

# MATH 615: Approaches to Quantitative Analysis in the Life Sciences

## Class 8

Autocorrelation, the bane of ecology and evolution graduate students

---

### Packages

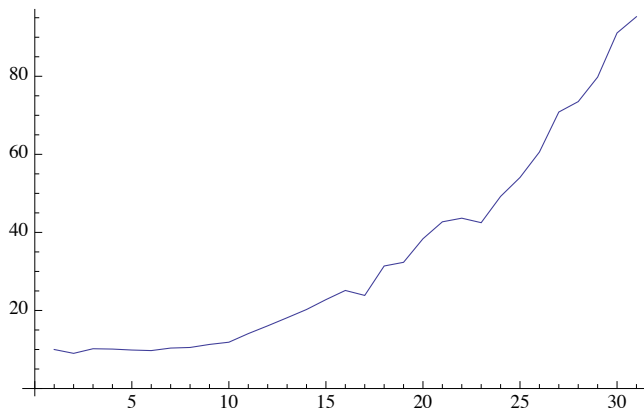
---

#### Cormorants and fish

```
cormorantNumbers = NestList[# + # * RandomReal[NormalDistribution[0.1, 0.1]] &, 10, 30]
```

```
{10, 9.01204, 10.1949, 10.1087, 9.86294, 9.72176, 10.3673, 10.52, 11.2975, 11.8681, 14.0716,  
16.064, 18.1348, 20.2448, 22.7757, 25.1252, 23.8544, 31.3925, 32.3363, 38.3713, 42.7122,  
43.6443, 42.492, 49.2218, 54.0757, 60.587, 70.8388, 73.509, 79.7878, 91.098, 95.2668}
```

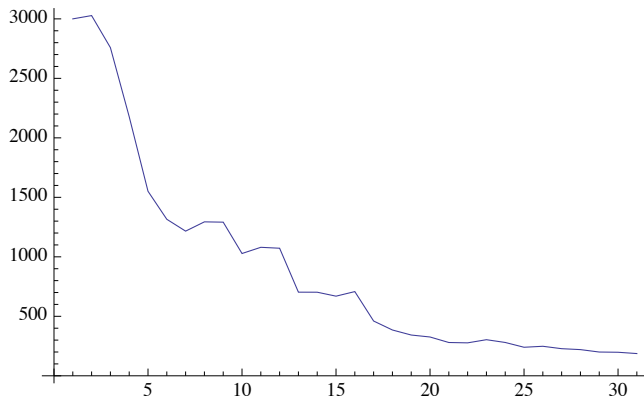
```
ListLinePlot[cormorantNumbers]
```



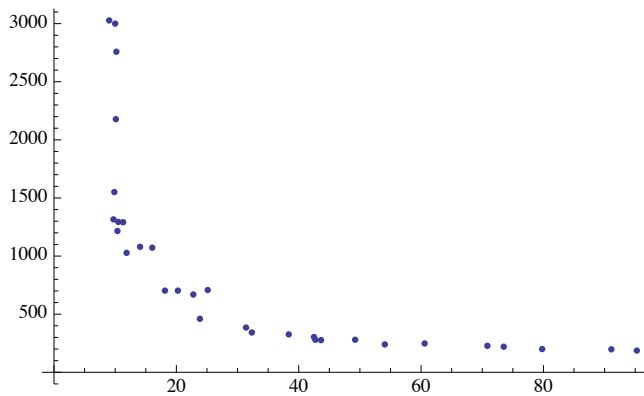
```
fishNumbers = NestList[# + # * RandomReal[NormalDistribution[-0.1, 0.1]] &, 3000, 30]
```

```
{3000, 3027.64, 2758.34, 2177.71, 1550.97, 1315.54, 1215.94, 1293.91, 1291.27, 1027.93, 1079.98,  
1072.18, 702.891, 702.556, 669.36, 707.743, 460.404, 384.985, 342.717, 326.149, 279.848,  
276.92, 303.443, 280.099, 239.873, 248.132, 227.924, 220.115, 199.77, 197.854, 186.929}
```

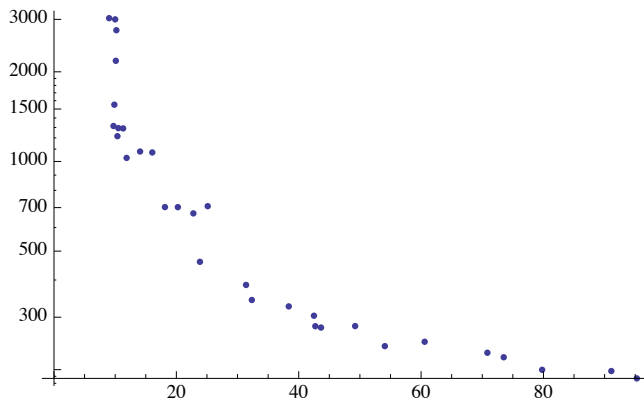
ListLinePlot[fishNumbers]



ListPlot[Transpose[{cormorantNumbers, fishNumbers}]]



ListLogPlot[Transpose[{cormorantNumbers, fishNumbers}]]



(Apparent) conclusion: increasing cormorants are associated with decreasing fish. And we *know* that cormorants eat fish, so...

