependence.

d. The AR(1) Model

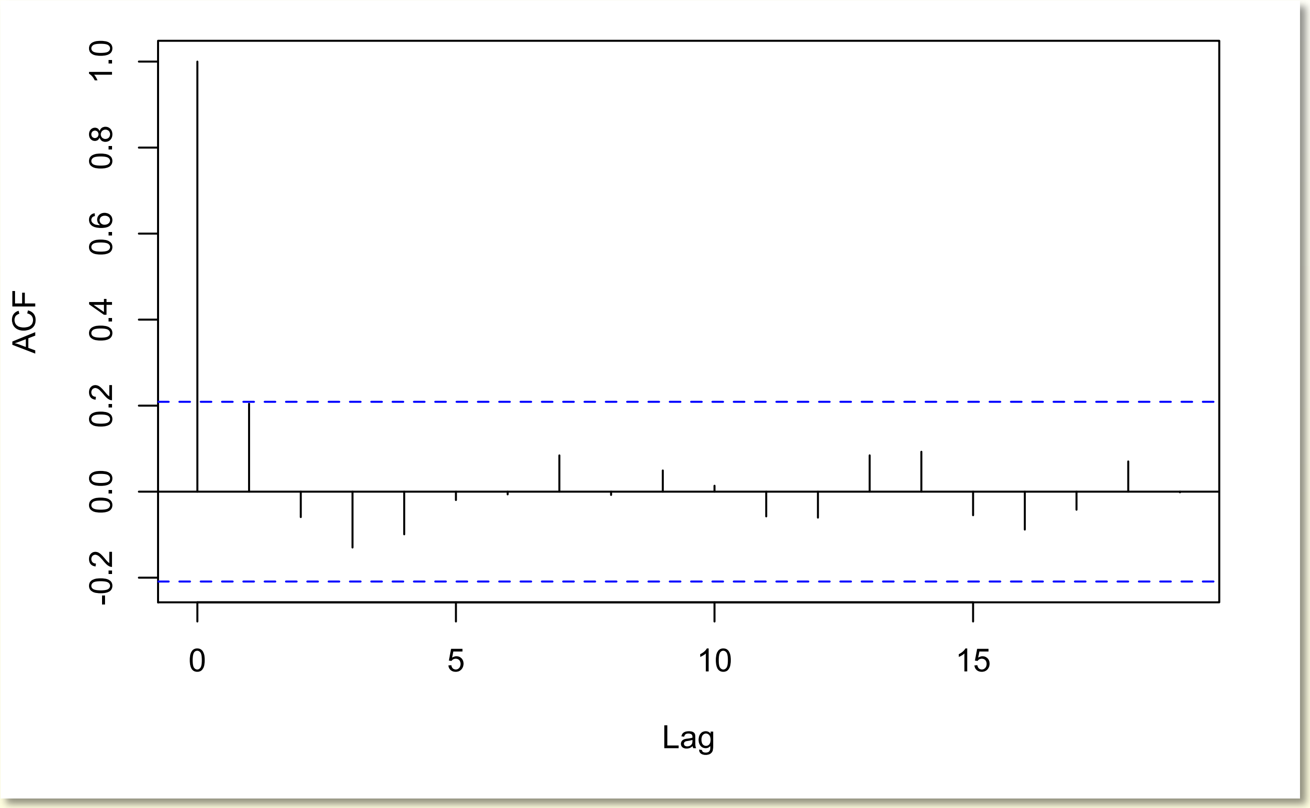
Let's try it out on the lake water level data...

```
> lmSumm(lm(Level~back(Level),data=lmich_yr))
Multiple Regression Analysis:
    2 regressors(including intercept) and 88 observations

lm(formula = Level ~ back(Level), data = lmich_yr)

Coefficients:
              Estimate Std Error t value p value
(Intercept)   36.7900  11.55000    3.18   0.002
back(Level)    0.7916   0.06543   12.10   0.000
---
Standard Error of the Regression:  0.2362
Multiple R-squared:  0.63  Ad
Overall F stat: 146.39 on 1 ar
```

```
> Box.test(lm(lmich_yr$Level~back(lmich_yr$Level))$res,type="Ljung",lag=20)

 Box-Ljung test

data:  lm(lmich_yr$Level ~ back(lmich_yr$Level))$res
X-squared = 19.7586, df = 20, p-value = 0.4731
```

d. The AR(1) Model

Now let's look at the ACF of the residuals…



Nothing much left!

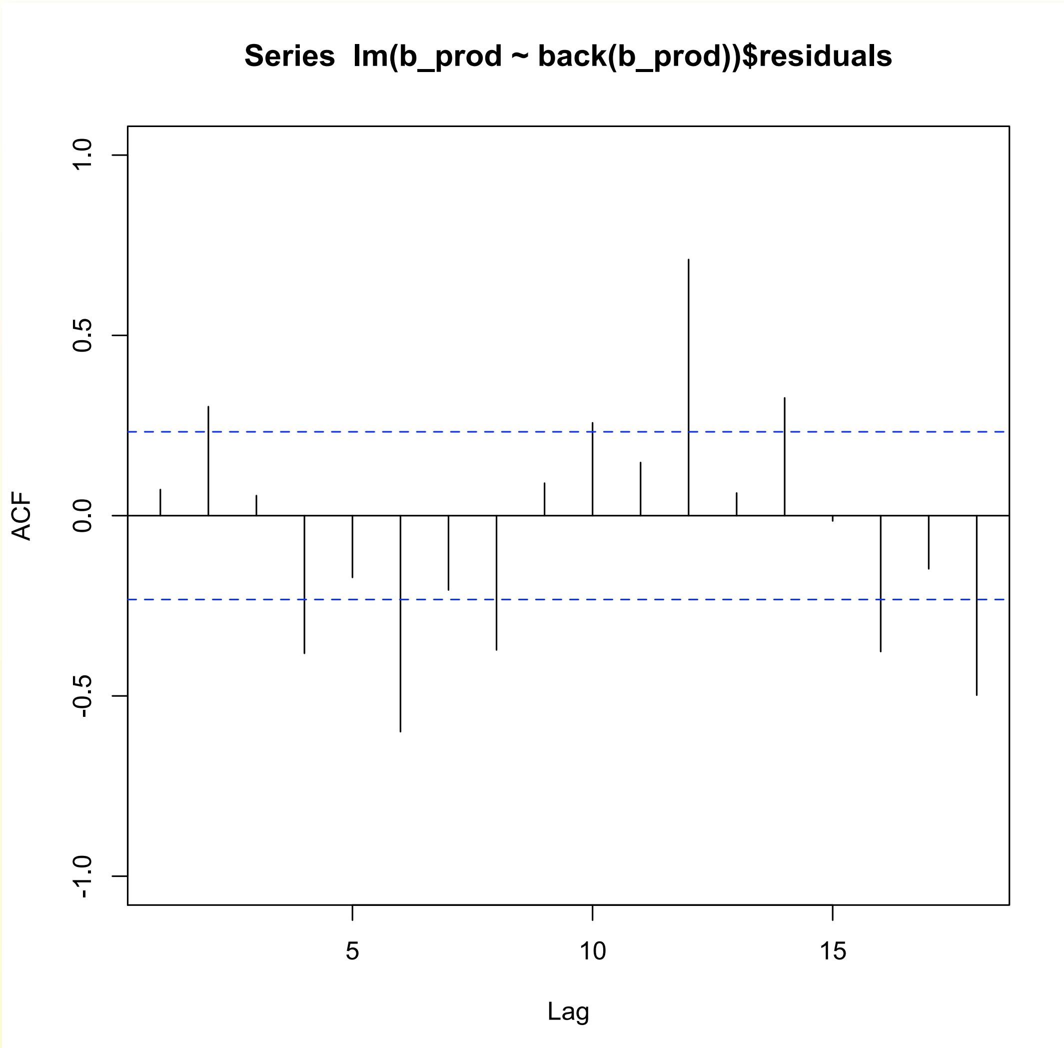
d. The AR(1) Model

Now let's try the beer data…

```
> data(beerprod)
> lmSumm(lm(b_prod~back(b_prod),data=beerprod))
Multiple Regression Analysis:
    2 regressors(including intercept) and 71 observations

lm(formula = b_prod ~ back(b_prod), data = beerprod)

Coefficients:
             Estimate Std Error t value p value
(Intercept)    4.7780   1.42500    3.35   0.001
back(b_prod)   0.7043   0.08724    8.07   0.000
---
Standard Error of the Regression:  1.386
Multiple R-squared:  0.486  Adjusted R-squared:  0.478
Overall F stat: 65.18 on 1 and 69 DF, pvalue= 0
```

d. The AR(1) Model

Now let's look at the ACF of the residuals…



There's a lot of auto-correlation left in. Why at lag 6 and 12?

