Stationary Time Series

Kyoung Hwan Lee

First of all, we need to define what is Stationary Time Series and figure out why we study Stationary Time Series. Even before that, what is time series? According to the textbook, a time series is a set of obserations x_t, each one being recorded at a specific time t. There are two types of Time Series. One is a *discrete time series* and the other is *continuous time series.* If the observations are over a time interval that is fixed, it's discrete and if the observations are recorded continously over some time interval, it's continuous.
Now, definition 1.3.1 from the textbook says that a **time series model** for the observed data $\{x_1\}$ is a specification of the joint distributions (or possibly only the means and covariances) of a sequence of random variables $\{X_t\}$ of which $\{x_t\}$ is postulated to be a realization.

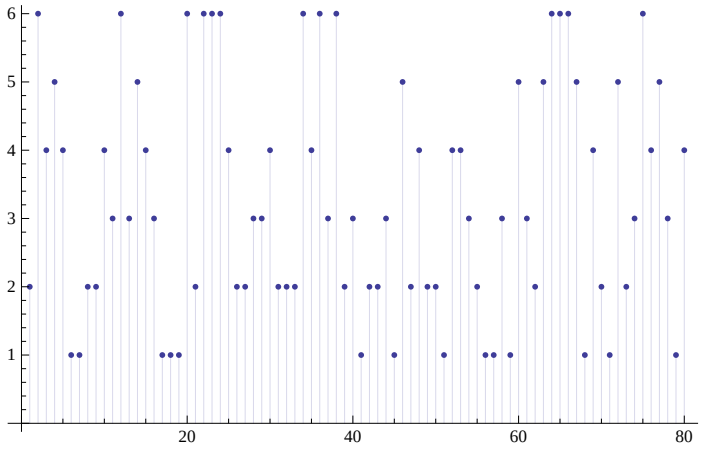
To state if the time series is stationary or not, we need to define the mean function and the covariance function first.
Definition 1.4.1
Let $\{X_t\}$ be a time series with $E\left(X_t^2\right) < \infty$. The **mean function** of $\{X_t\}$ is
$\mu_X(t) = E(X_t)$

The c**ovariance function of** $\{X_t\}$ is
$\gamma_X(r, s) = \text{Cov}(X_r, X_s) = E[(X_r - \mu_X(r))(X_s - \mu_X(s))]$
for all integers r and s.

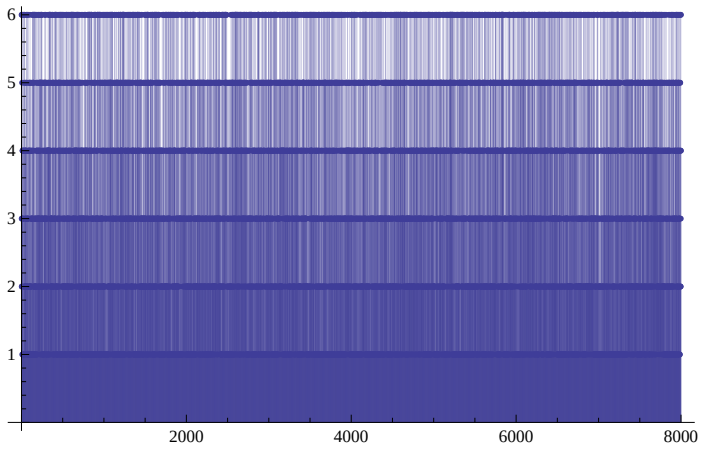
To figure out if the time series is stationary or not, we need to use definition 1.4.2 saying
$\{X_t\}$ is **(weakly) stationary** if
(i) $\mu_X(t) = E(X_t)$.
and
(ii) $\gamma_X(t + h, t)$ is independent of t for each h.

Noise

```
(* sample time series of a roll of a dice with 6 faces. Rolled 80 times *)
data = RandomVariate[DiscreteUniformDistribution[{1, 6}], 80];
ListPlot[data, Filling → Axis]
```




```
(* Dice rolled many more times, the horizontal bands are exactly 1, 2, 3, 4, 5, 6 *)
data = RandomVariate[DiscreteUniformDistribution[{1, 6}], 8000];
ListPlot[data, Filling → Axis]
```



Now, Calculate the mean and variance of the distribution.

```
{Mean[data], Variance[data]} // N
```

```
{3.51025, 2.88551}
```

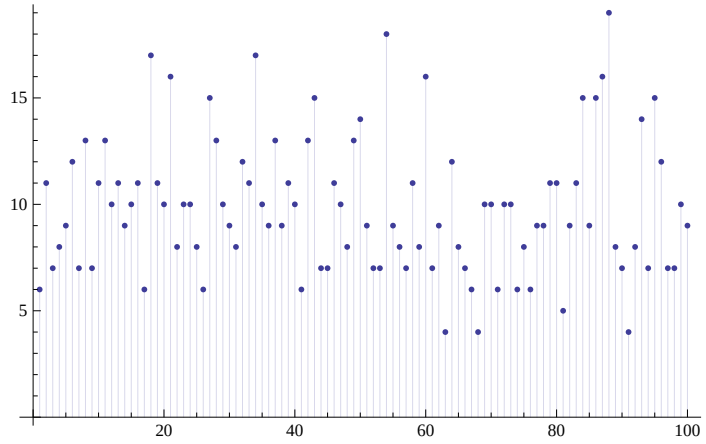
h is the sliding window of the model that determines whether the stationary definition of the time series holds or not. the code Length[data]-h and

```
t = 30;
(*timeshift by h *)
h = 1000;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N

(* change this time see if the covariance changes *)
t = 190;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N

t = 590;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N
```