---

## Random walks

These are simple functions with first-order autocorrelation — each value in the sequence depends on the previous value, with some error term added. Note that in every case, the error term has zero mean, so there is no inherent tendency for values to increase or decrease relative to the one before.

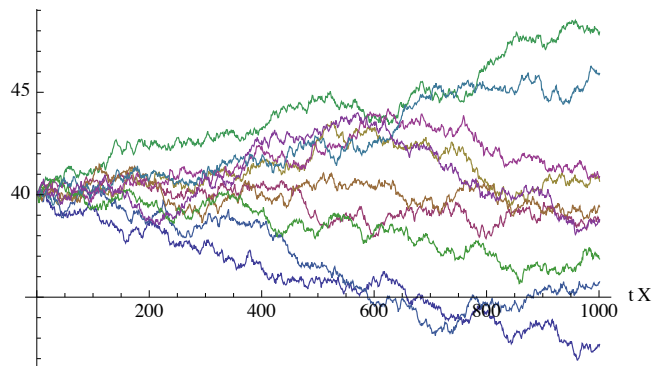
## Random walk

In this case, the error term is all there is

$$X_t = X_{t-1} + \epsilon_t$$

Here are 10 random walks

```
randomWalks = Table[NestList[#+ RandomReal[NormalDistribution[0, 0.1]] &, 40, 1000], {10}];
ListLinePlot[randomWalks, PlotRange -> All,
  PlotStyle -> Thickness[0.002], AxesLabel -> {"t", "X"}]
```



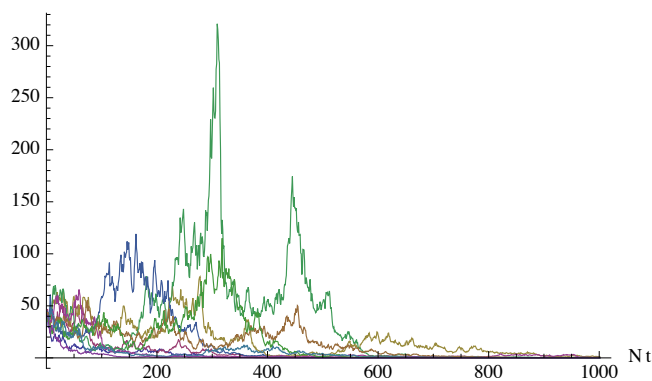
What do you notice about these timeseries? What would happen if you fit a linear regression to each of them?

## ■ Random population growth

In a population-type model, each value — a population size — is the previous population size multiplied by a growth factor  $\lambda$  (because reproduction is density-dependent — the more individuals there are, the more offspring they can have). The stochastic element can be thought of as affecting that multiplier term.

$$N_t = N_{t-1} (\lambda + \epsilon_t)$$

Here are some runs with  $\lambda = 1$ .



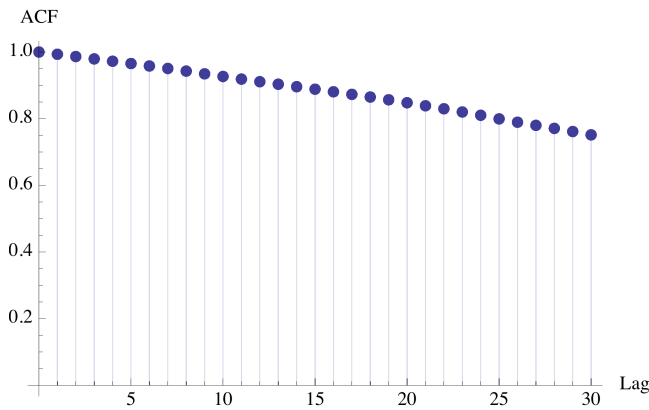

---

## Autocorrelation functions

### ■ Non-package functions

### Autocorrelation function

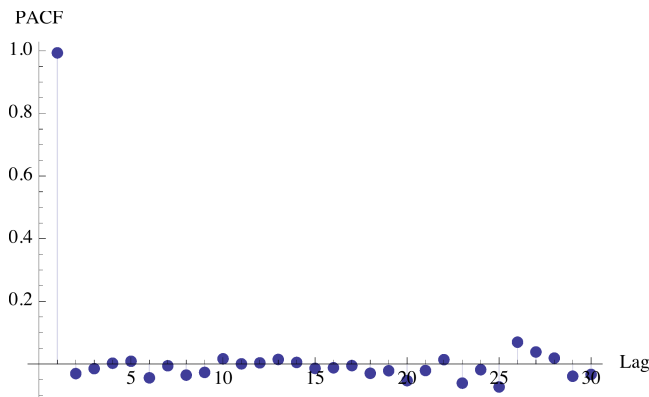
This measures how each value is related to the values 1, 2, 3, ... steps previously. Let's apply it to one of the random walks from the previous section.



Autocorrelation coefficients are always 1 for lag 0, and decline from there. Why does the function decrease so gently? It's because if  $X_t$  is closely related to  $X_{t-1}$ , and  $X_{t-1}$  is closely related to  $X_{t-2}$ , then  $X_t$  is also fairly closely related to  $X_{t-2}$  (and so on).