d. The AR(p) Model

A natural generalization of the AR(1) model is the AR(p) model:

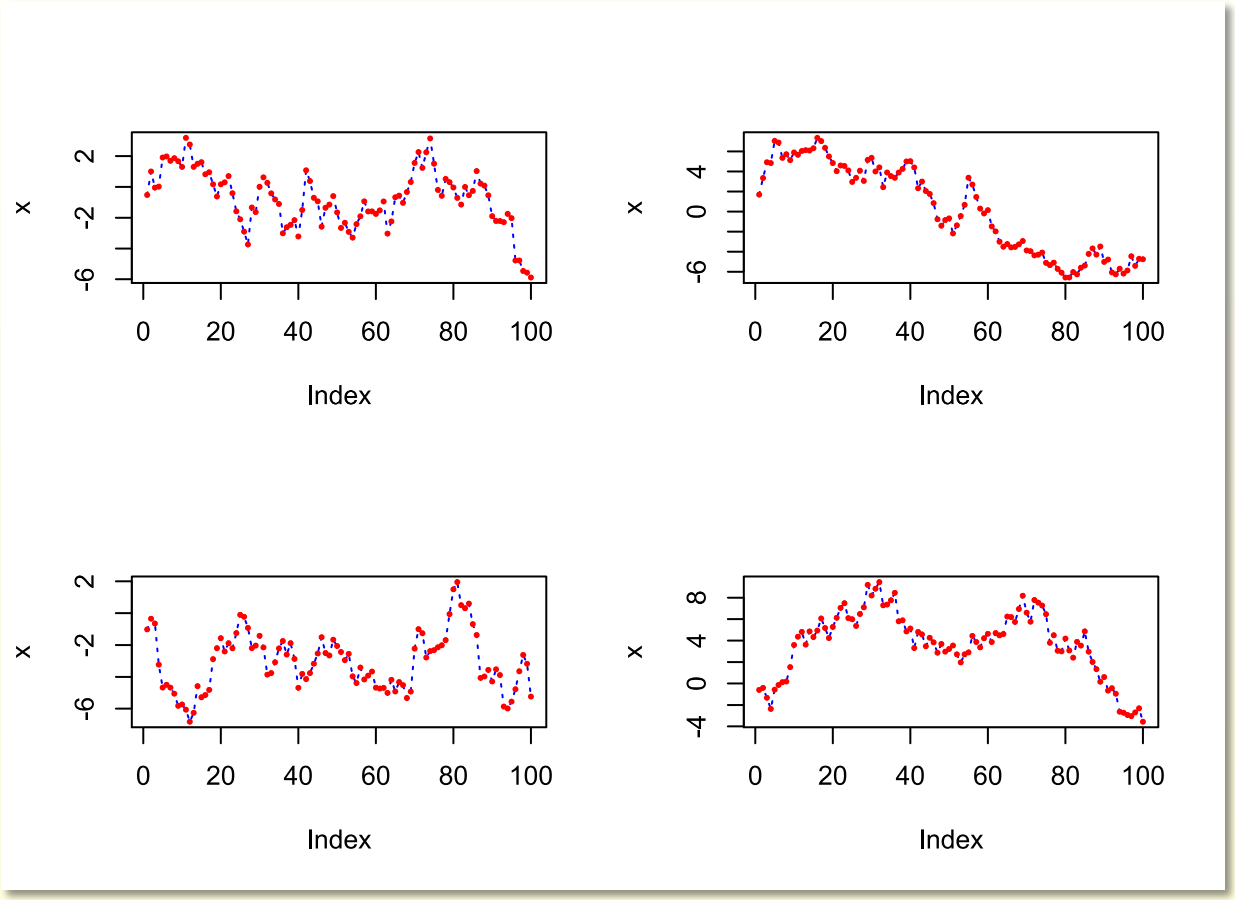
$$\boxed{Y_t = \beta_0 + \beta_1 Y_{t-1} + \ldots + \beta_p Y_{t-p} + \varepsilon_t}$$

How do you select p?

Fit AR(1)
Check residuals for autocorrelation using `acf()`
If uncorrelated
    stop
else
    add order (e.g. go from AR(1) to AR(2))

e. Random Walks

Now let's look at a series generated with a slope value of 1…

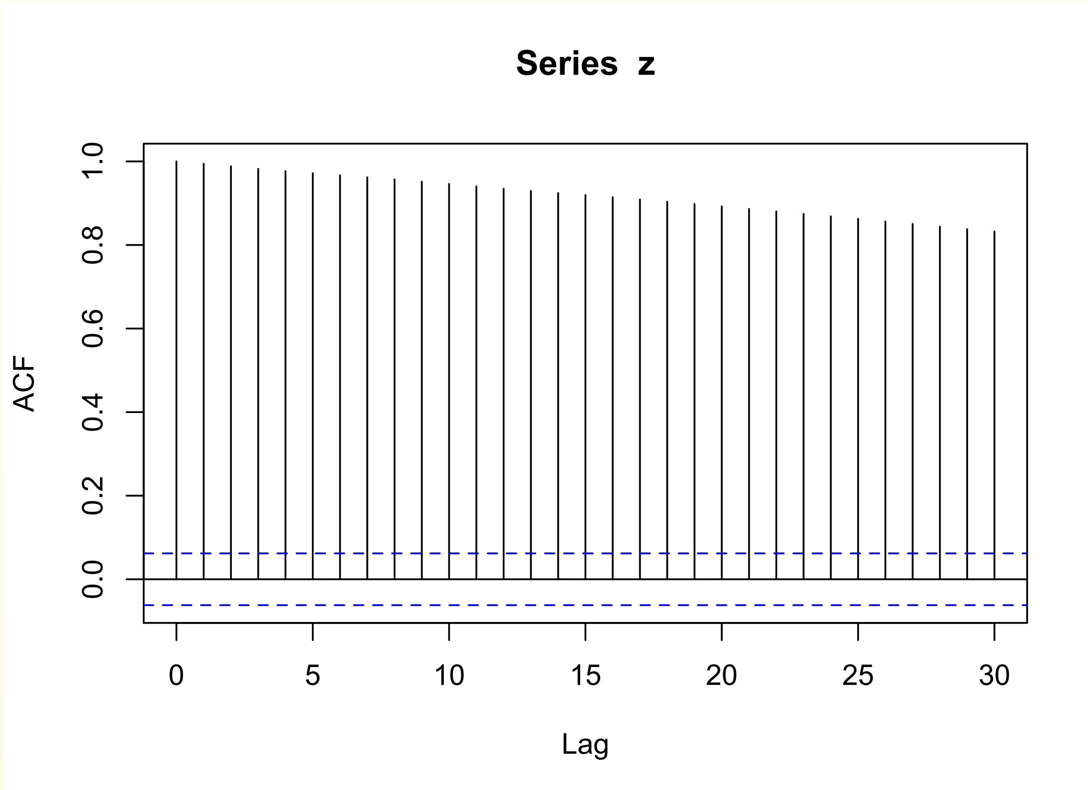


$$\beta_0 = 0$$
$$\beta_1 = 1.0$$

Wanders around quite a lot! Can exhibit what appears to be trends.

e. Random Walks

What about the ACF?



The first autocorrelation is close to 1. Does that mean the series is very predictable? We will return to the case of $\beta_1 = 1$ shortly

e. Random Walks

The case of $\beta_1 = 1$ deserves special attention because of it's importance in economic data series. Many economic and business time series display a "random walk character."

A random walk is an AR(1) model with $\beta_1 = 1$

Random Walk:

$$\boxed{Y_t = \beta_o + Y_{t-1} + \varepsilon_t}$$

The intercept, β_0 , is called the drift parameter for the random walk. Let's first consider the case of $\beta_0 = 0$.

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

e. Random Walks

The random "walk" gets its name from the idea of a random walker on the number line. A random walker is someone who has an equal chance of taking a step forward or a step backward. The size of the steps are random as well.