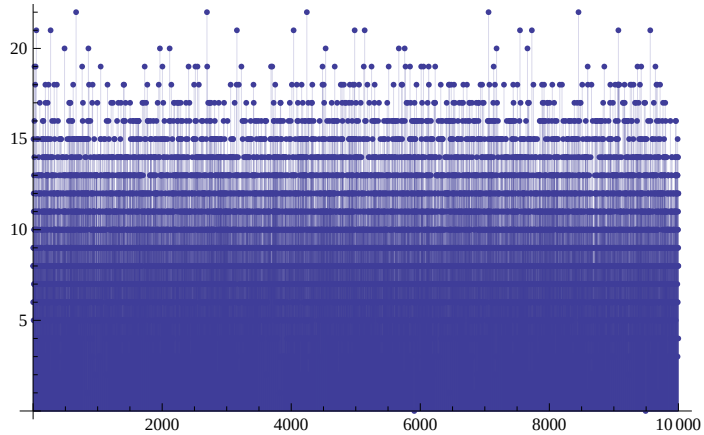
```
{3.50868, -0.0327035}

{3.51446, -0.0307901}

{3.51349, -0.0245833}
```

Let’s use Poisson instead of constant distribution for the dice

```
data = RandomVariate[PoissonDistribution[10], 100];
ListPlot[data, Filling → Axis]
```




```
(* Increase the number of samples in time series *)
data = RandomVariate[PoissonDistribution[10], 10^4];
ListPlot[data, Filling → Axis]
```



We do the same procedure as above. Calculate the mean and the variance of the data and vary t as we give any number of h to see if the sample of time series is stationary or not. As we change the time, we can observe whether the covariance changes to check if the series is stationary.

```
{Mean[data], Variance[data]} // N
```

```
{9.9531, 9.64767}
```

```
t = 30;
(*timeshift by h *)
h = 1000;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N

(* change this time see if the covariance changes *)
t = 190;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N

t = 590;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N
```

```
{9.95909, -0.182421}
```

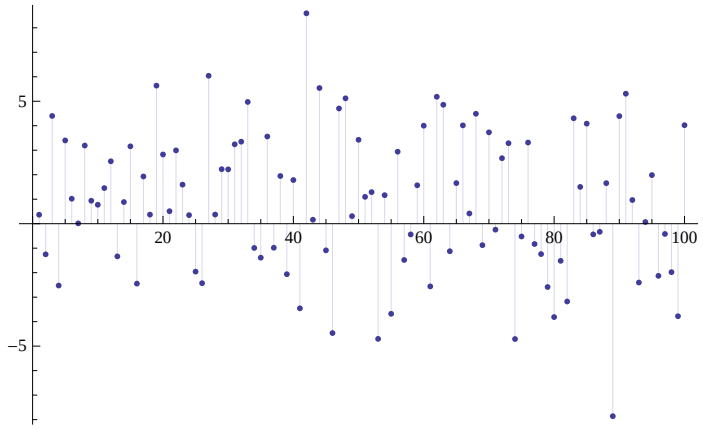
```
{9.96391, -0.190257}
```

```
{9.97206, -0.184724}
```

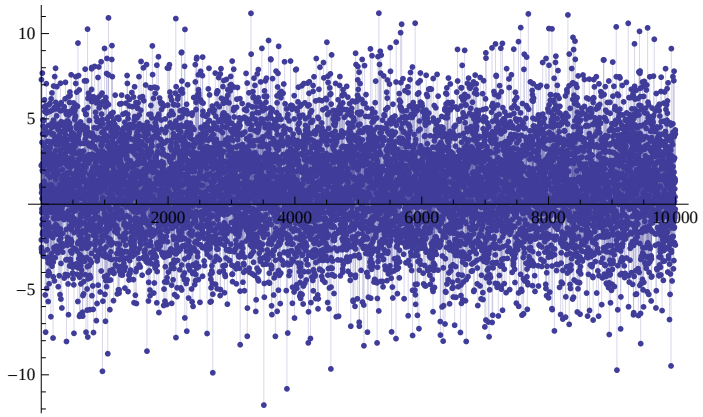
Now let's use another time series which is obtained frmo Normal distrubution, instead of rolling dice. Mean = 1 and Var = 3

We will have one plot that has small sample and another with large sample to see the pattern.

```
data = RandomVariate[NormalDistribution[1, 3], 100];
ListPlot[data, Filling → Axis]
```




```
(* Now choose much large sample for time series *)
data = RandomVariate[NormalDistribution[1, 3], 10^4];
ListPlot[data, Filling → Axis]
```



Once again, we repeat the procedure as above, calculate mean and variance and the covariance.

```
{Mean[data], Variance[data]} // N
```

```
{0.994628, 9.1332}
```

```
t = 900;
(*timeshift by h *)
h = 1000;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N

(* change this time see if the covariance changes *)
t = 190;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N

t = 590;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N
```

```
{0.981389, -0.0439288}

{0.977056, 0.00563836}

{0.964315, -0.0181857}
```

All of the samples above had reasonably small changes to their mean and covariance as the time changed which we can conclude that all of them were (weakly) stationary models.