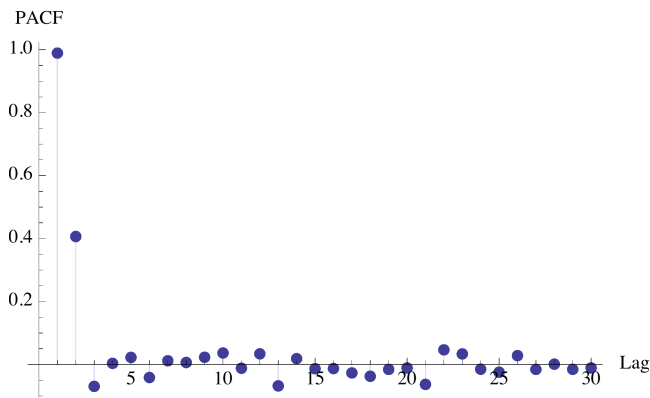
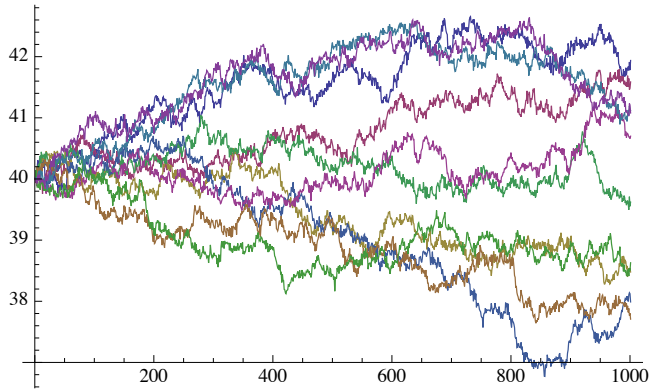
### Partial autocorrelation function

Partial autocorrelation corrects for the serial dependence of autocorrelation coefficients. For each lag, we factor out the effects of all lower-order (shorter) lags.

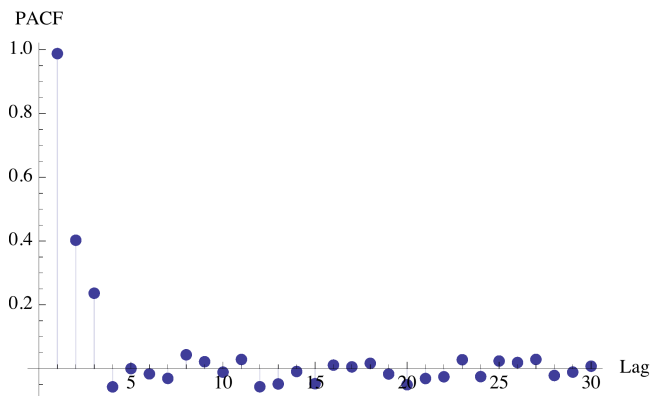
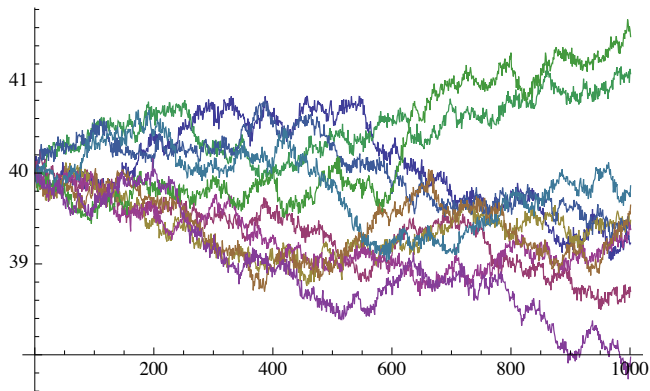


This plot, which starts at lag 1 (not lag 0), is very informative, as it reveals a single large positive coefficient at lag 1. This matches the model that was used to create the data; the model had only a first order autoregressive term. The PACF has 'discovered' the original model!

Let's test the ability of the PACF to discover models. Here are data from a second-order model (lags 1 and 2).



And from a third-order model (lags 1, 2 and 3).

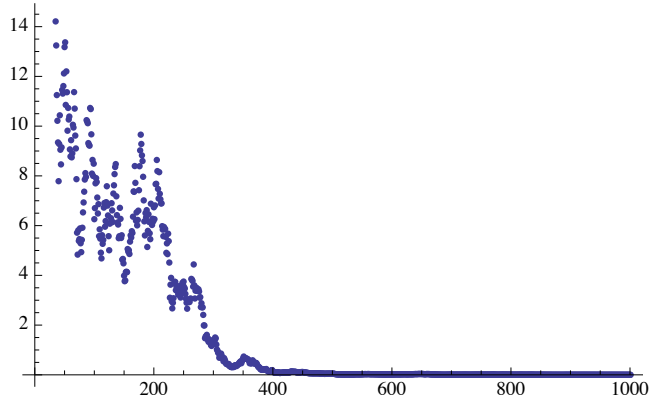


In each case, the PACF reveals the dependencies. It is a powerful exploratory tool.