To see this, it is very useful to re-express the random walk in term of *increments* or steps. Subtract Y_{t-1} from both sides,

$$\boxed{Z_t = Y_t - Y_{t-1} = \varepsilon_t}$$

The increments are an random sample (iid collection of rvs)!

e. Random Walks

A random walk with zero drift:

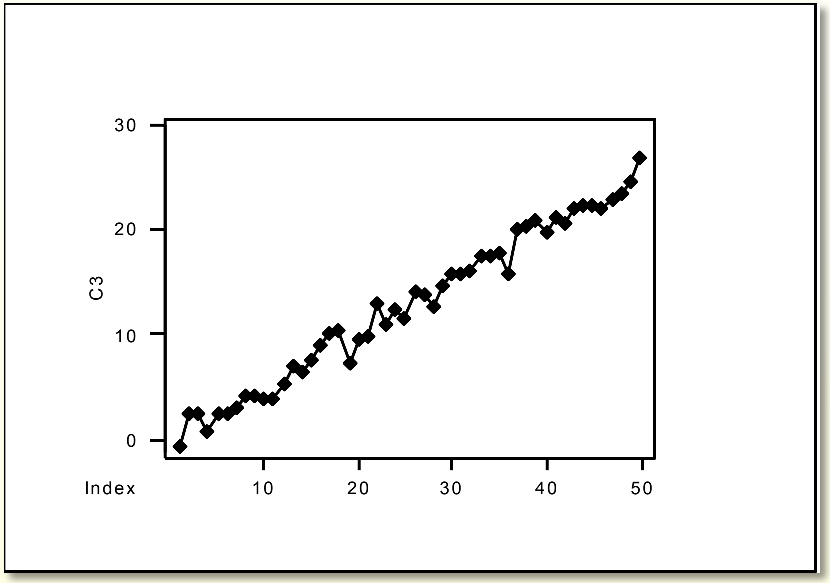
- "meanders" around zero with no particular trend.
- can take long "excursions" away from zero that look like trends.
- A zero drift will always return to zero.

If β_0 is positive, we have a random walk with positive drift and will not return to zero.

Here the average step size is β_0

f. Trend Models

Many times we want to allow for shifts in the mean of series over time.

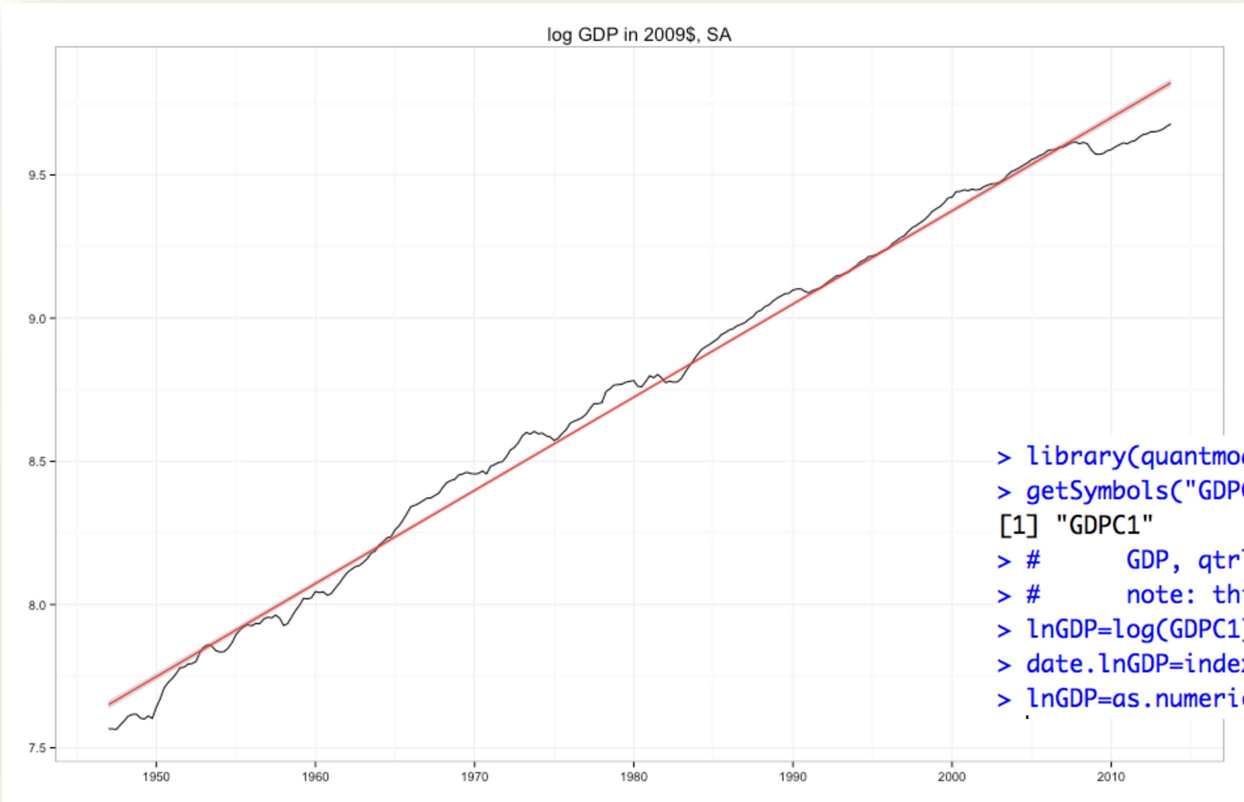


<u>Linear Trend Model:</u>

$$\boxed{Y_t = \beta_0 + \beta_1 t + \varepsilon_t}$$

Error terms are assumed independent or un-autocorrelated. This means there are no correlated deviations from trend, i.e. if you are below trend this period, you are as likely to above trend as below next period.

f. Trends and Random Walks



Is the graph from a trend or a random walk with a positive drift?

```
> library(quantmod)
> getSymbols("GDPC1",src = "FRED")
[1] "GDPC1"
> #      GDP, qtrly, seasonally adjusted, 2009 $
> #      note: this is an xts object not a ts object!
> lnGDP=log(GDPC1)
> date.lnGDP=index(lnGDP)
> lnGDP=as.numeric(lnGDP)
```

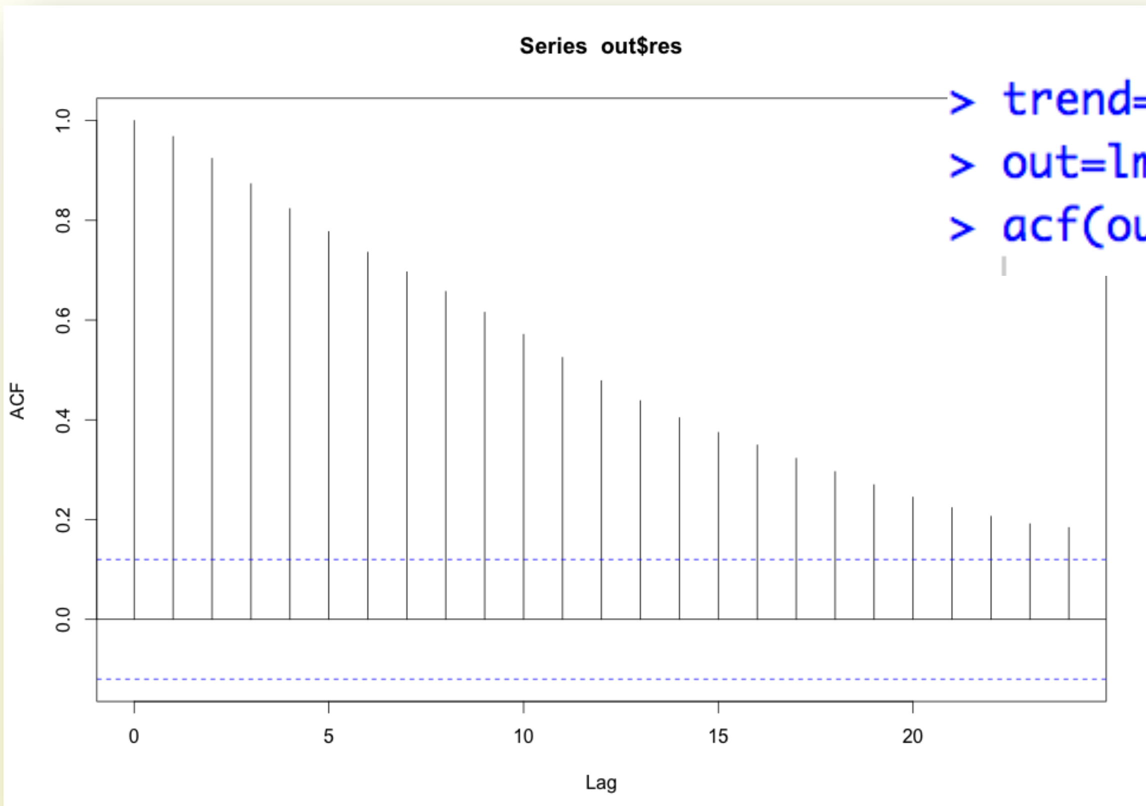
$$\boxed{Y_t = \beta_0 + Y_{t-1} + \varepsilon_t}$$

$$\boxed{Y_t = \beta_0 + \beta_1 t + \varepsilon_t}$$

or something else?