Random Walk

To study the random walk, it's a good idea to cover the textbook definition first.
Example 1.3.3 Random Walk
The random walk $\{S_t,\ t = 0,\ 1,\ 2,\ ...\}$(starting at zero) is obtained by cumulatively summing (or "integrating") iid random variables. Thus a random walk with zero mean is obtained by defining $S_0 = 0$ and $S_t = X_1 + X_2 + ... + X_t$, for $t = 1,\ 2,\ ...$ where $\{X_t\}$ is iid noise. If $\{X_t\}$ is the binary process of example 1.3.2, then $\{S_t,\ t = 0,\ 1,\ 2,\ ...\}$ is called a simple symmetric random walk. This walk can be viewed as the location of a pedestrian who starts at position zero at time zero and at each integer time tosses a fair coin, stepping one unit to the right each time a head appears, and one unit to the left for each tail. A realization of length 200 of a simple symmetric random walk is shown in Figure 1.7. Notice that the outcomes of the coin tosses can be recovered from $\{S_t,\ t = 0,\ 1,\ 2,\ ...\}$ by differencing, THus the result of the tth toss can be found from $S_t - S_{t-1} = X_t$.

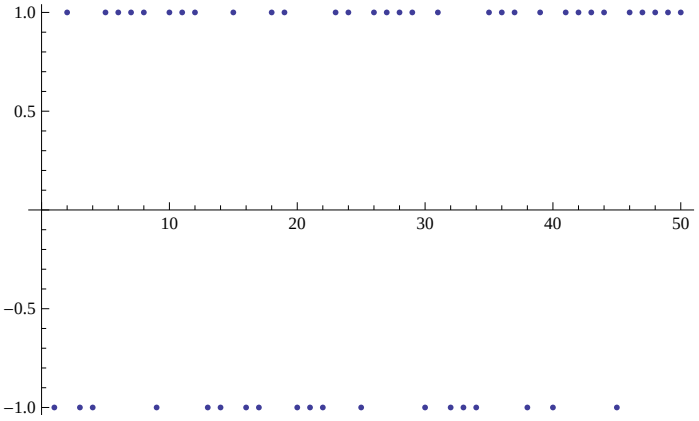
this is the example that covers the random walk.
Example 1.4.3 The Random Walk
If $\{S_t\}$is the random walk defined in Example 1.3.3 with $\{X_t\}$ as in Example 1.4.1 then $ES_t = 0,\ E(S_t^2) = t\sigma^2 < \infty$ for all t, and for h≥0,

$$\begin{aligned}\gamma_t(t+h,\ t) &= \mathrm{Cov}(S_{t+h},\ S_t)\\ &= \mathrm{Cov}(S_t + X_{t+1} + ... + X_{t+h,}\ S_t)\\ &= \mathrm{Cov}(S_t + S_t)\\ &= t\sigma^2.\end{aligned}$$

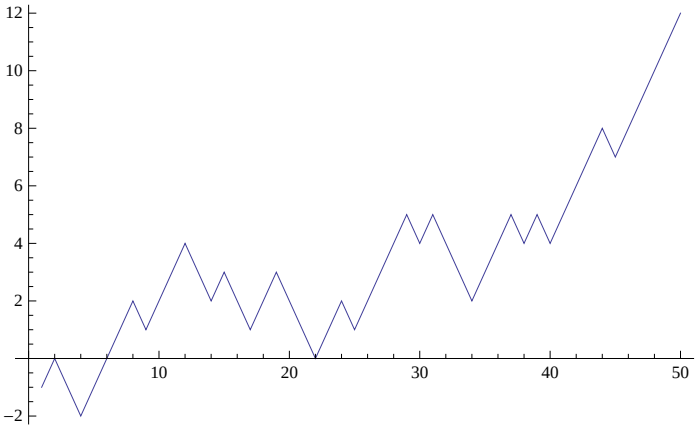
Since $\gamma_s(t+h,\ t)$ depends on t, series $\{S_t\}$is not stationary.

We now start with our own example. The model starts walking with flip a coin and H means go right +1 and T means go left -1. Take a step each time after the toss.
Lets start with a 50 tosses.

```
(* start walking but at each step flip a coin, H go right +1, T go left -1, *)
data = RandomChoice[{-1, 1}, 50];
dataaccum = Accumulate[data];
ListPlot[data]
ListLinePlot[dataaccum]
```