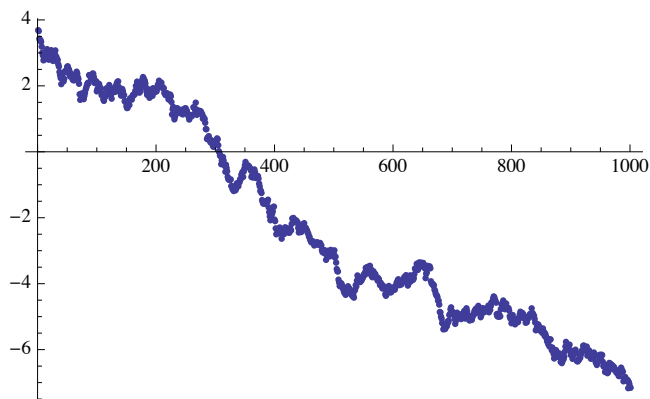
---

## How to tell if a population is growing or declining — the *wrong* way

```
ListPlot[randomPopulationWalks[[1]]]
```



```
ListPlot[Log[randomPopulationWalks[[1]]]]
```



```
linFit = LinearModelFit[Table[{i, Log[randomPopulationWalks[[1, i]]],
  {i, 1, Length[randomPopulationWalks[[1]]]}], {x}, {x}]
```

```
FittedModel[ 2.95504 - 0.0104504 x ]
```

```
linFit["ParameterPValues"][[2]]
```

```
1.148097462286419 × 10-658
```

This is hugely significant. But how do we square that with the fact that we know that the data came from a model with no bias up or down? Maybe it's chance (a Type 1 error). Let's check the  $p$ -values for all 10 random datasets.

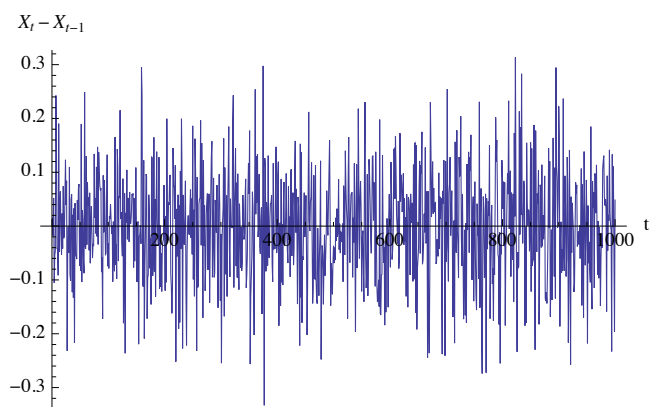
```
Table[LinearModelFit[Table[{i, Log[randomPopulationWalks[[j, i]]],
  {i, 1, Length[randomPopulationWalks[[j]]]}], {x}, {x}][
  "ParameterPValues"][[2]], {j, 1, Length[randomPopulationWalks]}]
{1.148097462286419 × 10-658, 3.113075692421734 × 10-581, 3.64639 × 10-200,
  9.14476 × 10-242, 6.801134561582374 × 10-369, 7.95414 × 10-75, 8.174003553053065 × 10-400,
  1.381282287728087 × 10-383, 7.853821719366238 × 10-375, 7.645796692266344 × 10-648}
```

All significant! That can't be due to chance. But is it correct?

## Removing autocorrelation

### ■ Random Walk

```
lag1DifferenceRandomWalk = Rest[randomWalks[[2]]] - Most[randomWalks[[2]]];
```

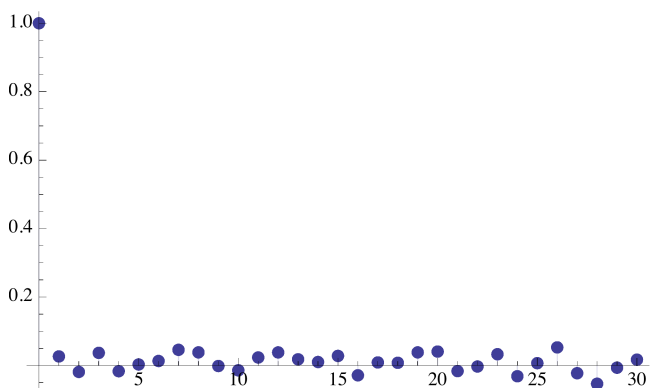


```
Mean[lag1DifferenceRandomWalk]
StandardDeviation[lag1DifferenceRandomWalk]
```

```
- 0.00127875
```

```
0.101441
```

This is the autocorrelation function of the differenced data



Now, all lag terms greater than 0 are negligible. The differences are uncorrelated with each other. Is this expected? We know the model that created the data:

$$X_t = X_{t-1} + \epsilon_t$$

We can rearrange this as follows:

$$X_t - X_{t-1} = \epsilon_t$$

which shows us that taking the difference is a way of recovering the error term, which was not correlated.