f. Random Walks and Trends

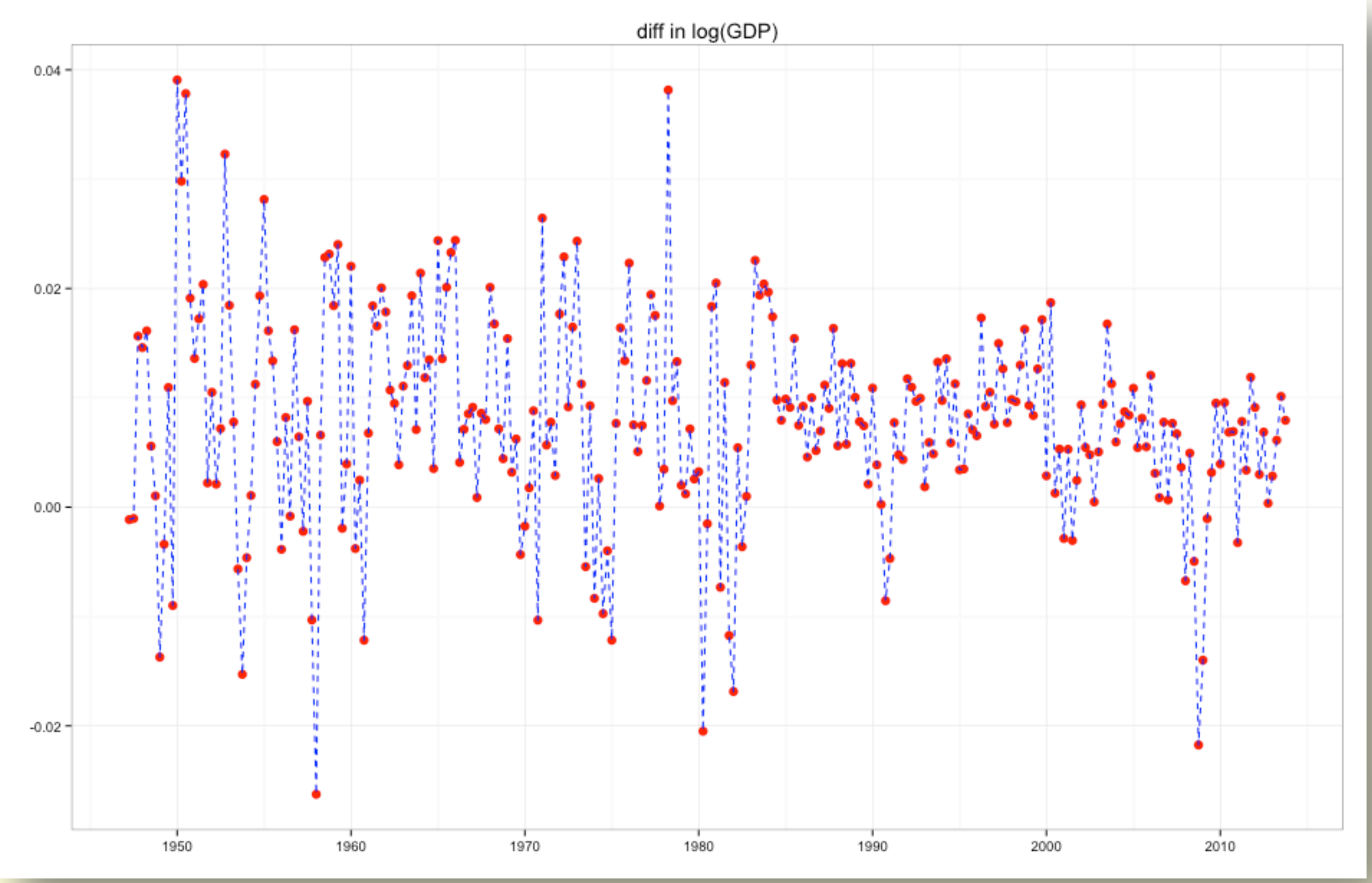
Let’s run the regression for the trend fit and look at residual acf. Looks pretty auto-correlated! Trend Model is not appropriate. Random walk might be more appropriate.



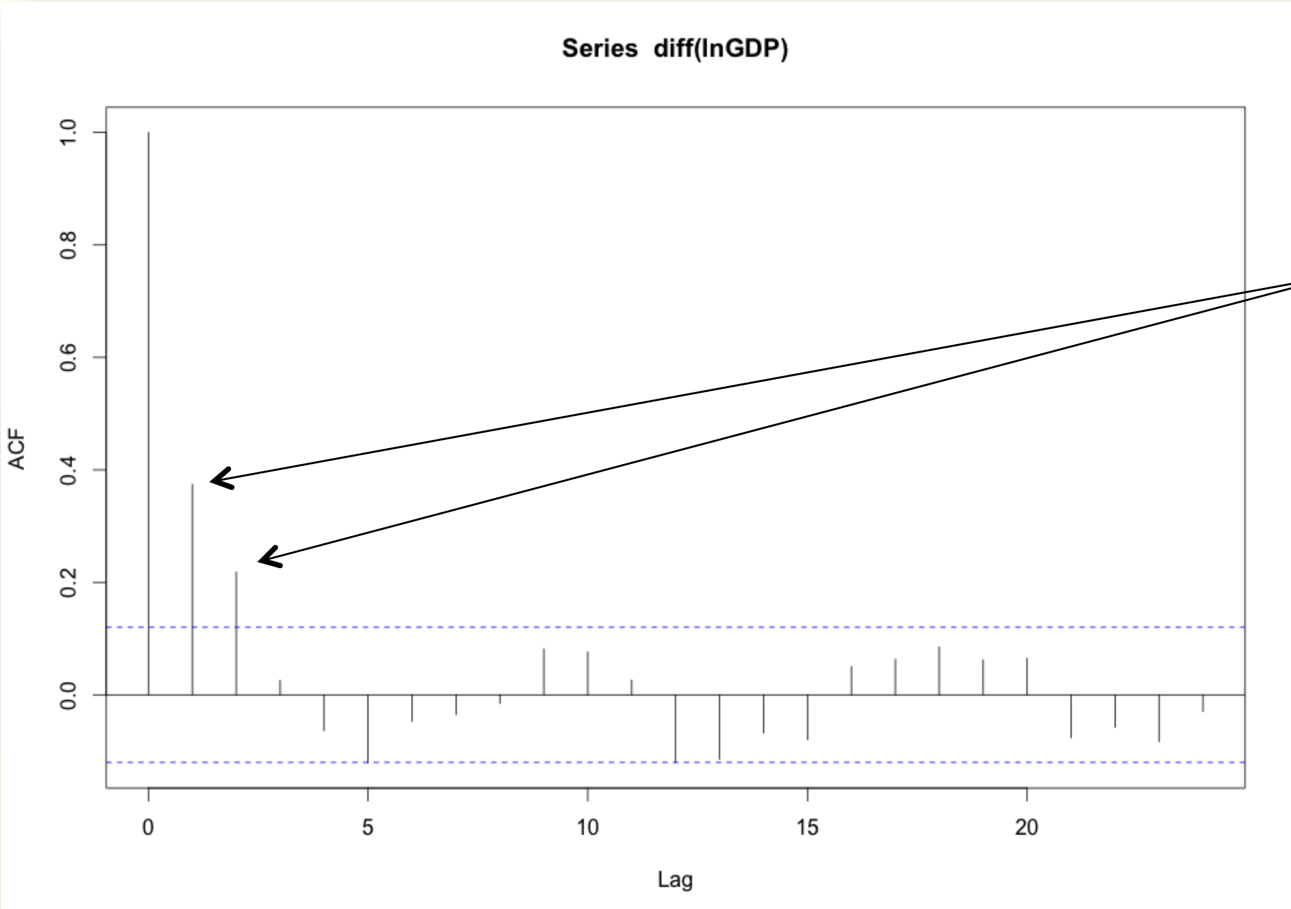

```
> trend=1:length(lnGDP)
> out=lm(lnGDP~trend)
> acf(out$res)
```

f. Random Walks and Trends

Let's look at the differences in log(GDP).