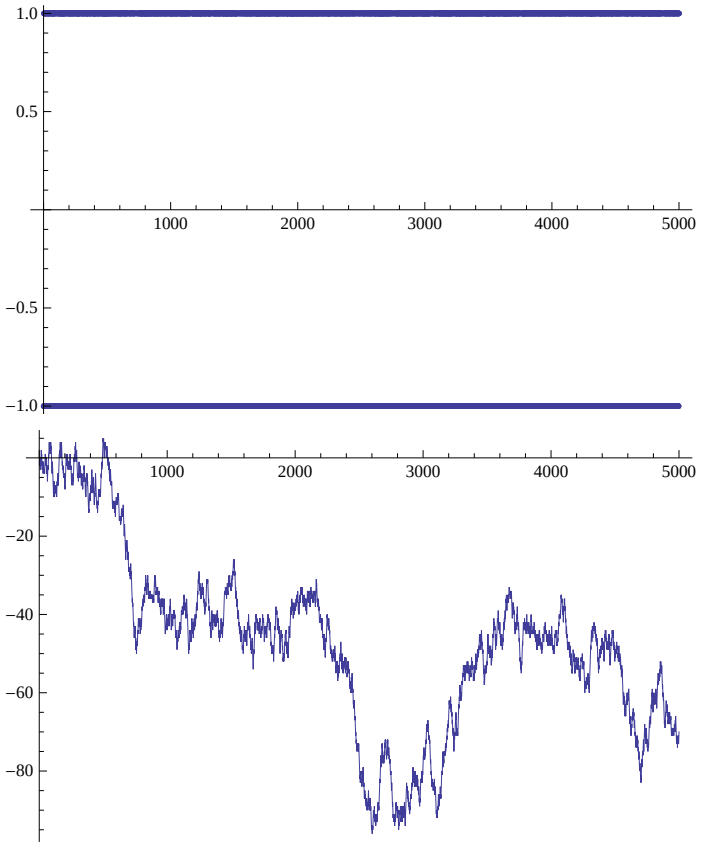


This graph gives you the distribustion of the accumlative steps you took from the origin:



Now that we have some distribution in 50 trials, lets try the large sample, 5000.

```
data = RandomChoice[{-1, 1}, 5000];
dataaccum = Accumulate[data];
ListPlot[data]
ListLinePlot[dataaccum]
```



We can see some trends on the graph above. Now we calculate the mean and covariance with fixed h in different times.

```
{Mean[data], Variance[data]} // N
```

```
{-0.014, 1.}
```

```
data = dataaccum;
t = 30;
(*timeshift by h *)
h = 150;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N

(* change this time see if the covariance changes,
as you can see it does!!! So random walk is NOT stationary *)
t = 90;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N

t = 190;

datatimet = data[[t ;; Length[data] - h]];

datatimetplush = data[[t + h ;; Length[data]]];
{Mean[datatimet], Covariance[datatimet, datatimetplush]} // N
```

```
{-46.619, 415.854}
```

```
{-47.1918, 394.477}
```

```
{-48.1304, 358.725}
```

The result shows that the mean and variance vary as the time changes. Therefore, Random Walk is not a stationary model.

Auto Correlation

Autocorrelation function is the function we can use to help finding the covariance of the time series. Here is the definition.
Definition 1.4.3 The autocorrelation function (ACF) of $\{X_t\}$ at lag h is
$\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = \text{Cor}(X_{t+h}, X_t)$.
Also, the definition 1.4.4 gives the definition of the sample correlation function
$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}, \ -n < h < n.$

The autocorrealtion function has to be constant in order to be independent of t. If it is independent of t, the series is stationary.

So we assign a function naming “autoCorrelation” as below.

```
(* Auto Correlation ρX(h). However the time t >
 0 is added for the sake of emphasis and in case the time series is not stationary.
   This function is actually called SAMPLE Auto Correlation *)
autoCorrelation[data0_, h0_, t0_] :=
 Module[{data = data0, h = h0, t = t0, datatimet, datatimetplush, cov1h, cov10},

  datatimet = data[[t ;; Length[data] - h]];

  datatimetplush = data[[t + h ;; Length[data]]];
  cov1h = Covariance[datatimet, datatimetplush];

  h = 0;
  datatimet = data[[t ;; Length[data] - h]];
  datatimetplush = data[[t + h ;; Length[data]]];
  cov10 = Covariance[datatimet, datatimetplush];

  cov1h / cov10 // N
 ]
```

We use the dice rolling example used above to see the correlation works on identifying the stationariness.

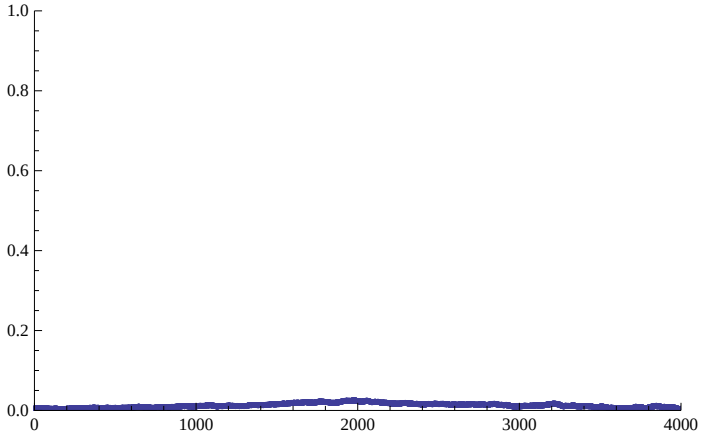
```
(* Dice rolled many more times, the horizontal bands are exactly 1, 2, 3, 4, 5, 6 *)
data = RandomVariate[DiscreteUniformDistribution[{1, 6}], 8000];
{Mean[data], Variance[data]} // N
```

```
{3.52625, 2.93818}
```

```
autoCorrelation[data, 900, 600]
```

```
0.01509
```