### Random population walk

For our population model, taking the difference doesn't work because there is a multiplication on the right hand side:

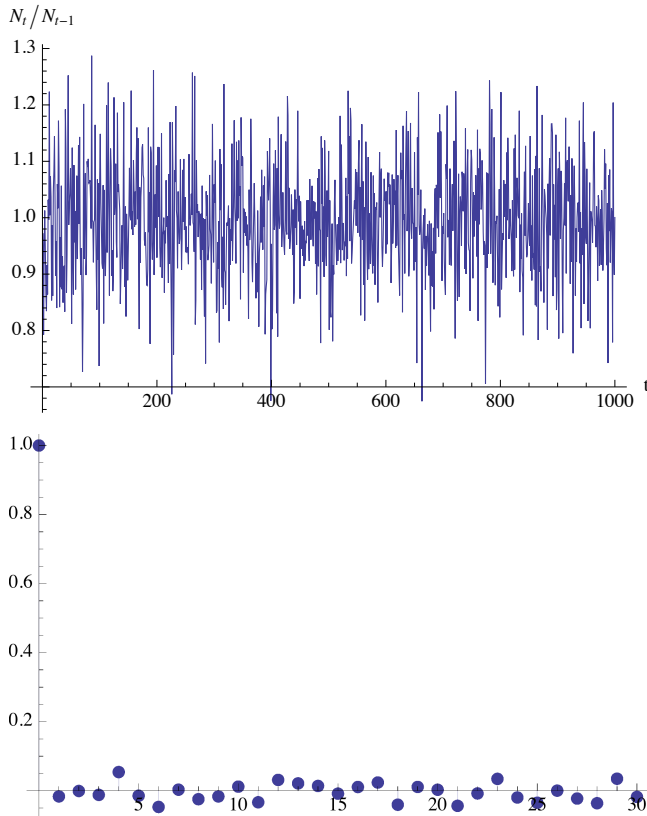
$$N_t = N_{t-1} (\lambda + \epsilon_t)$$

Instead, we take the ratio

$$\frac{N_t}{N_{t-1}} = (\lambda + \epsilon_t)$$

These ratios should be uncorrelated, and distributed with mean  $\lambda$  and distribution given by whatever distribution was used for  $\epsilon$ . Let's see.

```
lag1DifferenceRatioPopulationWalk =
  Rest[randomPopulationWalks[[1]]] / Most[randomPopulationWalks[[1]]];
```



Once again, we have uncorrelated random variates. Note that the mean of them is approximately 1:

```
Mean[lag1DifferenceRatioPopulationWalk]
0.994097
```

This is an estimate of  $\lambda$ .

---

## How to tell if a population is growing or declining — the *right* way

So, if our processing of the population data does recover the random variates  $\lambda + \epsilon_t$ , then we can simply ask if there is evidence that their mean differs from one. In practice, it is normal to take the log transformation of the ratio, and ask if this differs from zero (remember,  $\log(1)=0$ ).

```
lag1LogDifferenceRatioPopulationWalk =
  Rest[Log[randomPopulationWalks[[1]]]] - Most[Log[randomPopulationWalks[[1]]]];
Mean[lag1LogDifferenceRatioPopulationWalk]
-0.0108437

MeanTest[lag1LogDifferenceRatioPopulationWalk, 0, TwoSided → True]
TwoSidedPValue → 0.000613939
```

Rather different than the significance we got from the regression!

```
linFit["ParameterPValues"][[2]]
1.148097462286419 × 10-658
```

Let's do the test for all our random walks:

```
Table[
  lag1LogDifferenceRatioPopulationWalk =
    Rest[Log[randomPopulationWalks[[i]]]] - Most[Log[randomPopulationWalks[[i]]]];
  MeanTest[lag1LogDifferenceRatioPopulationWalk, 0, TwoSided → True],
  {i, 1, Length[randomPopulationWalks]}]
{TwoSidedPValue → 0.000613939, TwoSidedPValue → 0.045809,
  TwoSidedPValue → 0.249659, TwoSidedPValue → 0.169026, TwoSidedPValue → 0.053168,
  TwoSidedPValue → 0.167357, TwoSidedPValue → 0.0929384, TwoSidedPValue → 0.0360712,
  TwoSidedPValue → 0.0339098, TwoSidedPValue → 0.000751115}
```

Some of them are significant, some are not. Certainly a different story from the naive regressions, which were all highly significant.

**Question for the ambitious:** why are so many of the tests for  $\log(\lambda)$  significantly different from 0? If we use  $\alpha = 0.05$ , then we would expect about 1 in 20 to be significant by accident, but there are more significant results than that.

---

## Back to cormorants and fish

So how do we deal with the question of whether fish declines are caused by cormorant increases?

We now know how to estimate the growth parameters of the populations

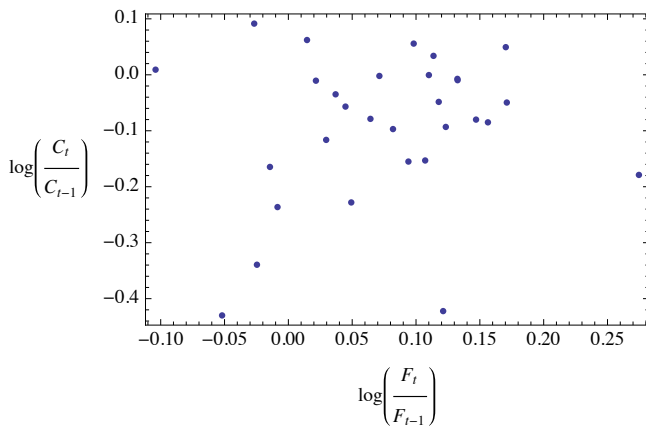
```
cormorantLambda = Mean[Rest[cormorantNumbers] / Most[cormorantNumbers]]
1.08132

fishLambda = Mean[Rest[fishNumbers] / Most[fishNumbers]]
0.919136
```