What about the acf of the differences?



Looks a bit auto-correlated

f. Random Walks and Trends

Let's use an AR model on the differences! This is called an ARIMA(1,1,0) model.

$$Y_t^{diff} = \beta_0 + \beta_1 Y_{t-1}^{diff} + \varepsilon_t$$

$$Y_t^{diff} = Y_t - Y_{t-1}$$

This kind of model can be fitted and predicted from using regressions with the differences variables.

$$\hat{Y}_{t+1} = Y_t + \hat{Y}_{t+1}^{diff}$$

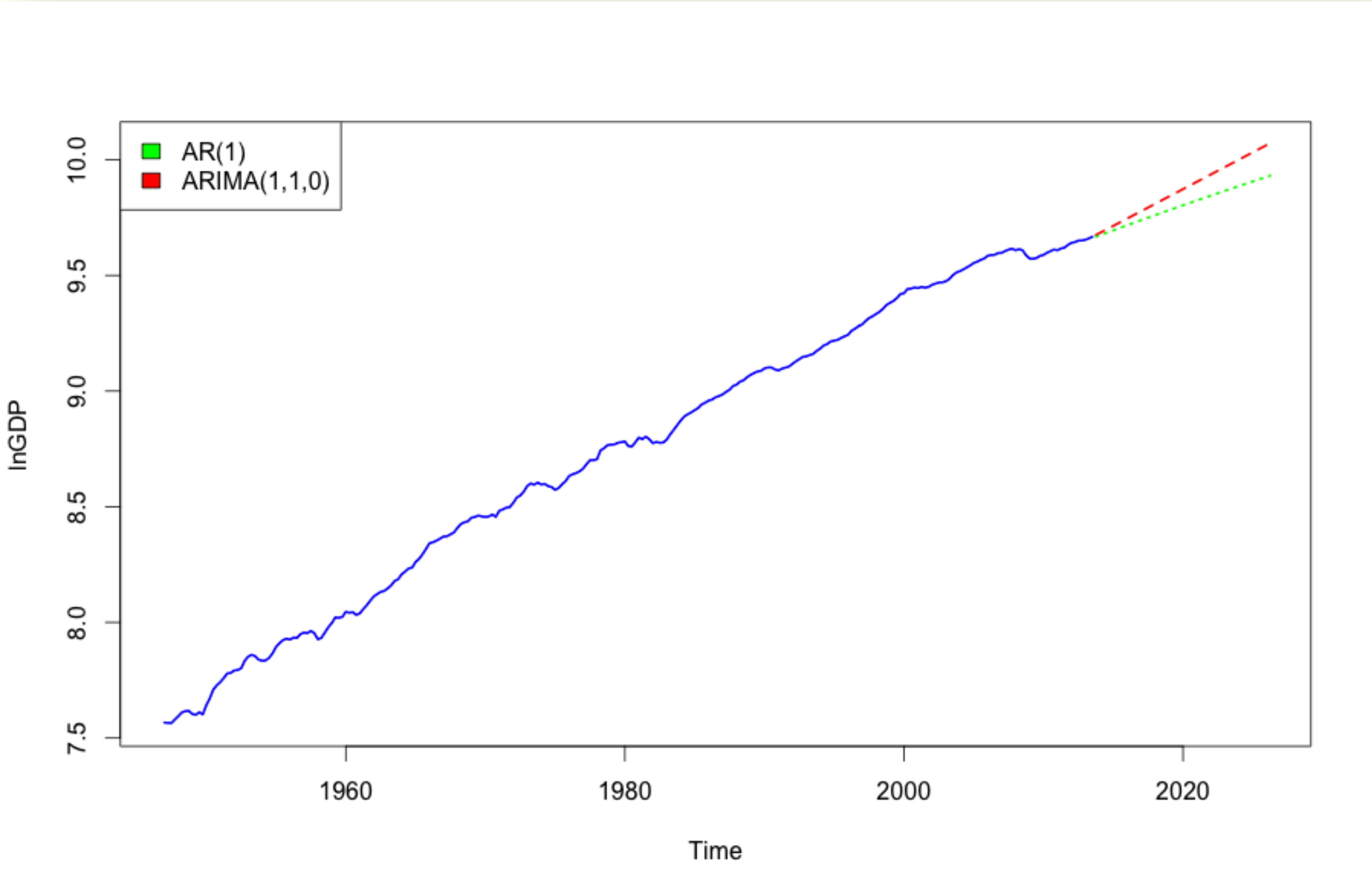
$$\hat{Y}_{t+2} = Y_t + \hat{Y}_{t+1}^{diff} + \hat{Y}_{t+2}^{diff}$$

$$\vdots$$

$$\hat{Y}_{t+s} = Y_t + \hat{Y}_{t+1}^{diff} + \ldots + \hat{Y}_{t+s}^{diff}$$

f. Random Walks and Trends

Compare forecasts from AR(1) and differenced AR(1) – called ARIMA(1,1,0).



f. Random Walks and Trends

We must compute a "forecast" profile or compute forecasts out many periods ahead using our fitted model. To do so, we must make R "roll-forward" forecasts from the AR(1) model.

That is, predict one period ahead.

Then use one period ahead forecast to forecast two periods ahead.

Start with T and predict T+1

```
pred.ar[1] = b0 + b1* Y_T
```

Then predict T+2 given T+1,

```
pred.ar[2] = b0 + b1*pred.ar[1]
```

Then predict T+3 given T+2,

```
pred.ar[3] = b0 + b1*pred.ar[2]
```

Then predict T+4 given T+3,

```
pred.ar[4] = b0 + b1*pred.ar[3]
```

and so on!

f. Random Walks and Trends

To do this we need a “loop” in R. A loop is a way of repeating R commands based on a counter index and using that index in the loop.

Basic structure

```
for(i in 1:nstep){
       << R commands that may depend on i >>
}
```

Start with i=1,

```
pred.ar[2] = b0 + b1*pred.ar[1]
```

Then set i=2,

```
pred.ar[3] = b0 + b1*pred.ar[2]
```

Then set i=3,

```
pred.ar[4] = b0 + b1*pred.ar[3]
```

and so on!

f. Random Walks and Trends

Here is the code:

```
# fit the model first
lnGDP=as.vector(lnGDP)
out.ar=lm(lnGDP~back(lnGDP))

nstep=50
pred.ar=double(nstep+1)
pred.ar[1]=lnGDP[length(lnGDP)] # last period (T)

for(i in 1:nstep){
        pred.ar[i+1] = out.ar$coef[1]+out.ar$coef[2]*pred.ar[i]
}
```