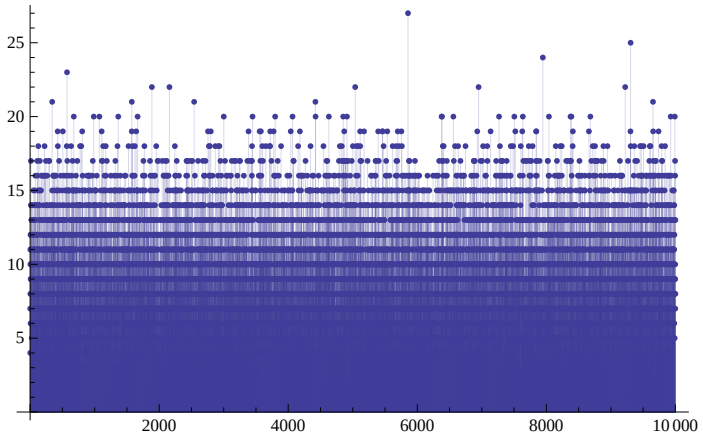
```
h = 1000;
ListPlot[Table[autoCorrelation[data, h, t], {t, 1, Ceiling[Length[data] / 2]}],
 PlotRange → {{0, 4000}, {0, 1}}]
```



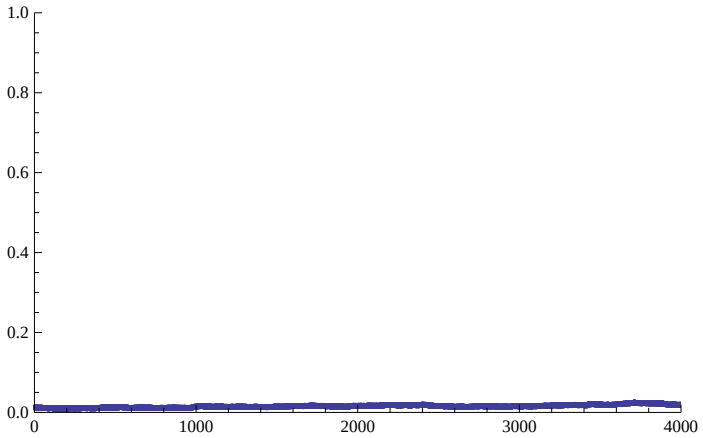
Both the value of the autocorrelation and the plot shows that it's constant. Therefore, the series is stationary.

We use the poisson distribution once more.

```
data = RandomVariate[PoissonDistribution[10], 10^4];
ListPlot[data, Filling → Axis]
```



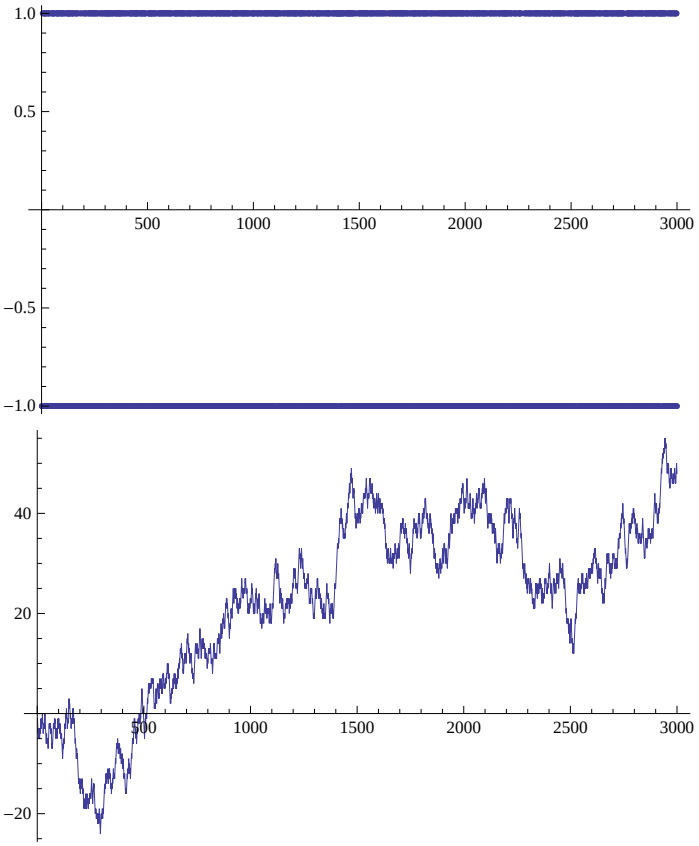

```
h = 1000;
ListPlot[Table[autoCorrelation[data, h, t], {t, 1, Ceiling[Length[data] / 2]}],
 PlotRange → {{0, 4000}, {0, 1}}]
```



the autocorrelation is constant again, thus stationary.

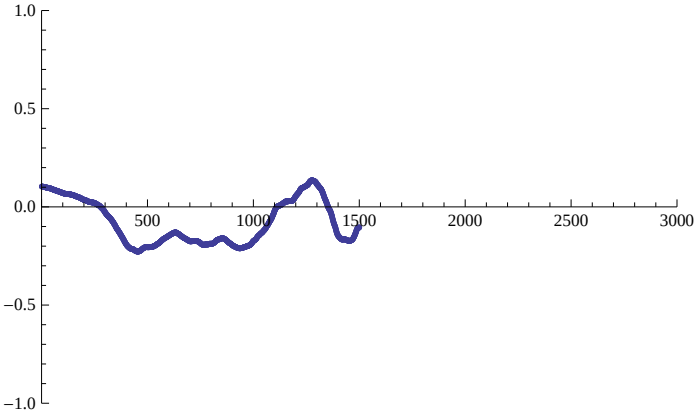
Create another random walk

```
data = RandomChoice[{-1, 1}, 3000];
dataaccum = Accumulate[data];
ListPlot[data]
ListLinePlot[dataaccum]
```