Sure, one is going up and one is going down, but is that enough? What exactly would we expect to see if cormorants were actually eating fish?

```
TableForm[logRatioData = Transpose[{Log[Rest[cormorantNumbers] / Most[cormorantNumbers]],
  Log[Rest[fishNumbers] / Most[fishNumbers]]}],
```

```
-0.104024  0.00917061
0.123329  -0.0931531
-0.00849702 -0.236358
-0.0246086 -0.339391
-0.0144178 -0.164635
0.0642944 -0.0787304
0.014617  0.062154
0.0712995 -0.00204355
0.0492722 -0.228078
0.170311  0.0493914
0.132419  -0.00724789
0.12125   -0.422245
0.110067  -0.000476744
0.117797  -0.0484037
0.0981778 0.0557593
-0.0519064 -0.429978
0.274602  -0.1789
0.0296231 -0.116299
0.171119  -0.0495497
0.107175  -0.153108
0.0215874 -0.0105196
-0.0267562 0.0914678
0.147021  -0.0800518
0.0940481 -0.155035
0.113694  0.0338526
0.156327  -0.0849458
0.0370019 -0.0348643
0.0819622 -0.096984
0.132565  -0.00963452
0.0447456 -0.056804
```



```
predationTest = LinearModelFit[logRatioData, {x}, {x}]
```

```
FittedModel[[-0.110052 + 0.233322 x]]
```

```
predationTest["ParameterPValues"]
```

```
{0.00315526, 0.464355}
```

No significance.

---

## AR, ARMA, ARIMA, etc.

Removing autocorrelation is relatively easy to do if it is simple. Frequently, it is more complicated. In those case, we model the autocorrelation directly. Here are some more complex autocorrelation models.

### ■ Autoregressive — AR( $p$ ) — model. Autocorrelated *process*.

Variable  $X$  is dependent on values of itself in the past.  $p$  is the *order* of the model — the maximum number of time-steps into the past that matter.

$$X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \epsilon_t.$$

#### ■ AR(1) model

$$X_t = c + \varphi X_{t-1} + \epsilon_t.$$

### ■ Moving average — MA( $q$ ) — model. Autocorrelated *errors*.

Variable  $X$  is dependent on values of some external independent force in the past.  $q$  is the *order* of the model — the maximum number of time-steps into the past that matter.

$$X_t = c + \sum_{i=1}^q \theta_i \epsilon_{t-i} + \epsilon_t.$$

### ■ Auto-regressive moving average — ARMA( $p,q$ ) — model. Autocorrelated *process and errors*.

Variable  $X$  is dependent on values of itself *and* some external independent force in the past.

$$X_t = c + \sum_{i=1}^p \varphi_i X_{t-i} + \sum_{i=1}^q \theta_i \epsilon_{t-i} + \epsilon_t.$$

#### ■ ARMA(1,1) model

$$X_t = c + \varphi X_{t-1} + \theta \epsilon_{t-1} + \epsilon_t.$$

### ■ Auto-regressive integrated moving average — ARIMA( $p,d,q$ ) — model

The integration refers to differencing the data first, which is done to the  $d$ th order.

#### ■ ARIMA(1,1,1) model

$$X_t = c + \varphi X_{t-1} + \phi(X_{t-1} - X_{t-2}) + \theta \epsilon_{t-1} + \epsilon_t.$$

- All simpler models are special cases of the ARIMA model

- Random walk or AR(1) = ARIMA(1,0,0)

$$X_t = c + \varphi X_{t-1} + \epsilon_t.$$

- Removing trends

Generally speaking, these models do not work if there are long-term trends in the data, which are more technically called non-stationarity. To fit them, then, one must remove any trends. Let's look at an example.

- Example

---

This reads in the data.

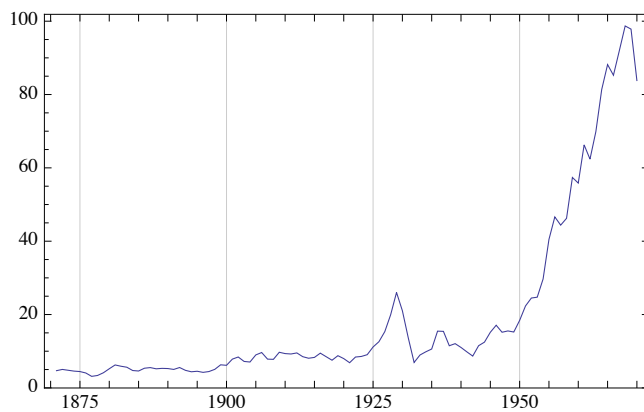
```
csp = ReadList[ToFileName[{"TimeSeries", "Data"}, "csp.dat"], Number];
```

As we have emphasized earlier, the first thing to do in analyzing a time series is to look at its time plot.