b_0 b_1

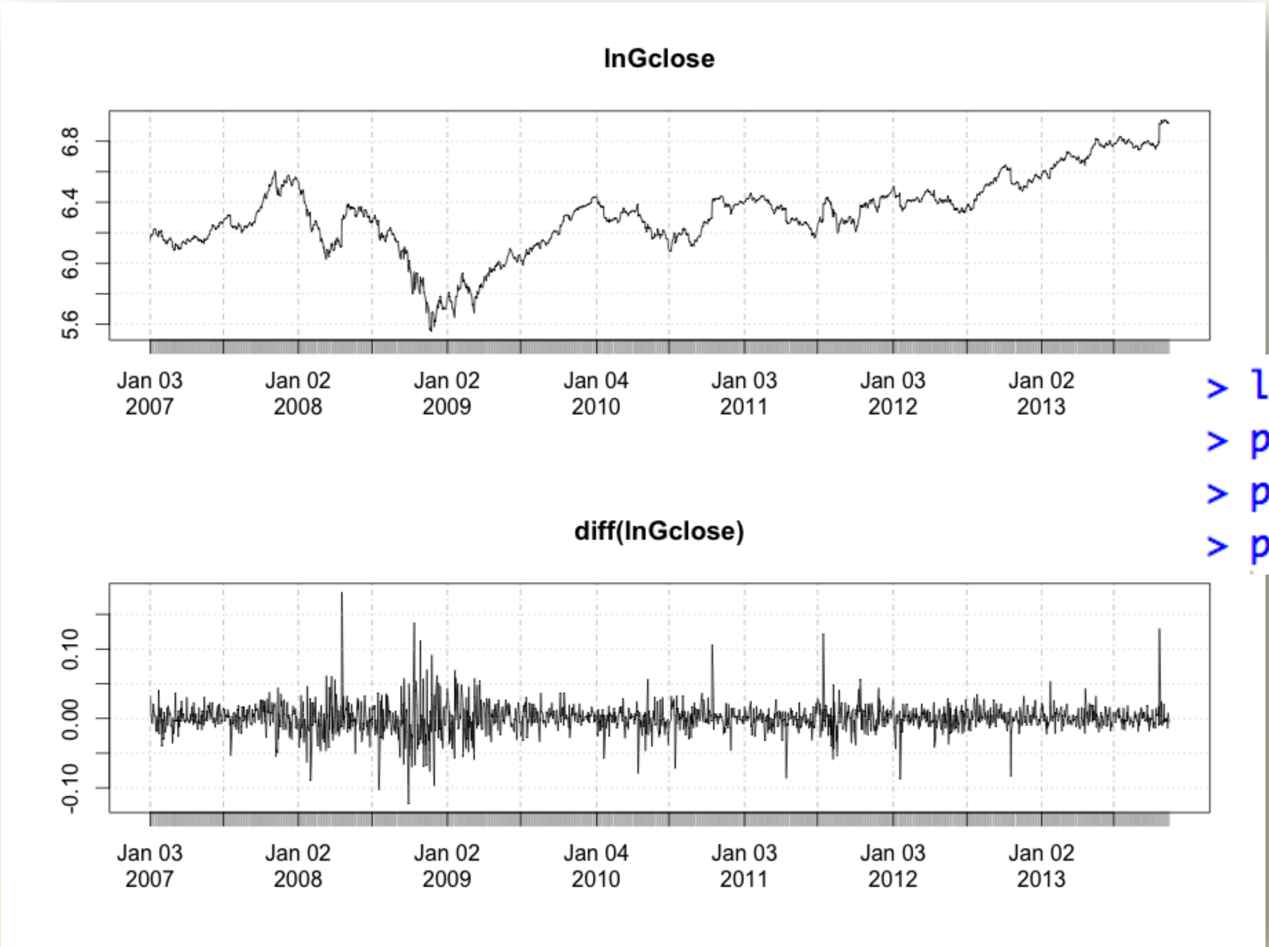
f. Random Walks and Trends

Now do the predictions from the AR(1) on the differences:.

```
#
# now fit AR(1) on the differences
#
out.arima=lm(diff(lnGDP)~back(diff(lnGDP)))
nstep=50
out.ar=lm(lnGDP~back(lnGDP))
pred.arima=double(nstep+1)
pred.arima[1]=lnGDP[length(lnGDP)]-lnGDP[length(lnGDP)-1]

for(i in 1:nstep){

    pred.arima[i+1] =
            out.arima$coef[1]+out.arima$coef[2]*pred.arima[i]
}
pred.arima=pred.arima[-1]  # all but the first one

pred.arima=lnGDP[length(lnGDP)]+cumsum(pred.arima)
```

g. Stock Prices and Market Efficiency

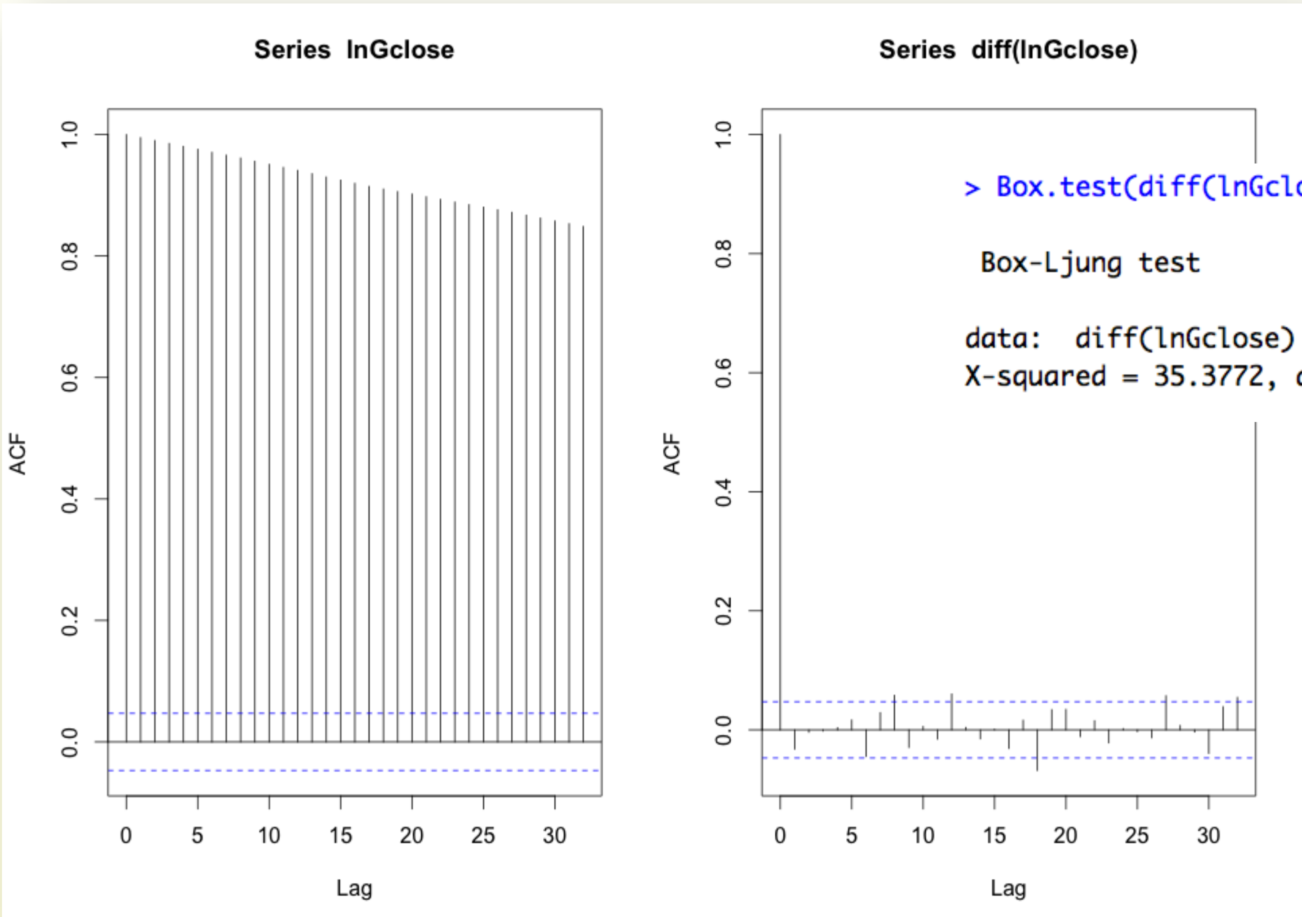
Stock price series present an interesting example of a time series. Let's look at log daily close of Google




```
> lnGclose=log(GOOG[,4])
> par(mfrow=c(2,1))
> plot(lnGclose)
> plot(diff(lnGclose))
```

g. Stock Prices and Market Efficiency

Now check out acf's




```
> Box.test(diff(lnGclose),type="Ljung",lag=20)

 Box-Ljung test

data:  diff(lnGclose)
X-squared = 35.3772, df = 20, p-value = 0.01819
```

Anything predictable about diff in log prices?

g. Stock Prices and Market Efficiency

It turns out that a model that fits many stock prices series is a random walk in the log of prices.

$$\log\left(p_t\right) = \alpha + \log\left(p_{t-1}\right) + \varepsilon_t$$

or

$$\log\left(p_t\right) - \log\left(p_{t-1}\right) = \alpha + \varepsilon_t$$

$$\log\left(1 + \frac{\Delta p_t}{p_{t-1}}\right) = \log\left(1 + \%\Delta p_t\right) \approx R_t$$

g. Stock Prices and Market Efficiency

Thus, if the log of stock prices follows a random walk, the changes in the log of the price are independent.

This has profound implications for the ability to predict future changes in stock prices. This says that stock price changes are completely independent of past changes.

This strongly suggests that any trading strategy that involves the past history of price changes can’t work (momentum etc.).

g. Stock Prices and Market Efficiency

What is the economic meaning of this finding?