As you can see its Auto Correlation function is not constant in time t

```
h = 1000;
ListPlot[Table[autoCorrelation[dataaccum, h, t],
  {t, 1, Ceiling[Length[dataaccum] / 2]}], PlotRange → {{0, Length[dataaccum]}, {-1, 1}}]
```



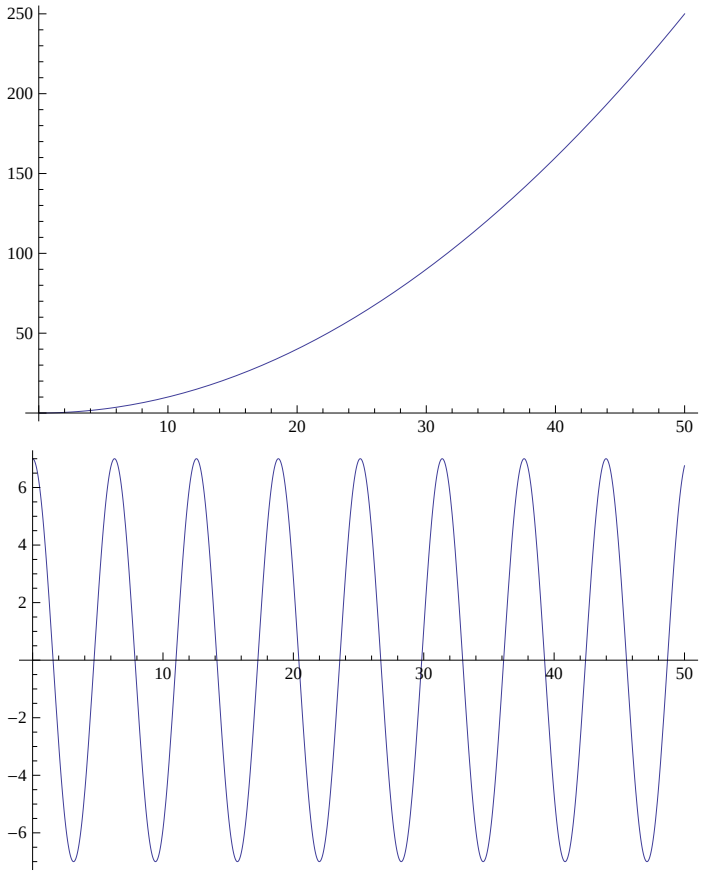
Classical Decomposition Model

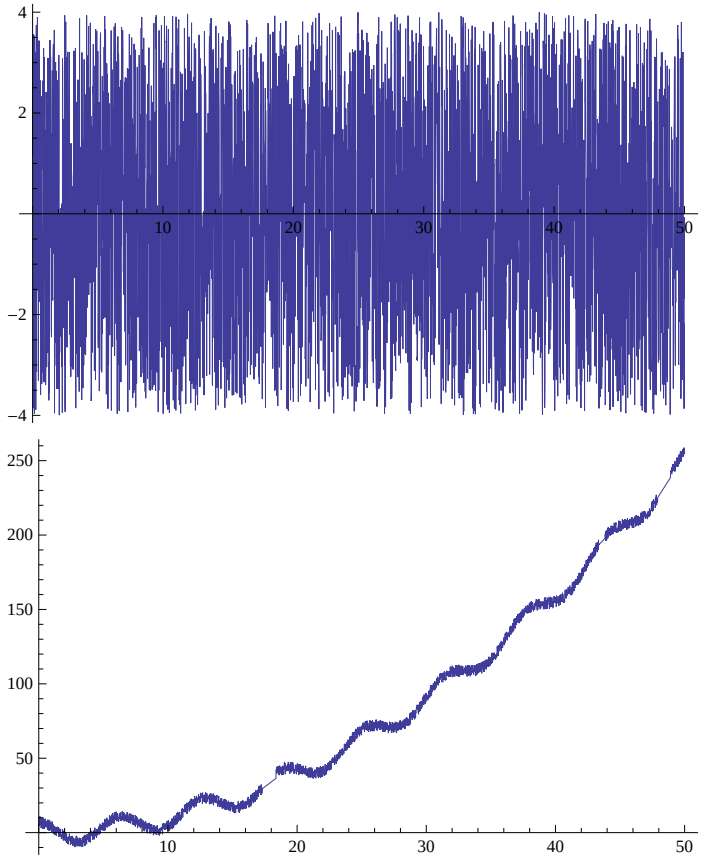
Section 1.5 emphasize on the estimation and elimination of trend and seasonal components.
First to study is a classical decomposition model.