---

Here is a plot of the data.

```
DateListPlot[csp, {{1871}, {1970}},  
PlotRange -> All, Joined -> True, AxesLabel -> {"t", "xt"}]
```

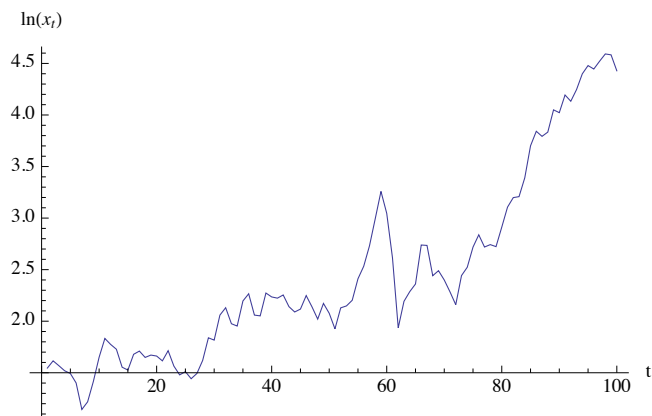


We see that the series rises sharply with time, clearly signaling the nonstationary nature of the series, and indicating the need to transform the data to make it stationary. Here is the time plot of the series after the logarithmic transformation.

---

This plots the transformed data.

```
ListLinePlot[Log[csp], AxesLabel -> {"t", "ln(xt")}]
```



The variability in the series has been reduced. However, a rising trend still persists. We difference the series to obtain a constant mean series.

---

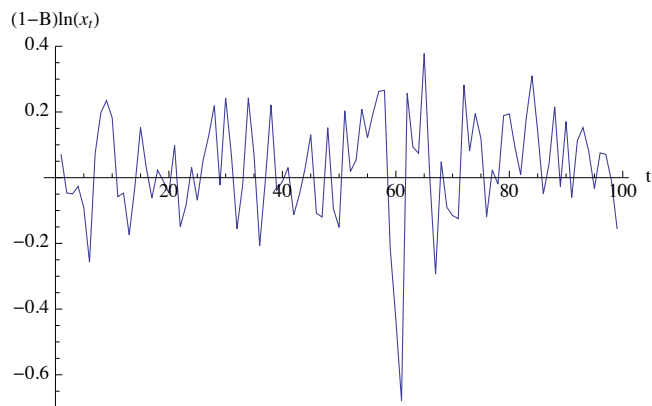
data contains the differenced series.

```
data = ListDifference[Log[csp], 1];
```

---

This is the plot of data.

```
ListLinePlot[data, AxesLabel -> {"t", "(1-B)ln(xt")}]
```



Now that the mean appears steady, we assume the series has a constant mean (the reader is encouraged to devise tests to check this) and evaluate it.

---

This gives the mean.

```
Mean[data]
```

```
0.0291236
```

We subtract this sample mean from the series to make it a zero-mean series.

---

The data are transformed to a zero-mean series.

```
data = data - Mean[data];
```

---

The length of the data is n.

```
n = Length[data]
```

```
99
```

Now we can fit `data` to a stationary time series model. In order to select an appropriate model, we first look at its correlation and partial correlation functions.

---

This gives the sample correlation function.

```
corr = CorrelationFunction[data, 20];
```

---

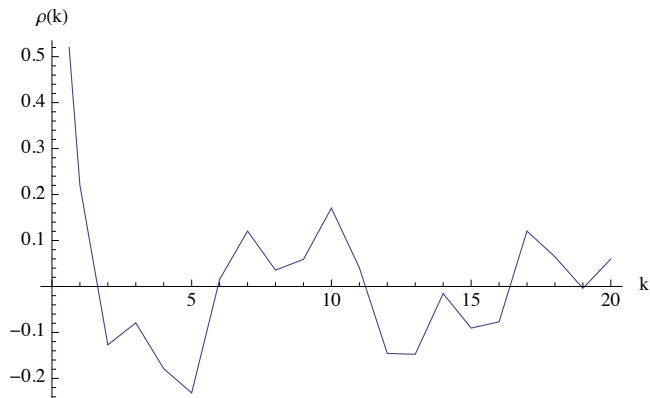
To plot the correlation function, we redefine the function `plotcorr` here.

```
plotcorr[corr_, opts___] := ListPlot[corr, DataRange -> {0, Length[corr] - 1}, opts]
```

---

This is the correlation plot.

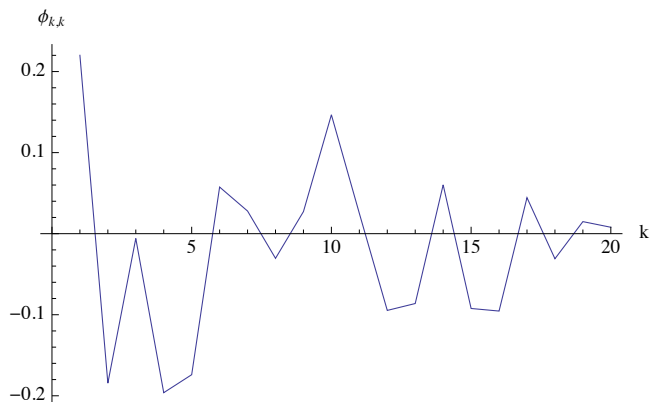
```
plotcorr[corr, Joined -> True, AxesLabel -> {"k", " $\rho(k)$ "}]
```




---

Here the sample partial correlation function is plotted.

```
ListLinePlot[PartialCorrelationFunction[data, 20],  
  AxesLabel -> {"k", " $\phi_{k,k}$ "}, PlotRange -> All]
```



We see that both the correlation and partial correlation functions decay quickly to small values. This indicates that the series can be fitted to models of relatively low order. We use the Hannan-Rissanen procedure to select models and get preliminary estimates.