One possible explanation is that stock prices reflect all available information at the time of trade. Competitive markets provide a sort of information aggregation mechanism by which information relevant to the stock price (e.g. future profitability of the firm) is incorporated into price.

This idea is often called the weak market efficiency hypothesis. This idea goes back to the fundamental principle of conditional prediction – i.e. forecast errors (changes in price) must be uncorrelated with any information available at the time of the forecast.

h. Building Time Series Models for Prediction

The basic idea of building time series models is

1). Build a model (e.g. AR(p)) that extracts the information from the past history of the variable

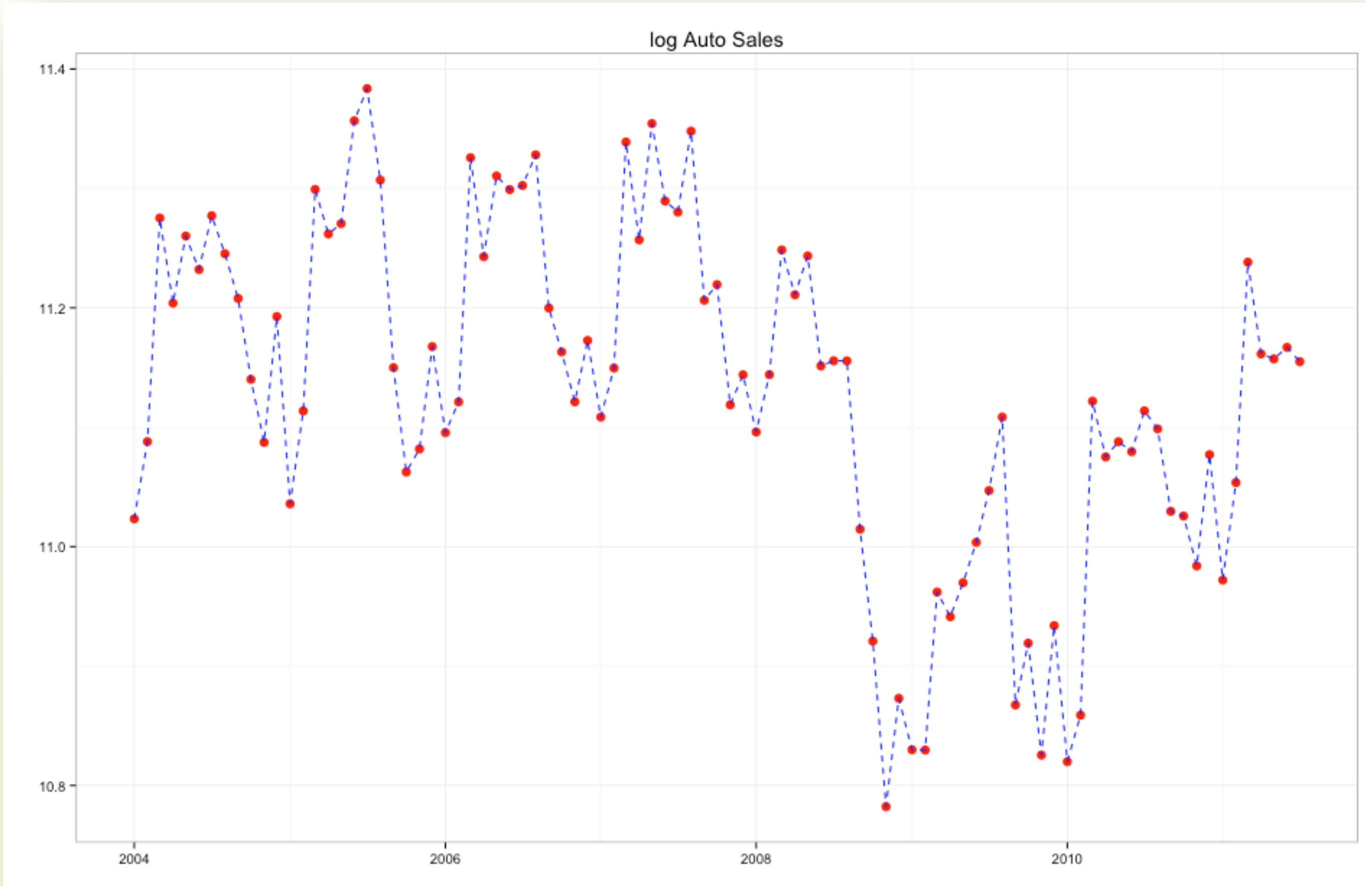
2). Then, and only then, bring in other variables to help predictions.

Example: predict US Auto sales

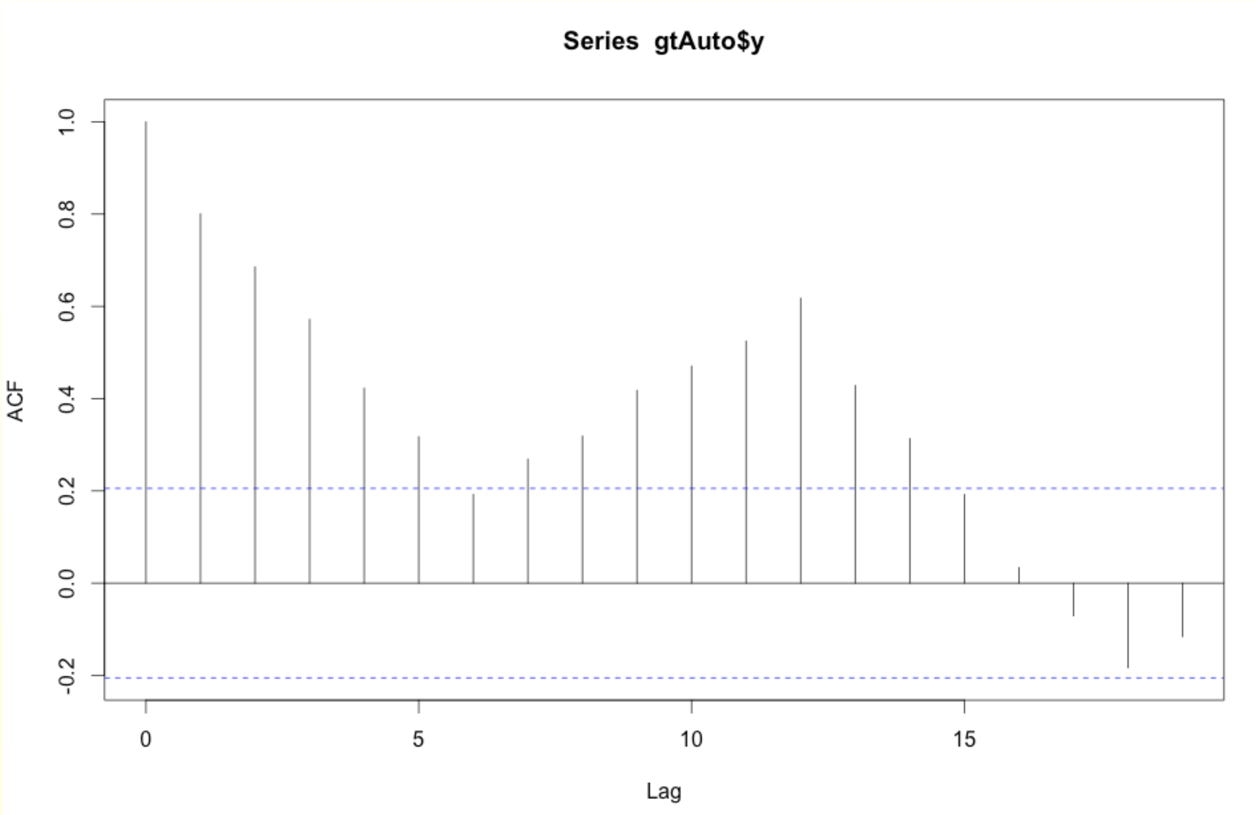
1) First, model lag structure
2) Second, bring in Google Trends data

h. Building Time Series Models for Prediction

Fetch monthly census data on auto sales.