Classical Decomposition Model
$X_t = m_t + s_t + Y_t,\ t = 1, 2, \ldots, n,$
where $EY_t = 0,\ s_{t+d} = s_t,$ and $\sum_{j=1}^{d} s_j = 0.$

The below plots are some of the example models with 1. slowly changing trend, seasonal component (periodic), and random noise. and the last one is the composition of all of those functions.

```
t = .;
m =.;
s =.;
X =.;
Y =.;
(* Trend, slowly changing function *)
m = 0.1 * t^2;
(* Seasonal Component, preodic cyclic *)
s = 7 * Cos[t];
(* Random Noise *)
Y = HoldForm[4 * RandomReal[{-1, 1}]];

X[w_] := (m + s + ReleaseHold[Y]) /. {t → w};

Plot[m, {t, 0, 50}]
Plot[s, {t, 0, 50}]
Plot[ReleaseHold[Y], {t, 0, 50}]
Plot[X[t], {t, 0, 50}]
```





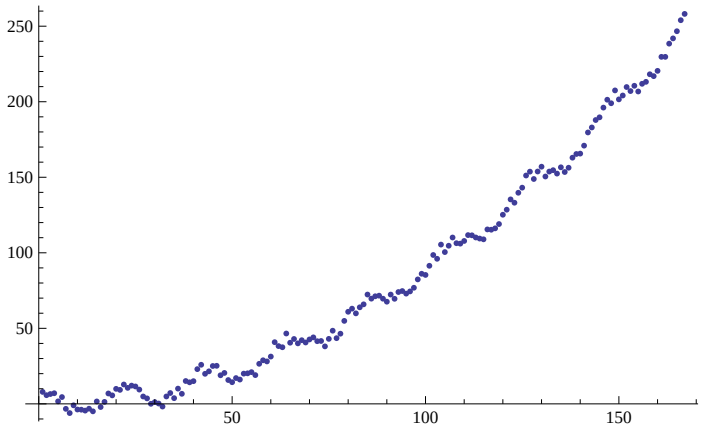
Smoothing

Sometimes, we need to smooth the time series to get a better sense of the trend because some people might get confused with the noise and the trend. To reduce the chance of error, lets study some of the examples of the smoothing. Firt of all, we assign a Xseries which is the sequence with difference of 0.3.

```
(* Grab a signal and set the time sequence different at 0.3.
  Look at the data to get an idea. *)
```

```
Xseries = Table[X[t], {t, 0, 50, 0.3}]
ListPlot[Xseries]
```

```
{7.91046, 5.7433, 6.51046, 7.03326, 1.64816, 4.50115, -3.29423, -6.10054, -0.968383,
 -3.74727, -3.857, -4.45881, -3.21314, -4.94624, 1.59646, -2.09578, 1.31484, 6.88267,
 5.62245, 9.9411, 9.25711, 12.8161, 10.5945, 12.0768, 11.5219, 9.44579, 4.87985, 3.62356,
 0.0410051, 1.15545, 0.222748, -1.77104, 4.89443, 7.16179, 3.77486, 10.0885, 6.65814,
 15.1217, 14.2563, 15.0318, 22.9859, 25.872, 19.8936, 21.5232, 25.0721, 25.1917, 18.9285,
 20.4744, 15.7677, 14.3381, 17.0802, 16.0887, 20.0152, 20.2112, 20.9959, 19.0454,
 26.5174, 28.8499, 28.0909, 31.3099, 40.8288, 38.2378, 37.4786, 46.5717, 40.4775,
 42.9514, 40.0472, 42.1362, 40.6273, 42.6415, 44.0122, 41.4832, 41.6209, 38.046,
 43.0223, 48.4272, 43.4861, 46.4494, 54.9282, 60.9636, 63.0211, 59.9061, 63.9164,
 65.8792, 72.3648, 69.6464, 71.2134, 71.6194, 69.6155, 67.6311, 72.3429, 69.5481,
 74.0273, 74.6787, 72.9778, 74.4403, 76.8592, 82.39, 86.1208, 85.3421, 91.4059, 98.5367,
 96.0306, 105.433, 100.477, 104.673, 110.122, 106.352, 106.068, 107.833, 111.715,
 111.572, 110.139, 109.422, 108.924, 115.475, 115.276, 116.176, 119.025, 125.192,
 128.538, 135.342, 133.146, 139.748, 143.115, 151.16, 153.721, 148.851, 153.811,
 156.999, 150.498, 153.77, 154.682, 152.43, 156.596, 153.471, 156.309, 162.958, 165.399,
 165.657, 170.93, 179.661, 182.96, 187.884, 189.681, 196.107, 201.328, 198.998, 207.487,
 201.562, 204.125, 209.725, 207.057, 210.56, 206.778, 211.869, 213.163, 218.181,
 216.931, 220.439, 229.712, 229.708, 238.448, 241.926, 246.667, 253.963, 258.135}
```



Moving Average

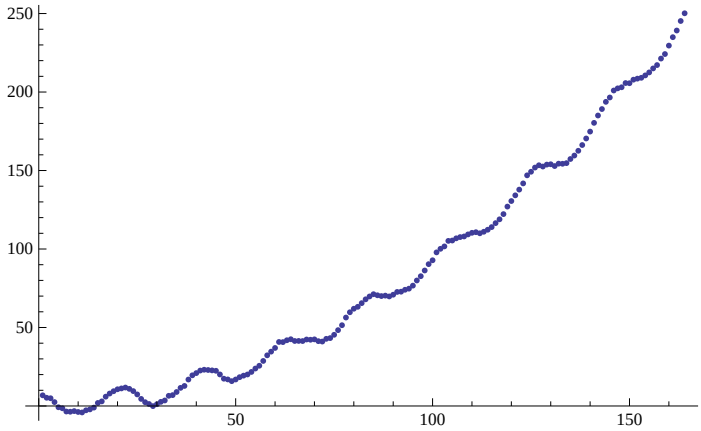
The first method to cover is the moving average. This is the definition from the book.
Smoothing with a finite moving average filter.
Let q be a nonnegative integer and consider the two-sided moving average
$W_t = (2\,q + 1)^{-1} \sum_{j=-q}^{q} X_{t-j}$
of the process $\{X_t\}$ defined by $X_t = m_t + Y_t,\ t = 1,\ ...,\ n,$ where $\mathrm{E}Y_t = 0.$ Then for $q + 1 \le t \le n - q,$
$W_t = (2\,q + 1)^{-1} \sum_{j=-q}^{q} m_{t-j} + (2\,q + 1)^{-1} \sum_{j=-q}^{q} Y_{t-j} \approx m_t,$
assuming that m_t is approximately linear over the interval $[t - q,\ t + q]$ and that the average of the error terms over this interval is close to zero.
The moving average thus provides us with the estimates
$\hat{m}_t = (2\,q + 1)^{-1} \sum_{j=-q}^{q} X_{t-j},\ q + 1 \le t \le n - q.$