---

These are five models selected by the Hannan-Rissanen procedure, which uses BIC (a variant of AIC) to find the  $n$  most parsimonious models.

```
HannanRissanenEstimate[data, 10, 6, 6, 5]
{ARModel[{0.223037}, 0.0246806], MAModel[{0.218863}, 0.0250601],
 ARModel[{0.265727, -0.1874}, 0.0239958], MAModel[{0.231584, -0.153218}, 0.0246679],
 ARMAModel[{0.523774, -0.255097}, {-0.306381}, 0.0238409]}
```

These selected models with their estimated parameters are used as input to the function that performs the conditional maximum likelihood estimate.

---

The five models are further estimated using the conditional maximum likelihood method.

```
models = ConditionalMLEstimate[data, #] & /@ %
{ARModel[{0.223037}, 0.0246806], MAModel[{0.30951}, 0.0239137],
 ARModel[{0.265727, -0.1874}, 0.0239958], MAModel[{0.251582, -0.131713}, 0.0235999],
 ARMAModel[{0.937435, -0.318558}, {-0.718786}, 0.0233558]}
```

We obtain the AIC values for the five models, AR(1), MA(1), AR(2), MA(2), and ARMA(2, 1), specified above.

---

This gives the AIC values.

```
AIC[#, n] & /@ %
{-3.68154, -3.7131, -3.68947, -3.70611, -3.6963}
```

We see that the lowest AIC values correspond to MA(1) and MA(2) models. We choose these two models for further exploration using the maximum likelihood method.

---

This gives the maximum likelihood estimate of the MA(1) model.

```
ma1 = MLEstimate[data, MAModel[{ $\theta_1$ }, 1], { $\theta_1$ , {0.3, 0.305}}]
MAModel[{0.309508}, 0.0239101]
```

---

This is the AIC value of the MA(1) model.

```
AIC[% , n]
-3.71325
```

---

This gives the maximum likelihood estimate of the MA(2) model.

```
ma2 = MLEstimate[data, MAModel[{ $\theta_1$ ,  $\theta_2$ }, 1], { $\theta_1$ , {0.25, 0.255}}, { $\theta_2$ , {-0.13, -0.125}}]
MAModel[{0.256541, -0.122621}, 0.0235922]
```

---

This is the AIC value of the MA(2) model.

```
AIC[% , n]
-3.70643
```

Again MA(1) is superior according to the AIC criterion. Note that in this particular example, the estimated parameters using the maximum likelihood method are very close to those obtained using the conditional maximum likelihood method. We now tentatively choose MA(1) as our model and proceed to check its adequacy.

From (5.6) the variance of the sample correlation for an MA(1) model for lags  $k > 1$  can be calculated.

---

This calculates the variance of the sample correlation.

```
(1 + 2 corr[[2]] ^ 2) / n
```

```
0.0110795
```

---

Here is the bound.

```
2 Sqrt[%]
```

```
0.210518
```

The sample correlation function shown above is within this bound with the exception of  $\hat{\rho}(5)$ . So the behavior of the sample correlation function obtained from the data is consistent with the MA(1) model. Next we calculate the residuals and see if they form a white noise process.

---

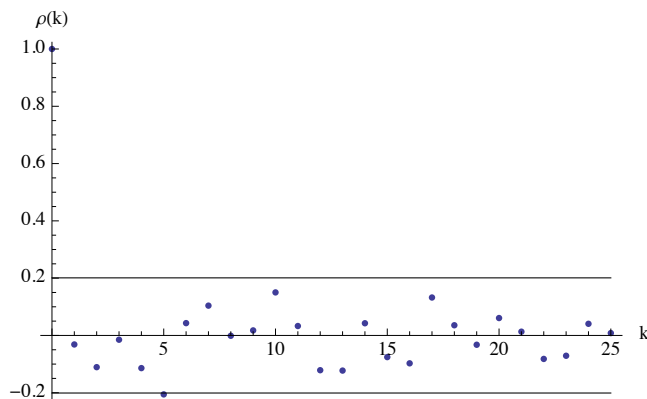
The residuals are calculated first.

```
res = Residual[data, mal];
```

---

The residual correlation function is displayed along with the bounds.

```
Show[plotcorr[CorrelationFunction[res, 25],  
Plot[{-2 / Sqrt[n], 2 / Sqrt[n]}, {x, 0, 25}, PlotStyle -> {{Black}}],  
PlotRange -> All, AxesLabel -> {"k", " $\rho(k)$ "}]
```




---

## Spatial autocorrelation

### ■ Distance matrices

Can be geographic distance or anything you like (even something like species community dissimilarity).

- **Moran's I**

Simple correlation coefficient between pairs of values in some defined distance class. Produces a correlogram similar to the ACF plots above.

- **Mantel Test**

Permuting data in one of two distance matrices to test relationship.

- **Partial Mantel Test**

Testing for a relationship between two distance matrices after factoring out a third. For example: “How much of the variability in species composition is explained by the environmental matrix?” and “Is there residual variability in species composition that is spatially structured, after removing the effects of the environmental variables?”

---

## **Phylogenetic autocorrelation!**

Just a different kind of space.

Distance matrices and Moran's I still work.

We know something about evolution, so we can use that.