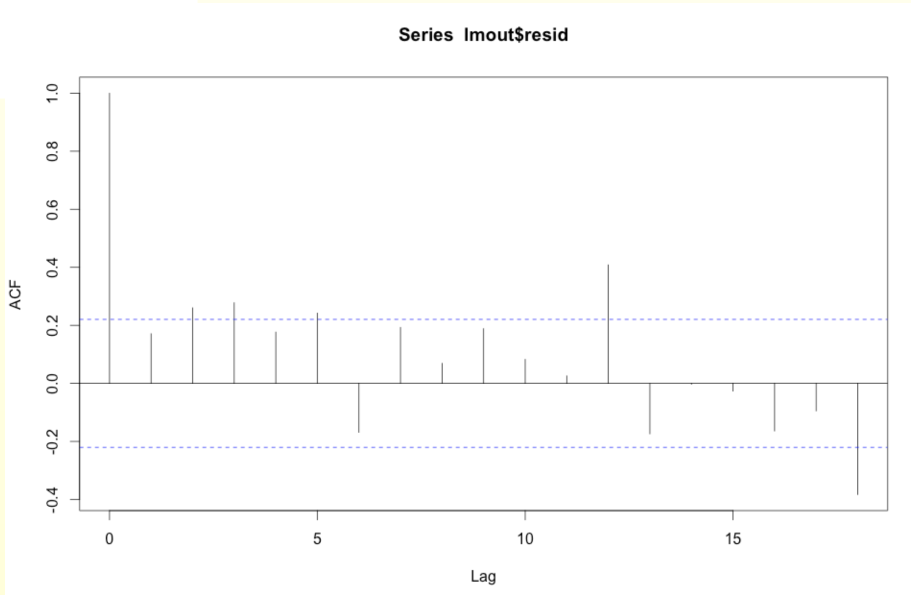
h. Building Time Series Models for Prediction



acf suggests adding lag one and lag 12 terms.

h. Building Time Series Models for Prediction

```
> lmSumm(lmout)
Multiple Regression Analysis:
    3 regressors(including intercept) and 79 observations

lm(formula = y ~ back(y) + back(y, 12), data = gtAuto)

Coefficients:
             Estimate Std Error t value p value
(Intercept)    0.6727   0.76360    0.88   0.381
back(y)        0.6435   0.07332    8.78   0.000
back(y, 12)    0.2957   0.07282    4.06   0.000
---
Standard Error of the Regression:  0.07985
Multiple R-squared:  0.719  Adjusted R-squared:
Overall F stat: 97 on 2 and 76 DF, pvalue= 0
```



h. Building Time Series Models for Prediction

Now add in Google Trends search index data.

```
> lmSumm(lmout_trends)
Multiple Regression Analysis:
    5 regressors(including intercept) and 79 observations

lm(formula = y ~ back(y) + back(y, 12) + suvs + insurance, data = gtAuto)

Coefficients:
            Estimate Std Error t value p value
(Intercept)  -0.4580   0.78440   -0.58   0.561
back(y)       0.6195   0.06318    9.81   0.000
back(y, 12)   0.4287   0.06535    6.56   0.000
suvs          1.0570   0.16690    6.34   0.000
insurance    -0.5297   0.15210   -3.48   0.001
---
Standard Error of the Regression:  0.06509
Multiple R-squared:  0.818  Adjusted R-squared:  0.808
Overall F stat: 83.08 on 4 and 74 DF, pvalue= 0
```

h. Building Time Series Models for Prediction

Let's fit and forecast one step ahead from both "baseline" and model with Google Trend data.

Start at time 17,

1. estimate the model with first 17 observations
2. predict 18^{th} observation

Move to time 18,

1. estimate the model with first 18 observations
2. predict the 19^{th} observation

.

.

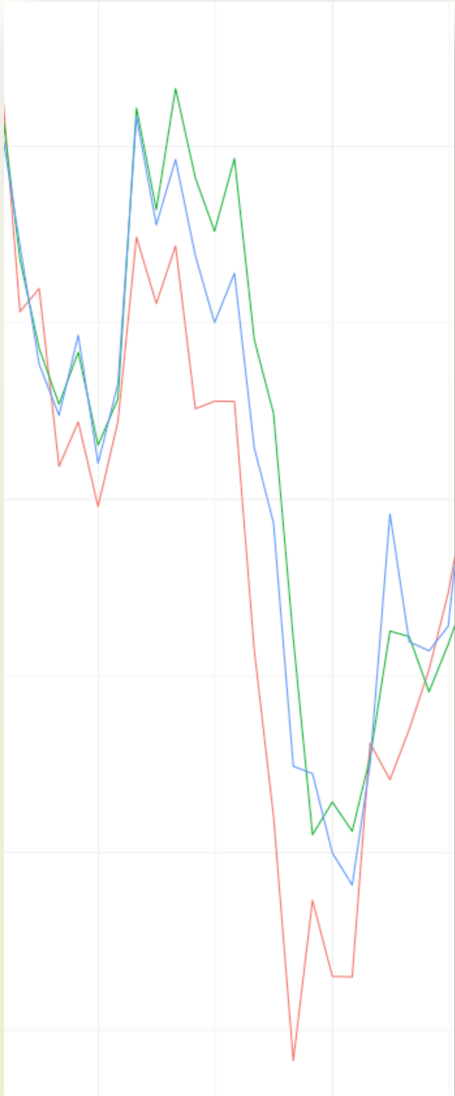
.

h. Building Time Series Models for Prediction

```
n = length(gtAuto$y)
k=17 #start with only the first 17 months
gtAuto$y.lag1 = back(gtAuto$y)
gtAuto$y.lag12 = back(gtAuto$y,12)

for (t in k:(n-1)) {
  # roll forward the regressions
  reg1 = lm(y~y.lag1+y.lag12,data=gtAuto[1:t,])
  reg2 = lm(y~y.lag1+y.lag12+suvs+insurance,
          data=gtAuto[1:t,])
  t1 = t+1
  gtAuto$Actual[t1] = gtAuto$y[t1]
  gtAuto$Baseline[t1] = predict(reg1,newdata=gtAuto[t1,])
  gtAuto$Trends[t1] = predict(reg2,newdata=gtAuto[t1,])
}
```

h. Building Time Series Models for Prediction

h. Building Time Series Models for Prediction



Does the model with trends do better than baseline model?

Trends model captures downturns better.

h. Building Time Series Models for Prediction

Mean Absolute Error (the primary alternative to RMSE) can be used to evaluate model fit. By what percent does the Trends model beat the baseline in MAE?

```
> mae1<-mean(abs(exp(z$Actual)-exp(z$Baseline))/exp(z$Actual))
> mae2<-mean(abs(exp(z$Actual)-exp(z$Trends))/exp(z$Actual))
> mae2/mae1-1
[1] -0.1149367
```

Compare to in-sample fit.

```
> ActualSales=gtAuto$sales[13:length(gtAuto$sales)]
> mae1_insam=mean(abs(ActualSales-exp(lmout_base$fitted)))
> mae2_insam=mean(abs(ActualSales-exp(lmout_trends$fitted)))
> mae2_insam/mae1_insam-1
[1] -0.1660874
```

Glossary of Symbols

ρ_s - sth order autocorrelation

r_s - sth order sample autocorrelation

Important Equations

$$\rho_s = \frac{\operatorname{cov}(Y_t, Y_{t-s})}{\sqrt{\operatorname{Var}(Y_t) \times \operatorname{Var}(Y_{t-s})}} = \frac{\operatorname{cov}(Y_t, Y_{t-s})}{\operatorname{Var}(Y_t)}$$

$$r_s = \frac{\sum_{t=s}^{T} (Y_t - \bar{Y})(Y_{t-s} - \bar{Y})}{\sum_{t=1}^{T} (Y_t - \bar{Y})^2}$$

Population and Sample Autocorrelations

$$\operatorname{Std Err}(r_s) = \frac{1}{\sqrt{T}}$$

Std error of sample autocorrelation

Important Equations

$$\boxed{AR(1):\ Y_t = \beta_0 + \beta_1 Y_{t-1} + \varepsilon_t}$$

definition of AR(1) model

$$\boxed{Y_t - \mu = \beta_1 \left(Y_{t-1} - \mu\right) + \varepsilon_t}$$

Mean Reversion form of AR(1)

$$\boxed{Y_t = \beta_o + Y_{t-1} + \varepsilon_t}$$

Random Walk

Glossary of R Commands

- `acf()`: Computes (and by default plots) estimates of the autocorrelation function
- `back()`: Computes a lagged version of a time series, shifting the time base back once.
- `diff()` : Returns the differences between a value and its lagged value.
- `c(1:30)`: Generates 30 numbers from 1 to 30 with increment of 1
- `arima(x,order=c(p,d,q))`: fits an arima(p,d,q) model