So we perform the moving average below to see if there is any change to the plot.

```
MovingAverage[{a, b, c, d, e, f, g}, 3]
```

$$\left\{\frac{1}{3}(a+b+c), \frac{1}{3}(b+c+d), \frac{1}{3}(c+d+e), \frac{1}{3}(d+e+f), \frac{1}{3}(e+f+g)\right\}$$

```
movingaverage = MovingAverage[Xseries, 4];
ListPlot[movingaverage]
```



The plot flows better than the original graph.

Exponential Moving Average (Smoothing)

`ExponentialMovingAverage[x, α]` generates a list of results y in which $y_{i+1} = y_i + \alpha(x_{i+1} - y_i)$.

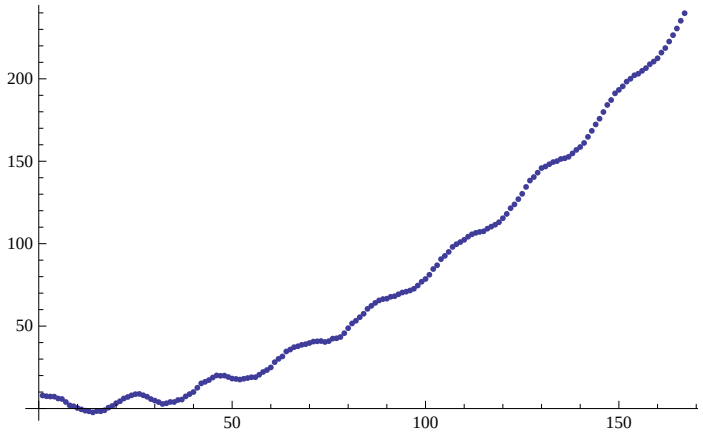
Let study the definition of the exponential smoothing.
For any fixed $\alpha \in [0,1]$, the one sided moving average $\hat{m}_t$, $t = 1, \ldots, n$, defined by the recursions

$$\hat{m}_t = \alpha X_t + (1-\alpha)\,\hat{m}_{t-1},\ t = 2, \ldots, n, \text{ and } \hat{m}_1 = X_1$$

```
ExponentialMovingAverage[{a, b, c}, α]
```

$$\{a,\ a + (-a+b)\,\alpha,\ a + (-a+b)\,\alpha + \alpha\,(-a+c-(-a+b)\,\alpha)\}$$

```
movingaverage = ExponentialMovingAverage[Xseries, .2];
ListPlot[movingaverage]
```



Differencing

If *list* has length *s*, `Differences [list, n]` has length $s - n$.

We can also eliminate the trend with the method of differencing.
According to the textbook,
Instead of attempting to remove the noise by smoothing we now attempt to eliminate the trend by differencing.
So this is a different method from smoothing techniques as above. We define the lag-1 difference operator ∇ by
$\nabla X_t = X_t - X_{t-1} = (1 - B) X_t$, where B is the backward shift operator,
$BX_t = X_{t-1}$.

```
Differences[{a, b, c, d, e}]
```

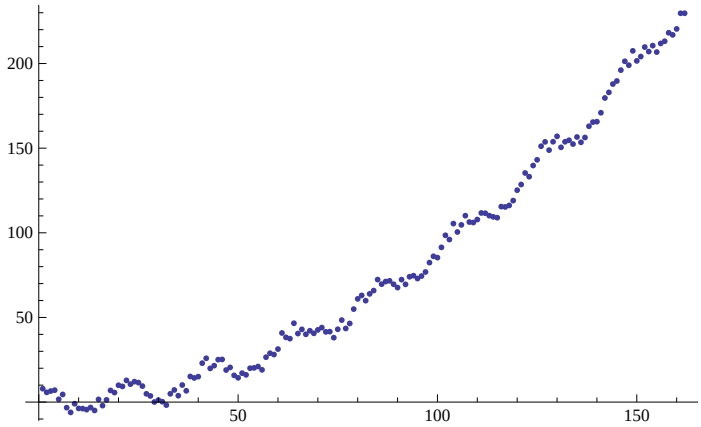
```
{-a + b, -b + c, -c + d, -d + e}
```

```
Differences[{a, b, c, d, e}, 2]
```

```
{a - 2 b + c, b - 2 c + d, c - 2 d + e}
```

```
power = 5
diff = Differences[Xseries, power];
ListPlot[Drop[Xseries, -power]]
```

5



By applying the the differencing method, the graph is detrended even though it's not as smooth as the smoothing techniques.